zenon Treiber Handbuch

BACnetNG

v.7.10

© 2013 Ing. Punzenberger COPA-DATA GmbH

Alle Rechte vorbehalten.

Die Weitergabe und Vervielfältigung dieses Dokuments ist - gleich in welcher Art und Weise – nur mit schriftlicher Genehmigung der Firma COPA-DATA gestattet. Technische Daten dienen nur der Produktbeschreibung und sind keine zugesicherten Eigenschaften im Rechtssinn. Änderungen – auch in technischer Hinsicht - vorbehalten.

Inhalt

1. Willkommen bei der COPA-DATA Hilfe 5

2. BACnetNG 5

3. BACNETNG - Datenblatt 6

4. Treiber-Historie 8

5. Voraussetzungen 8

5.1 PC 8

5.2 Steuerung 8

6. Konfiguration 9

6.1 Anlegen eines Treibers 9

6.2 Einstellungen im Treiberdialog 11

6.2.1 Allgemein 11

6.2.2 Settings 15

6.2.3 Devices 17

7. Variablen anlegen 20

7.1 Variablen im Editor anlegen 20

7.2 Adressierung 23

7.3 Treiberobjekte und Datentypen 50

7.3.1 Treiberobjekte 50

7.3.2 Zuordnung der Datentypen 55

7.4 Variablen anlegen durch Import 57

7.4.1 XML Import 57

7.4.2 DBF Import/Export 58

7.4.3 Online-Import 64

7.5 Treibervariablen 67

8. Treiberspezifische Funktionen 72

8.1 Zugriffsverfahren (spontanes und pollendes Lesen) 73

8.2 Abbildung der BACnet Status-Flags auf die Status-Bits der Variablen 73

8.3 Abbildung von BACnet Datentypen auf Stringvariablen 74
8.4 Device Management 78
8.5 Import 79

9. Treiberkommandos 79

10. Fehleranalyse 81

10.1 Analysetool 81
10.2 BACnet Fehlercodes 83
10.3 Checkliste 87

11. PICS - Protocol Implementation Conformance Statement 87

12. Kommunikation mit der Steuerung 91

1. Willkommen bei der COPA-DATA Hilfe

ALLGEMEINE HILFE

Falls Sie in diesem Hilfekapitel Informationen vermissen oder Wünsche für Ergänzungen haben, wenden Sie sich bitte per E-Mail an documentation@copadata.com (mailto:documentation@copadata.com).

PROJEKTUNTERSTÜTZUNG

Unterstützung bei Fragen zu konkreten eigenen Projekten erhalten Sie vom Support-Team, das Sie per E-Mail an support@copadata.com (mailto:support@copadata.com) erreichen.

LIZENZEN UND MODULE

Sollten Sie feststellen, dass Sie weitere Module oder Lizenzen benötigen, sind unsere Mitarbeiter unter sales@copadata.com (mailto:sales@copadata.com) gerne für Sie da.

2. BACnetNG

BACNET/IP CLIENT TREIBER

Das BACnet-Protokoll wurde von der ASHRAE (American Society of Heating, Refrigeration and Air-Conditioning Engineers, Inc.) definiert, und wird im ASHRAE Standard 135-2001 + Appendix A – L „A Data Communication Protocol for Building Automation and Control Networks" ausführlich beschrieben. BACnet/IP wird im Appendix J spezifiziert und beschreibt die BACnet Kommunikation über IP/UDP Telegramme.

Dieser Treiber ermöglicht die Kommunikation zwischen einem oder mehreren BACnet-fähigen Geräten (BACnet-Automatisierungsstationen) und der zenon Runtime über BACnet/IP. Dabei wird vorausgesetzt, dass die angeschlossenen BACnet-Geräte als Server betrieben werden. Im Treiber ist nur Client-Funktionalität implementiert.

Das BACnet Protokoll definiert Objekte und Objekt-Eigenschaften. Der Treiber ermöglicht das Lesen und Schreiben einer oder mehrerer beliebiger Eigenschaften eines Objekts. Grundsätzlich wird sowohl pollendes Lesen als auch spontane Kommunikation über COV (Change-of-value) Subscriptions unterstützt. Allerdings unterstützen die meisten Geräte spontane Kommunikation nur für die Eigenschaften `PRESENT-VALUE` und `STATUS-FLAGS`.

Die Daten einer BACnet Eigenschaft werden in so genannte **`BACnet Tags`** übertragen. Eine Eigenschaft kann einen oder mehrere Tags enthalten. Ein Tag besteht aus **`Tag-Klasse`**, **`Tag-Nummer`** und einem Wert.

Für die Applikation-Tagklasse geht aus der Tagnummer eindeutig der Datentyp des enkodierten Wertes hervor.

Für Tags mit der Klasse `kontextspezifisch` ist der Datentyp zusätzlich von der Eigenschaft und vom Objekttyp abhängig, d.h. für die Dekodierung/Enkodierung des Wertes ist Wissen über Eigenschaft und Objekt bzw. über den Datentyp der Eigenschaft notwendig.

Eigenschaften, die aus einem Applikation-Tag bestehen können auch auf primitive Typen abgebildet werden. Alle anderen Eigenschaften können nur auf Stringvariablen abgebildet werden.

3. BACNETNG - Datenblatt

Allgemein:	
Treiberdateiname	BACNETNG.exe
Treiberbezeichnung	BACnet Treiber Next Generation
Steuerungs-Typen	Alle Steuerungen mit BACnet/IP Unterstützung
Steuerungs-Hersteller	Siemens; Kieback + Peter; BACnet;

Treiber unterstützt:	
Protokoll	BACnet/IP;
Adressierung: Adress-basiert	-
Adressierung: Namens-basiert	x
Kommunikation spontan	x
Kommunikation pollend	x
Online Browsing	x
Offline Browsing	-
Echtzeitfähig	-
Blockwrite	-
Modemfähig	-
Serielles Logging	-
RDA numerisch	-
RDA String	-

Voraussetzungen:	
Hardware PC	Standard Netzwerkkarte
Software PC	-
Hardware Steuerung	-
Software Steuerung	-
Benötigt v-dll	x

Plattformen:	
Betriebssysteme	Windows CE 5.0, CE 6.0; Windows Vista, 7, 8, Server 2008/R2, Server 2012;
CE Plattformen	x86; ARM;

4. Treiber-Historie

Datum	Treiberversion	Änderung
27.07.09	100	Treiberdokumentation wurde neu erstellt

5. Voraussetzungen

Dieses Kapitel enthält Informationen zu den Voraussetzungen, die für die Verwendung des Treibers erforderlich sind.

5.1 PC

Zur BACnet/IP Kommunikation wird eine IP-Netzwerkverbindung mit Unterstützung für das UDP Protokoll benötigt.

5.2 Steuerung

Steuerungen müssen das BACnet Protokoll unterstützen. Wird eine Steuerung nicht über BACnet/IP mit der zenon Runtime verbunden, muss ein entsprechender BACnet-Router verwendet werden.

6. Konfiguration

In diesem Kapitel lesen Sie, wie Sie den Treiber im Projekt anlegen und welche Einstellungen beim Treiber möglich sind.

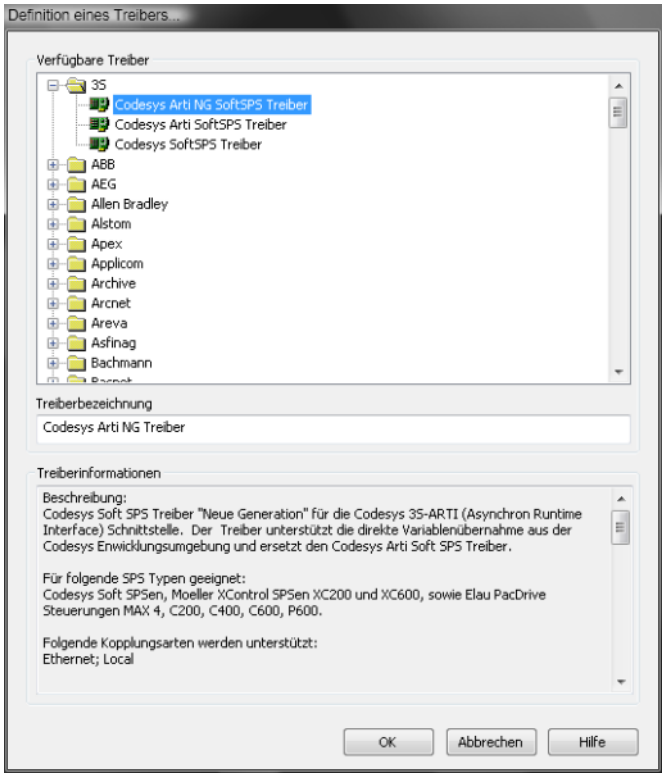
> **Info**
>
> *Weitere Einstellungen, die Sie für Variablen in zenon vornehmen können, finden Sie im Kapitel Variablen (main.chm::/15247.htm) der Online-Hilfe.*

6.1 Anlegen eines Treibers

Um einen neuen Treiber anzulegen:

1. Klicken Sie mit der rechten Maustaste im Projektmanager auf **`Treiber`** und selektieren Sie im Kontextmenü **`Treiber neu`**.

2. In der folgenden Dialogbox bietet Ihnen das Programm eine Auflistung aller verfügbaren Treiber an.



3. Selektieren Sie den gewünschten Treiber und vergeben Sie eine Bezeichnung für diesen:
   - Die Treiberbezeichnung muss eindeutig sein, d.h. wird ein und derselbe Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
   - Die Treiberbezeichnung ist Bestandteil des Dateinamens. Daher darf Sie nur Zeichen enthalten, die vom Betriebssystem unterstützt werden. Nicht gültige Zeichen werden durch einen Unterstrich (_) ersetzt.
   - Achtung: Die Bezeichnung kann später nicht mehr geändert werden.
4. Bestätigen Sie den Dialog mit ok. Im folgenden Dialog werden die einzelnen Konfigurationen der jeweiligen Treiber eingestellt.

Für ein Projekt müssen nur die jeweils notwendigen Treiber eingebunden werden. Späteres Einbinden eines weiteren Treibers ist problemlos möglich.

Info

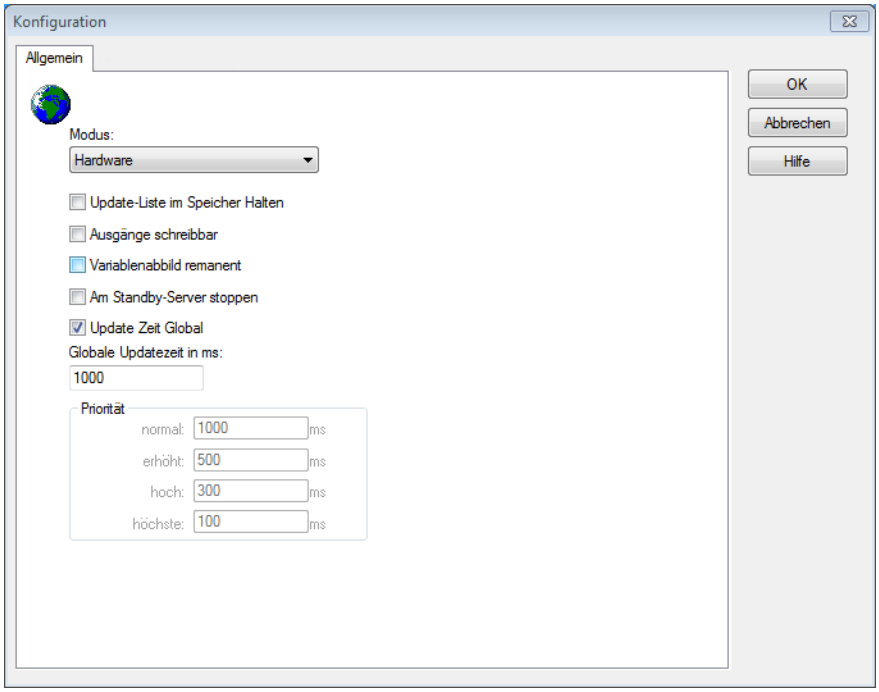
Bei neu angelegten Projekten oder bei bestehenden Projekten, die in eine Version ab 6.21 konvertiert werden, werden die folgenden Treiber automatisch angelegt:

- Intern
- MathDr32
- SysDrv.

6.2 Einstellungen im Treiberdialog

Folgende Einstellungen können Sie beim Treiber vornehmen:

6.2.1 Allgemein



Parameter	Beschreibung
Modus	Ermöglicht ein Umschalten zwischen Hardware und Simulationsmodus ▸ Hardware: Die Verbindung zur Steuerung wird hergestellt. ▸ Simulation - statisch Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert. In diesem Modus bleiben die Werte konstant bzw. die Variablen behalten die über zenon Logic gesetzen Werte. Jede Variable hat seinen eigenen Speicherbereich. Zum Beispiel zwei Variablen vom Typ Merker mit Offset 79, können zur Runtime unterschiedliche Werte haben und beeinflussen sich gegenseitig nicht. Ausnahme: Der Simulatortreiber. ▸ Simulation - zählend Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert. In diesem Modus zählt der Treiber die Werte innerhalb ihres Wertebereichs automatisch hoch. ▸ Simulation - programmiert Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden von einem frei programmierbaren Simulationsprojekt berechnet. Das Simulationsprojekt wird mit der zenon Logic Workbench erstellt und läuft in einer in den Treiber integrierten zenon Logic Runtime ab. Details siehe Kapitel Treibersimulation. (main.chm::/25206.htm)
Update-Liste im Speicher Halten	Einmal angeforderte Variablen werden weiterhin von der Steuerung angefordert, auch wenn diese aktuell nicht mehr benötigt werden. Dies hat den Vorteil, dass z B. mehrmalige Bildumschaltungen nach dem erstmaligen Aufschalten beschleunigt werden, da die Variablen nicht neu angefordert werden müssen. Der Nachteil ist eine erhöhte Belastung der Kommunikation zur Steuerung.
Ausgänge schreibbar	Aktiv: Ausgänge können beschrieben werden. Inaktiv: Das Beschreiben der Ausgänge wird unterbunden.

	Hinweis: Steht nicht für jeden Treiber zur Verfügungen.
Variablenabbild remanent	Diese Option speichert und restauriert den aktuellen Wert, den Zeitstempel und die Status eines Datenpunkts. Grundvoraussetzung: Die Variable muss einen gültigen Wert und Zeitstempel besitzen. Das Variablenabbild wird im Modus Hardware gespeichert wenn: ▸ einer der Status S_MERKER_1(0) bis S_MERKER8(7), REVISION(9), AUS(20) oder ERSATZWERT(27) aktiv ist Das Variablenabbild wird immer gespeichert wenn: ▸ die Variable vom Objekttyp Treibervariable ist ▸ der Treiber im Simulationsmodus läuft. (nicht programmierte Simulation) Folgende Status werden beim Start der Runtime nicht restauriert: ▸ SELECT(8) ▸ WR-ACK(40) ▸ WR-SUC(41) Der Modus **Simulation - programmiert** beim Treiberstart ist kein Kriterium, um das remanente Variablenabbild zu restaurieren.
Am Standby-Server stoppen	Einstellung für Redundanz bei Treibern, die nur eine Kommunikationsverbindung erlauben. Dazu wird der Treiber am Standby-Server gestoppt und erst beim Hochstufen wieder gestartet. Achtung: Ist diese Option aktiv, ist die lückenlose Archivierung nicht mehr gewährleistet. Aktiv: Versetzt den Treiber am nicht-prozessführenden Server automatisch in einen Stopp-ähnlichen Zustand. Im Unterschied zum Stoppen über Treiberkommando erhält die Variable nicht den Status **abgeschaltet** (**statusverarbeitung.chm::/24150.htm**), sondern einen leeren Wert. Damit wird verhindert, dass beim Hochstufen zum Server nicht relevante Werte in AML , CEL und Archiv erzeugt werden.
Update Zeit Global	Aktiv: Die eingestellte Globale Update Zeit in ms wird für alle Variablen im Projekt verwendet. Die bei den Variablen eingestellte Priorität

	wird nicht verwendet. Inaktiv: Die eingestellten Prioritäten werden für die einzelnen Variablen verwendet.
Priorität	Hier werden die Pollingzeiten der einzelnen Prioritäten eingestellt. Alle Variablen mit der entsprechenden Priorität werden in der eingestellten Zeit gepollt. Die Zuordnung zu den Variablen erfolgt separat bei jeder Variablen über die Einstellungen in den Variableneigenschaften. Mit den Prioritäten kann die Kommunikation der einzelnen Variablen auf die Wichtigkeit bzw. benötigte Aktualität abgestuft werden. Daraus ergibt sich eine verbesserte Verteilung der Kommunikationslast.
OK	Übernimmt Einstellungen aus allen Registerkarten und schließt den Dialog.
Abbrechen	Verwirft alle Änderungen und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

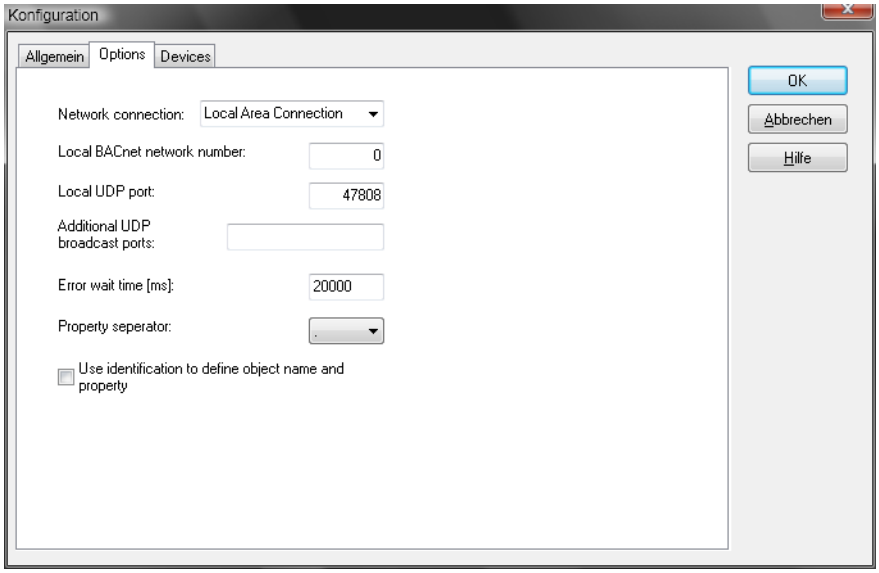
UPDATE ZEIT ZYKLISCHE TREIBER

Für zyklische Treiber gilt:

Beim **Sollwert Setzen**, **Advicen** von Variablen und bei **Requests** wird sofort ein Lesezyklus für alle Treiber ausgelöst - unabhängig von der eingestellten Update Zeit. Damit wird sicher gestellt, dass der Wert nach dem Schreiben in der Visualisierung sofort zur Verfügung steht. Update-Zeiten können damit für zyklische Treiber kürzer ausfallen als eingestellt.

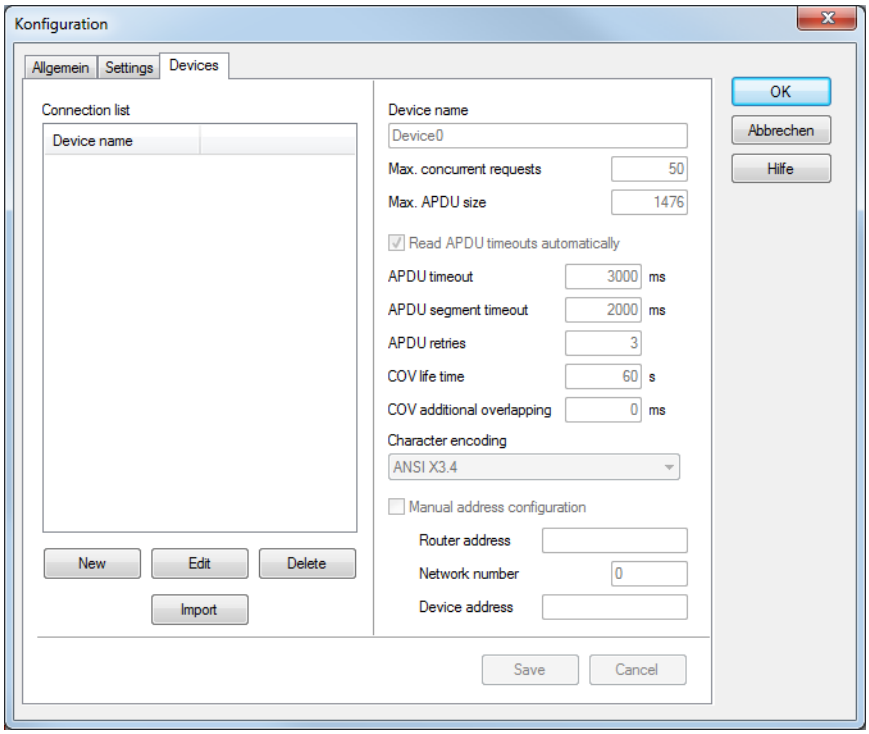
6.2.2 Settings

ALLGEMEINE EINSTELLUNGEN GÜLTIG FÜR ALLE KONFIGURIERTEN BACNET GERÄTE



Eigenschaft	Beschreibung
Network connection	Name der Netzwerkverbindung, die für die BACnet/IP Kommunikation genützt werden soll. Der Name entspricht dem Namen der unter Systemsteuerung -> Netzwerkverbindungen angezeigt wird. Die Dropdownliste enthält die Namen aller am System vorhanden Verbindungen. Alternativ kann auch ein beliebiger Name eingeben werden. Wird kein Name angegeben, wird der erste Netzwerkadapter des Systems verwendet. Bei Systemen mit mehreren Netzwerkkarten sollte immer der Verbindungsname eingetragen werden.
Local BACnet network number	BACnet Netzwerknummer des lokalen BACnet Netzes.
Local UDP port	Lokaler UDP Port, der für die Kommunikation verwendet werden soll Default: 47808
Additional UDP broadcast ports	Zusätzliche UDP Ports (neben dem Standard BACnet/IP Port BAC0h bzw. 47808), an die BACnet/IP Broadcasts gerichtet werden sollen.
Error wait time [ms]	Fehlerwartezeit in Millisekunden.
Property seperator	Trennzeichen in Variablenname bzw. Kennung, um den Objektnamen vom Eigenschaftennamen zu trennen.
Use identification to define object name and property	Aktiv: Der Gerätename, der Objektname und der Eigenschaftenname werden nicht aus dem Variablennamen sondern in der Kennung definiert.

6.2.3 Devices


Konfiguration
Allgemein
Settings
Devices
Connection list
Device name
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

LISTE DER BACNET GERÄTE UND GERÄTESPEZIFISCHE EINSTELLUNGEN

Eigenschaft	Beschreibung
Connection list	Listet alle konfigurierten Geräte auf.
New	Fügt ein neues Gerät in die Liste ein.
Delete	Löscht das selektierte Gerät aus der Liste.
Edit	Bearbeiten eines existierenden Geräts.
Import	Durchsucht Netzwerk nach vorhandenen Geräten und importiert diese in die Device-Liste oder gibt eine entsprechende Meldung aus, wenn keine Geräte gefunden werden.
Gerätespezifische Einstellungen	
Device name	Name des BACnet-Geräts (Name des Device-Objekts).
Max. concurrent requests	Maximale Anzahl gleichzeitig ausstehender Requests. Minimum: 1 Mximum: 256 Default: 5 Wenn ein Gerät mehr Requests als eingestellt sendet, werden solange neue Puffer allokiert bis das theoretische Maximum von 256 erreicht wird. Hinweis: Aus Performancegründen sollte die Einstellung auf das Maximum des verbundenen Geräts konfiguriert werden. Nur wenn das Gerät auch mit anderen Geräten kommuniziert, sollte eventuell ein kleinerer Wert eingestellt werden.
Max. APDU size	Maximal übertragbare Größe einer APDU bzw. eines APDU Segments. Default: 1476
Read APDU timeouts automatically	Aktiv: Die APDU Timeout Einstellungen werden vom Gerät gelesen - das heißt, es werden die am Gerät eingestellten Timeouts übernommen.
APDU timeout [ms]	Timeout für die Bestätigung einer Anfrage. Default 3000
APDU segment	Timeout für die Bestätigung eines bzw. mehrer Segmente einer

timeout [ms]	Anfragen. Default 2000
APDU retries	Anfrage Wiederholungen im Fall eines Timeouts. Default: 3
COV life time [s]	Lebenszeit von COV Subscriptions. Default: 60
COV additional overlapping [ms]	Zusätzlicher Parameter zur Festlegung, wann die erneuerte Subscription abgeschickt wird. Verhindert bei verzögerter BACnet Reaktion den Verlust von gesendeten Werten durch Timeout. Default: 0 Formel: (APDU_timeout*(APDU_retries+1))-COV_additional_overlapping
Character encoding	Verwendeter Zeichensatz. Auswahl aus Dropdownliste: ▸ ANSI X3.4 ▸ ISO 10646 (UCS-4) ▸ ISO 10646 (UCS-2) ▸ ISO 8859-1
Manual address configuration	Aktiv: Die Adresse wird nicht über Broadcast und den Gerätenamen automatisch bestimmt, sondern kann über Router address, Network number und Device address manuell angegeben werden.
Router address	BACnet MAC Adresse des Routers. Diese besteht aus der IP Adresse des Routers und dem UDP Port im Format z. B. 192.168.0.5:47808.
Network number	BACnet Netzwerknummer des Geräts.
Device address	BACnet MAC Adresse des Geräts. Für BACnet/IP Geräte bestehend aus IP Adresse und UDP Port (siehe

	`Router address`). Für alle anderen BACnet Geräte als durch `:` getrennte Bytefolge im Hexformat (z. B. `06:0A:67:EE`).
`Save`	Änderungen speichern.
`Cancel`	Änderungen verwerfen.

7. Variablen anlegen

So werden Variablen im zenon Editor angelegt:

7.1 Variablen im Editor anlegen

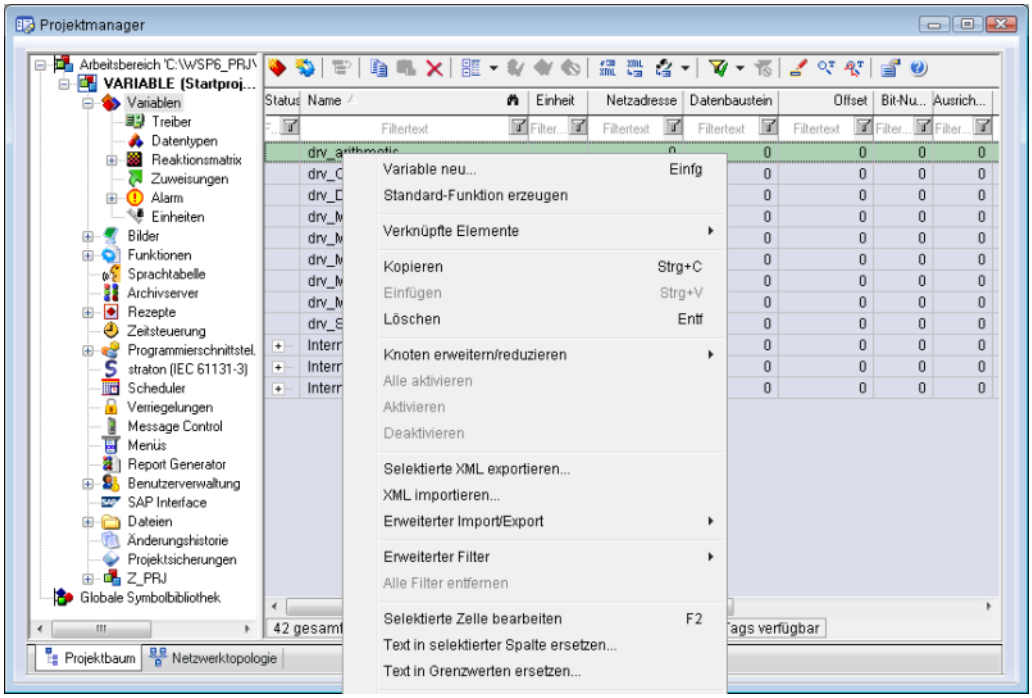
Variablen können angelegt werden:

- als einfache Variable
- in Arrays (main.chm::/15262.htm)
- als Struktur-Variablen (main.chm::/15278.htm)

DIALOG VARIABLE

Um eine neue Variable zu erstellen, gleich welchen Typs:

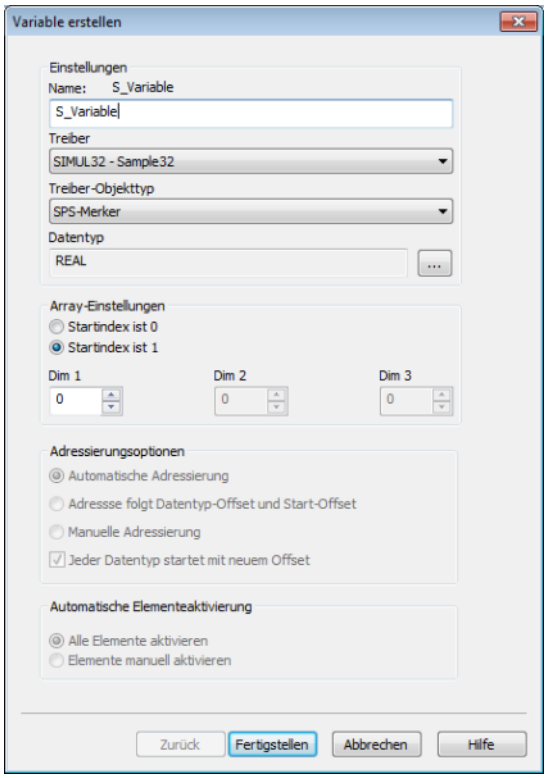
1. wählen Sie im Knoten **`Variablen`** im Kontextmenü den Befehl **`Variable neu`**



2. der Dialog zur Konfiguration der Variable wird geöffnet

3. konfigurieren Sie die Variable

4. welche Einstellungen möglich sind, hängt ab vom Typ der Variablen



Eigenschaft	Beschreibung
Name	Eindeutiger Name der Variablen. Ist eine Variable mit gleichem Namen im Projekt bereits vorhanden, kann keine weitere Variable mit diesem Namen angelegt werden. Maximale Länge: 128 Zeichen Achtung: Das Zeichen **#** ist im Variablennamen nicht zugelassen. Bei Verwendung nicht zugelassener Zeichen kann die Variablenerstellung nicht abgeschlossen werden, die Schaltfläche **Fertigstellen** bleibt inaktiv.
Treiber	Wählen Sie aus der Combobox den gewünschten Treiber. Hinweis: Sollte im Projekt noch kein Treiber angelegt sein, wird automatisch der Treiber für interne Variable (Intern.exe (Main.chm::/Intern.chm::/Intern.htm)) geladen.
Treiber-Objekttyp (cti.chm::/28685.htm)	Wählen Sie aus der Dropdownliste den passenden Treiber-Objekttyp aus.

Datentyp	Wählen Sie den gewünschten Datentyp. Klick auf die Schaltfläche ... öffnet den Auswahl-Dialog.
Array-Einstellungen	Erweiterte Einstellungen für Array-Variablen. Details dazu lesen Sie im Abschnitt Arrays.
Adressierungsoptionen	Erweiterte Einstellungen für Arrays und Struktur-Variablen. Details dazu lesen Sie im jeweiligen Abschnitt.
Automatische Elementeaktivierung	Erweiterte Einstellungen für Arrays und Struktur-Variablen. Details dazu lesen Sie im jeweiligen Abschnitt.

ABLEITUNG VOM DATENTYP

Messbereich, Signalbereich und Sollwert Setzen werden immer:

- vom Datentyp abgeleitet
- beim Ändern des Datentyps automatisch angepasst

Hinweis Signalbereich: Bei einem Wechsel auf einen Datentyp, der den eingestellten Signalbereich nicht unterstützt, wird der Signalbereich automatisch angepasst. Zum Beispiel wird bei einem Wechsel von **INT** auf **SINT** der Signalbereich auf 127 geändert. Die Anpassung erfolgt auch dann, wenn der Signalbereich nicht vom Datentyp abgeleitet wurde. In diesem Fall muss der Messbereich manuell angepasst werden.

7.2 Adressierung

Gruppe/Eigenschaft	Beschreibung
Allgemein	
Name	Name der Variablen bestehend aus Gerätename, Objektname und Eigenschaftenname. Geräte und Objektname werden durch einen Punkt (**.**)getrennt. Das Trennzeichen zwischen Objekt und Eigenschaftenname ist konfigurierbar. Soll ein einzelnes Item eines Arrays adressiert werden, muss das Offset als Postfix in eckigen Klammern angeben werden. (z. B.: **MyServer.AnalogValue7#priority-array[8]**) Achtung: Je zenon Projekt muss der Name eindeutig sein. Auch das Device-Objekt eines Geräts wird nach obigem Schema adressiert. In

	diesem Fall wiederholt sich der Gerätename als Objektname (z. B. **`MyServer.MyServer#object-name`**).
Kennung	Kann alternativ zum Namen zur Adressierung der Variablen verwendet werden, wenn dies in der Treiberkonfiguration eingestellt wird. Das Format für Name und Kennung ist identisch.
Adressierung	
Netzadresse	Busadresse oder Netzadresse der Variablen. Diese Adresse bezieht sich auf die Busadresse der Verbindungsprojektierung im Treiber. Damit wird ausgewählt, auf welcher Steuerung sich die Variable befindet.
Datenbaustein	Wird für diesen Treiber nicht verwendet.
Offset	Wird für diesen Treiber nicht verwendet.
Ausrichtung	Wird für diesen Treiber nicht verwendet.
Bit-Nummer	Bitoffset innerhalb des adressierten Items. Mögliche Eingabe: `0` ... `65535` Achtung: Variablen mit einer Bitnummer ungleich `0` können nicht geschrieben werden.
Stringlänge	Nur verfügbar bei String-Variablen: Maximale Anzahl von Zeichen, die die Variable aufnehmen kann.

Treiber Anbindung/-	Abhängig vom verwendeten Treiber wird bei der Erstellung der Variablen ein Objekttyp ausgewählt und kann hier später geändert werden.
Treiber Anbindung/Datentyp	Datentyp der Variablen. Wird beim Erstellen der Variablen ausgewählt und kann hier später verändert werden. Achtung: Wenn der Datentyp nachträglich geändert wird, müssen alle anderen Eigenschaften der Variablen überprüft bzw. angepasst werden.
BACnet data type	BACnet Datentyp der Eigenschaft. Anhand des Datentyps wird festgelegt wie eine numerische Variable oder ein String auf eine BACnet Eigenschaft abgebildet wird. Eine Beschreibung der möglichen Datentypen ist in der folgenden Tabelle enthalten.
BACnet property ID	Für benutzerspezifische Eigenschaften oder wenn die Eigenschaft nicht in Name oder Kennung angeben werden soll, kann hier die Property ID eingestellt werden. Achtung: Nur wenn `-1` als Property ID eingestellt ist, wird die Eigenschaft über Name bzw. Kennung ermittelt.
BACnet string length	Nur verfügbar, wenn als `BACnet data type` `BITSTRING` ausgewählt wird. Gibt die Bitlänge des Bitstrings an.
BACnet write priority	Schreibpriorität. Gültige Schreibprioritäten sind `1 - 16`. Wird `0` eingestellt, wird die Variable ohne eine Angabe der Priorität geschrieben. Mögliche Eingabe: `0` ... 16
Priority	Normal: Spontanes lesen über COV-Subscriptions Achtung: Nur für `PRESENT-VALUE` und `STATUS-FLAGS` zur Angabe des Update-Zyklus laut Treiberkonfiguration. Mögliche Werte: ▸ erhöht ▸ hoch ▸ höchste

VOM TREIBER UNTERSTÜTZTEN DATENTYPEN FÜR BACNET OBJEKT-EIGENSCHAFTEN

BACnet Datentyp	Beschreibung	Kompatible IEC Typen	String Kodierungsbeispiel
NULL	Datentyp ohne Wert	String	[0]
BOOLEAN	Boolean	BOOL, String	TRUE

UNSIGNED	Positive Ganzzahl	UDINT, String	8
SIGNED	Ganzzahl	DINT, String	-1
REAL	Gleitkommazahl	REAL, String	7.9
DOUBLE	Gleitkommazahl	LREAL, String	8.0
OCTETSTRING	Bytefolge	String	65A8B900
CHARACTERSTRING	Zeichenfolge	String	abcd
BITSTRING	Bitfolge	String, UDINT (wenn Länge < 32)	01110
ENUMERATED	Enumeration	UDINT, String	8
DATE	Datum	UDINT, String	109.12.1.7
TIME	Uhrzeit	UDINT, String	17:00:00:000
OBJECTIDENTIFIER	Objekt ID	String	0008 0000001
[Any]	Beliebiger Wert	String	[2] 78
[DateTime]	Datum (DATE) gefolgt von Uhrzeit (TIME)	String	{[10] 109.12.1.7},{[11] 17:00:00:000}
[TimeStamp]	BACnetTimeStamp Enthält entweder: 0: TIME 1: Unsigned 2: [DateTime]	String	{<2> {[10] 109.12.1.7},{[11] 17:00:00:000}}
[PriorityArray]	Array mit 16 Elementen vom Datentyp der PRESENT-VALUE Eigenschaft oder NULL	String	{[0]},{[4] 5.0},{[0]}, ...

[Recipient]	BACnetRecipient	String	
[DeviceObjectReference]	BACnetDeviceObjectReference	String	
[DeviceObjectPropertyReference]	BACnetDeviceObjectPropertyReference	String	
[ObjectPropertyReference]	BACnetObjectProperyReference	String	
[SetpointReference]	BACnetSetPointReference	String	
[ActionList]	BACnetActionList	String	
[EventParamater]	BACnetEventParameter	String	
[DateRange]	Start-Datum (DATE) gefolgt von End-Datum (DATE)	String	{109.12.1.255},{109.12.2.255}
[DailySchedule]	BACnetDailySchedule Liste von: TIME gefolgt von [Any]	String	
[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 Enthält entweder: 0: DATE	String	(2) 255.255.1

	1: [DateRange] 2: [WeekNDay]		
[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)	Bitstring mit Länge 4	String	0010
BITSTRING (EventTransitionBits)	Bitstring mit Länge 3	String	111
BITSTRING (ServicesSupported)	Bitstring mit Länge 40	String	
[Raw]	Daten sollen immer als Bytefolge kodiert werden	String	%10% 6D071603

BACNET OBJEKTE UND DEREN EIGENSCHAFTEN MIT DEN ENSTPRECHENDEN DATENTYPEN

Objekttyp (Typ)	Eigenschaft (ID)	Datentyp	Konformität
ANALOG-INPUT (0)			
	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	Lesen
	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	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	out-of-service (81)	BOOLEAN	Lesen
	present-value (85)	REAL	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional
	resolution (106)	REAL	Optional

	status-flags (111)	BITSTRING (StatusFlags), Length 4	Lesen
	time-delay (113)	UNSIGNED	Optional
	units (117)	ENUMERATED	Lesen
	update-interval (118)	UNSIGNED	Optional
ANALOG-OUTPUT (1)			
	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	Lesen
	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	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	out-of-service (81)	BOOLEAN	Lesen
	present-value (85)	REAL	Schreiben

	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	Lesen
	time-delay (113)	UNSIGNED	Optional
	units (117)	ENUMERATED	Lesen
	update-interval (118)	UNSIGNED	Optional
ANALOG-VALUE (2)			
	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	Lesen
	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	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	out-of-service (81)	BOOLEAN	Lesen

	present-value (85)	REAL	Lesen
	priority-array (87)	[PriorityArray]	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional
	relinquish-default (104)	REAL	Lesen
	resolution (106)	REAL	Optional
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Lesen
	time-delay (113)	UNSIGNED	Optional
	units (117)	ENUMERATED	Lesen
AVERAGING (18)			
	attempted-samples (124)	UNSIGNED	Schreiben
	average-value (125)	REAL	Lesen
	description (28)	CHARACTERSTRING	Optional
	maximum-value (135)	REAL	Lesen
	maximum-value-timestamp (149)	[TimeStamp]	Optional
	minimum-value (136)	REAL	Lesen
	minimum-value-timestamp (150)	[TimeStamp]	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-property-reference (78)	[ObjectPropertyReference]	Lesen
	object-type (79)	ENUMERATED	Lesen
	present-value (85)	REAL	Schreiben
	profile-name (168)	CHARACTERSTRING	Optional
	valid-samples (146)	UNSIGNED	Lesen
	variance-value (151)	REAL	Optional

	window-interval (147)	UNSIGNED	Schreiben
	window-samples (148)	UNSIGNED	Schreiben
BINARY-INPUT (3)			
	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	Lesen
	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	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	out-of-service (81)	BOOLEAN	Lesen
	polarity (84)	ENUMERATED	Lesen
	present-value (85)	ENUMERATED	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional

	status-flags (111)	BITSTRING (StatusFlags), Length 4	Lesen
	time-delay (113)	UNSIGNED	Optional
	time-of-active-time-reset (114)	[TimeStamp]	Optional
	time-of-state-count-reset (115)	[TimeStamp]	Optional
BINARY-OUTPUT (4)			
	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	Lesen
	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	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen

	out-of-service (81)	BOOLEAN	Lesen
	polarity (84)	ENUMERATED	Lesen
	present-value (85)	ENUMERATED	Schreiben
	priority-array (87)	[PriorityArray]	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional
	relinquish-default (104)	ENUMERATED	Lesen
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Lesen
	time-delay (113)	UNSIGNED	Optional
	time-of-active-time-reset (114)	[TimeStamp]	Optional
	time-of-state-count-reset (115)	[TimeStamp]	Optional
BINARY-VALUE (5)			
	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	Lesen
	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	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	out-of-service (81)	BOOLEAN	Lesen
	present-value (85)	ENUMERATED	Lesen
	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	Lesen
	time-delay (113)	UNSIGNED	Optional
	time-of-active-time-reset (114)	[TimeStamp]	Optional
	time-of-state-count-reset (115)	[TimeStamp]	Optional
CALENDAR (6)			
	datelist (23)	[CalendarEntry], List	Lesen
	description (28)	CHARACTERSTRING	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	present-value (85)	BOOLEAN	Lesen
	profile-name (168)	CHARACTERSTRING	Optional

COMMAND (7)			
	action (2)	ACTIONLIST, Array	Lesen
	action-text (3)	CHARACTERSTRING, Array	Lesen
	all-writes-successful (9)	BOOLEAN	Lesen
	description (28)	CHARACTERSTRING	Optional
	in-process (47)	BOOLEAN	Lesen
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	present-value (85)	UNSIGNED	Schreiben
	profile-name (168)	CHARACTERSTRING	Optional
DEVICE (8)			
	active-cov-subscriptions (152)	[COVSubscription], List	Optional
	active-vt-sessions (5)	[VTSession], List	Optional
	apdu-segment-timeout (10)	UNSIGNED	Optional
	apdu-timeout (11)	UNSIGNED	Lesen
	application-software-version (12)	CHARACTERSTRING	Lesen
	backup-failure-timeout (153)	UNSIGNED	Optional
	configuration-files (154)	OBJECTIDENTIFIER, Array	Optional
	database-revision (155)	UNSIGNED	Lesen
	daylight-savings-status (24)	BOOLEAN	Optional
	description (28)	CHARACTERSTRING	Optional
	device-address-binding (30)	[AddressBinding], List	Lesen

	firmware-revision (44)	CHARACTERSTRING	Lesen
	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	Lesen
	max-info-frames (63)	UNSIGNED	Optional
	max-master (64)	UNSIGNED	Optional
	max-segments-accepted (167)	UNSIGNED	Optional
	model-name (70)	CHARACTERSTRING	Lesen
	number-of-apdu-retries (73)	UNSIGNED	Lesen
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-list (76)	OBJECTIDENTIFIER, Array	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	protocol-revision (139)	UNSIGNED	Lesen
	protocol-services-supported (97)	BITSTRING (ServicesSupported), Length 40	Lesen
	protocol-version (98)	UNSIGNED	Lesen
	segmentation-supported (107)	ENUMERATED	Lesen
	system-status (112)	ENUMERATED	Lesen
	time-synchronization-recipients (116)	[Recipient], List	Optional

	utc-offset (119)	SIGNED	Optional
	vendor-identifier (120)	UNSIGNED	Lesen
	vendor-name (121)	CHARACTERSTRING	Lesen
	vt-classes-supported (122)	ENUMERATED, List	Optional
EVENTENROLLMENT (9)			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Lesen
	description (28)	CHARACTERSTRING	Optional
	event-enable (35)	BITSTRING (EventTransitionBits), Length 3	Lesen
	event-parameters (83)	[EventParameters]	Lesen
	event-state (36)	ENUMERATED	Lesen
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	event-type (37)	ENUMERATED	Lesen
	issue-confirmed-notifications (51)	BOOLEAN	Optional
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Lesen
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-property-reference (78)	[ObjectPropertyReference]	Lesen
	object-type (79)	ENUMERATED	Lesen
	priority (86)	UNSIGNED	Optional
	process-identifier (89)	UNSIGNED	Optional
	profile-name (168)	CHARACTERSTRING	Optional
	recipient (101)	RECIPIENT	Optional

FILE (10)			
	archive (13)	BOOLEAN	Schreiben
	description (28)	CHARACTERSTRING	Optional
	file-access-method (41)	ENUMERATED	Lesen
	file-size (42)	UNSIGNED	Lesen
	file-type (43)	CHARACTERSTRING	Lesen
	modification-date (71)	DATETIME	Lesen
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	read-only (99)	BOOLEAN	Lesen
	record-count (141)	UNSIGNED	Optional
GROUP (11)			
	description (28)	CHARACTERSTRING	Optional
	list-of-group-members (53)	[ReadAccessSpecification], List	Lesen
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	present-value (85)	[ReadAccessResult], List	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
LIFESAFETYPOINT (21)			
	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	Lesen
	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	Schreiben
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	operation-expected (161)	ENUMERATED	Lesen
	out-of-service (81)	BOOLEAN	Lesen
	present-value (85)	ENUMERATED	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Lesen
	setting (162)	UNSIGNED	Optional
	silenced (163)	ENUMERATED	Lesen
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Lesen
	time-delay (113)	UNSIGNED	Optional
	tracking-value (164)	ENUMERATED	Optional

	units (117)	ENUMERATED	Optional
LIFESAFETYZONE (22)			
	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	Lesen
	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	Schreiben
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	operation-expected (161)	ENUMERATED	Lesen
	out-of-service (81)	BOOLEAN	Lesen
	present-value (85)	ENUMERATED	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Lesen

	silenced (163)	ENUMERATED	Lesen
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Lesen
	time-delay (113)	UNSIGNED	Optional
	tracking-value (164)	ENUMERATED	Optional
	zone-members (165)	[DeviceObjectReference]	Lesen
LOOP (12)			
	acked-transitions (0)	BITSTRING (EventTransitionBits), Length 3	Optional
	action (2)	ENUMERATED	Lesen
	bias (14)	REAL	Optional
	controlled-variable-reference (19)	[ObjectPropertyReference]	Lesen
	controlled-variable-units (20)	ENUMERATED	Lesen
	controlled-variable-value (21)	REAL	Lesen
	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	Lesen
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	integral-constant (49)	REAL	Optional
	integral-constant-units (50)	ENUMERATED	Optional

	manipulated-variable-reference (60)	[ObjectPropertyReference]	Lesen
	maximum-output (61)	REAL	Optional
	minimum-output (68)	REAL	Optional
	notification-class (17)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	out-of-service (81)	BOOLEAN	Lesen
	output-units (82)	ENUMERATED	Lesen
	present-value (85)	REAL	Lesen
	priority-for-writing (88)	UNSIGNED	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	proportional-constant (93)	REAL	Optional
	proportional-constant-units (94)	ENUMERATED	Optional
	reliability (103)	ENUMERATED	Lesen
	setpoint (108)	REAL	Lesen
	setpoint-reference (109)	[SetPointReference]	Lesen
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Lesen
	time-delay (113)	UNSIGNED	Optional
	update-interval (118)	UNSIGNED	Optional
MULTISTATE-INPUT (13)			
	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	Lesen
	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	Lesen
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	out-of-service (81)	BOOLEAN	Lesen
	present-value (85)	UNSIGNED	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional
	state-text (110)	CHARACTERSTRING, Array	Optional
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Lesen
	time-delay (113)	UNSIGNED	Optional
MULTISTATE-OUTPUT (14)			
	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	Lesen

	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	Lesen
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	out-of-service (81)	BOOLEAN	Lesen
	present-value (85)	UNSIGNED	Schreiben
	priority-array (87)	[PriorityArray]	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	reliability (103)	ENUMERATED	Optional
	relinquisdefhault	UNSIGNED	Lesen
	state-text (110)	CHARACTERSTRING, Array	Optional
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Lesen
	time-delay (113)	UNSIGNED	Optional
MULTISTATE-VALUE (19)			
	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	Lesen
	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	Lesen
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	out-of-service (81)	BOOLEAN	Lesen
	present-value (85)	UNSIGNED	Lesen
	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	Lesen
	time-delay (113)	UNSIGNED	Optional
NOTIFICATIONCLASS (15)			
	ack-required (1)	BITSTRING (EventTransitionBits), Length 3	Lesen
	description (28)	CHARACTERSTRING	Optional
	notification-class (17)	UNSIGNED	Lesen
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	priority (86)	UNSIGNED, List	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	recipient-list (102)	[Destination], List	Lesen
PROGRAM (16)			

	description (28)	CHARACTERSTRING	Optional
	description-of-halt (29)	CHARACTERSTRING	Optional
	instance-of (48)	CHARACTERSTRING	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	out-of-service (81)	BOOLEAN	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	program-change (90)	ENUMERATED	Schreiben
	program-location (91)	CHARACTERSTRING	Optional
	program-state (92)	ENUMERATED	Lesen
	reason-for-halt (100)	ENUMERATED	Optional
	reliability (103)	ENUMERATED	Optional
	status-flags (111)	BITSTRING (StatusFlags), Length 4	Lesen
SCHEDULE (17)			
	description (28)	CHARACTERSTRING	Optional
	effective-period (32)	[DateRange]	Optional
	exception-schedule (38)	[SpecialEvent], Array	Optional
	list-of-object-property-references (54)	[DeviceObjectPropertyReference], List	Lesen
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	present-value (85)	[Any]	Lesen
	priority-for-writing (88)	UNSIGNED	Lesen
	profile-name (168)	CHARACTERSTRING	Optional
	weekly-schedule (123)	[DailySchedule], Array	Optional

TRENDLOG (20)			
	acked-transitions (0)		Optional
	buffer-size (126)	UNSIGNED	Lesen
	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	Lesen
	event-time-stamps (130)	[TimeStamp], Array[3]	Optional
	log-buffer (131)	[LogRecord], List	Lesen
	log-device-object-property (132)	[DeviceObjectPropertyReference]	Optional
	log-enable (133)	BOOLEAN	Schreiben
	log-interval (134)	UNSIGNED	Optional
	notification-class (17)	UNSIGNED	Optional
	notification-threshold (137)	UNSIGNED	Optional
	notify-type (72)	ENUMERATED	Optional
	object-identifier (75)	OBJECTIDENTIFIER	Lesen
	object-name (77)	CHARACTERSTRING	Lesen
	object-type (79)	ENUMERATED	Lesen
	previous-notify-time (138)	[DateTime]	Optional
	profile-name (168)	CHARACTERSTRING	Optional
	record-count (141)	UNSIGNED	Schreiben

	records-since-notification (140)	UNSIGNED	Optional
	start-time (142)	[DateTime]	Optional
	stop-time (143)	[DateTime]	Optional
	stop-when-full (144)	BOOLEAN	Lesen
	total-record-count (145)	UNSIGNED	Lesen

7.3 Treiberobjekte und Datentypen

Treiberobjekte sind in der Steuerung verfügbare Bereiche wie z. B. Merker, Datenbausteine usw. Hier lesen Sie, welche Treiberobjekte vom Treiber zur Verfügung gestellt werden und welche IEC-Datentypen dem jeweiligen Treiberobjekt zugeordnet werden können.

7.3.1 Treiberobjekte

Folgende Objekttypen stehen in diesem Treiber zur Verfügung:

Treiberobjekttyp	Kanaltyp	Lesen / Schreiben	Unterstützte Datentypen	Beschreibung
Command status	11	L	UDINT	Status ob Command erfolgreich ausgeführt wurde: 0: Erfolgreich 1: Kommando wird ausgeführt 2: Fehler
Device event information	10	L	STRING	Zusammenfassung aller offenen Alarme eines Gerätes
Event	9	L/S	BOOL, STRING	Empfang von BACnet EventNotifications. Adressierung: **<Gerätename>.<Objektname>.Event.<Event state>.bool** bzw.: **<Gerätename>.<Objektname>.Event.<Event state>.string**
Property	8	L/S	BOOL, DINT, INT, LREAL, REAL, SINT, STRING, UDINT, UINT, USINT	Entspricht SPS-Merker
Treibervariable	35	L/S	BOOL, DINT, INT, REAL, SINT, STRING, UDINT, UINT, USINT	Variablen zur statistischen Auswertung der Kommunikation. Weitere Infos finden Sie bei den Treibervariablen (auf Seite 67).

EVENT

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

Empfang von BACnet EventNotifications (BACnet Alarmen und Events). Eine BACnet EventNotification wird von BACnet Objekten versendet. Voraussetzung: Verknüpfung mit einem Notification Class Objekt.

Über den Treiberobjekttyp **Event** können diese Ereignisse in zenon ausgewertet werden.

Adressierung:

<Gerätename>.<Objektname>.Event.<Event state>.bool
bzw.

<Gerätename>.<Objektname>.Event.<Event state>.string

Hinweise:

- Ob Punkt (.) oder **#** als Seperator verwendet werden, hängt ab von den Einstellungen in der Eigenschaft `Property Seperator` in der Registerkarte **Options** in der Treiberkonfiguration (auf Seite 15).
- Ausnahme: Der Gerätename ist immer durch einen Punkt (.) vom Objektnamen getrennt.
- Das letzte Element des Namens (im Beispiel **bool** bzw. **string)** kann vom Benutzer frei vergeben werden.

WERTE <EVENT STATE>

<Event state> Kann folgende Werte annehmen:

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

Über die Eigenschaft `BACnet event state` kann der Event state auch als Zahl eingestellt werden. In diesem Fall kann der **Event state** im Namen durch einen beliebigen String ersetzt werden.

Voraussetzung: Die Process ID in der Notification class (ID für den Empfänger der Events in der Empfängeradresse) muss auf `0` gesetzt werden.

ABLAUF

- Immer wenn eine Notification des Objekts mit dem angegebenem `Event states` empfangen wird, wird die assozierte BOOL-Variable zuerst auf `false` gesetzt und dann auf `true`.
- Wird eine Notification des Objekts mit einem anderen `Event states` empfangen, wird die Variable `false` gesetzt. Dadurch ist auch eine Verarbeitung von Events ohne Änderung des `Event states` möglich.
- Ist `Event state` aktiv, wird die gesamte empfangene Event Notification der String-Variablen zugewiesen.
- Wird ein `Event state` ungleich des konfigurierten empfangen, wird die Variable auf einen Leer-String gesetzt.
- Die BOOL-Variablen werden beim Starten des Treibers initialisiert, die String Variablen auf Leerstring gesetzt.

Das Format der Stringvariablen ist analog dem Eigenschaftenformat.

FORMAT AUS BACNET SPEZIFIKATION

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

BEISPIEL

{{0} 0},{{1} 0008 0000010},{{2} 0002 0000000},{<3> {<2> {[10] 110.08.09 1},{[11] 12:03:52:018}}},{{4} 1},{{5} 128},{{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: Über Schreiben auf die Event-Variable wird der Event auf `Acknowledged` gesetzt (AcknowledgeAlarm Telegram). Dabei wird der zuletzt mit diesem `Event state` empfangene Event `Acknowledged`.

AE-INFO-A ALARM AND EVENT INFORMATION-A

Lesen einer Zusammenfassung aller offenen Alarme eines Geräts. Über eine String-Variable vom Objekttyp **`Event info`** kann diese Zusammenfassung gelesen werden.

FORMAT AUS BACNET SPEZIFIKATION

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

}

BEISPIEL

{(0) 0005 0000001},{(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 0000001},{(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}}},{(4) 0},{(5) 010},{<6> {[2] 128},{[2] 128},{[2] 128}},{(0) 0002 0000000},{(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 Zuordnung der Datentypen

Alle Variablen in zenon werden von IEC-Datentypen abgeleitet. In folgender Tabelle werden zur besseren Übersicht die IEC-Datentypen den Datentypen der Steuerung gegenübergestellt.

Steuerung	zenon	Datenart
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 und alle konstruierten bzw. kombinierten Typen	STRING	12
-	WSTRING	21
-	DATE	18
-	TIME	17
-	DATE_AND_TIME	20
-	TOD (Time of Day)	19

Datenart: Die Eigenschaft `Datenart` ist die interne numerische Bezeichnung des Datentyps. Diese wird auch für den erweiterten DBF Import/Export der Variablen verwendet.

7.4 Variablen anlegen durch Import

Variablen können auch mittels Variablenimport angelegt werden. Für jeden Treiber stehen XML- und DBF-Import zur Verfügung.

> **Info**
>
> *Details zu Import und Export von Variablen finden Sie im Handbuch Import-Export (main.chm::/13028.htm) im Abschnitt Variablen (main.chm::/13045.htm).*

7.4.1 XML Import

Für den Import/Export von Variablen gilt:

- Der Import/Export darf nicht aus dem Globalprojekt gestartet werden.
- Der Start erfolgt über:
  - Kontextmenü zu Variablen bzw. Datentyp im Projektbaum
  - oder Kontextmenü einer Variablen bzw. eines Datentyps
  - oder Symbol in der Symbolleiste Variablen

> **Achtung**
>
> *Beim Import/Überschreiben von existierenden Datentypen werden alle Variablen geändert, die auf diesem existierenden Datentyp basieren.*
>
> *Beispiel:*
>
> *Es existiert ein Datentyp XYZ abgeleitet vom Typ `INT` mit Variablen, die auf diesem Datentyp basieren. Ihre zu importierende XML-Datei enthält ebenfalls einen Datentyp mit Namen XYZ, allerdings abgeleitet vom Typ `STRING`. Wird dieser Datentyp importiert, so wird der existierende Datentyp überschrieben und bei allen auf ihm basierenden Variablen der Typ angepasst. D.h. die Variablen sind jetzt `STRING`- und keine `INT`-Variablen mehr.*

7.4.2 DBF Import/Export

Daten können nach dBase exportiert und aus dBase importiert werden.

> **Info**
>
> *Import und Export über CSV oder dBase unterstützt keine treiberspezifischen Variableneinstellungen wie z. B. Formeln. Nutzen Sie dafür den Export/Import über XML.*

IMPORT DBF-DATEI

Um den Import zu starten:

1. führen Sie einen Rechtsklick auf die Variablenliste aus
2. wählen Sie im Dropdownliste von **`Erweiterter Export/Import`** ... den Befehl **`dBase importieren`**
3. folgen Sie dem Importassistenten

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

> **Info**
>
> *Beachten Sie:*
>
> - Treiberobjekttyp und Datentyp müssen in der DBF-Datei an den Zieltreiber angepasst werden, damit Variablen importiert werden.
> - dBase unterstützt beim Import keine Strukturen oder Arrays (komplexe Variablen).

EXPORT DBF-DATEI

Um den Export zu starten:

1. führen Sie einen Rechtsklick auf die Variablenliste aus
2. wählen Sie im Dropdownliste von **`Erweiterter Export/Import`** ... den Befehl **`dBase exportieren`**
3. folgen Sie dem Exportassistenten

Achtung

DBF-Dateien:

- müssen in der Benennung dem 8.3 DOS Format für Dateinamen entsprechen (8 alphanumerische Zeichen für Name, 3 Zeichen Erweiterung, keine Leerzeichen)
- dürfen im Pfadnamen keinen Punkt (.) enthalten.
  Z. B. ist der Pfad `C:\users\Max.Mustermann\test.dbf` ungültig.
  Gültig wäre: `C:\users\MaxMustermann\test.dbf`
- müssen nahe am Stammverzeichnis (Root) abgelegt werden, um die eventuelle Beschränkungen für Dateinamenlänge inklusive Pfad zu erfüllen: maximal 255 Zeichen

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

Info

dBase unterstützt beim Export keine Strukturen oder Arrays (komplexe Variablen).

Dateiaufbau der dBase Exportdatei

Für den Variablenimport und -export muss die dBaseIV-Datei folgende Struktur und Inhalte besitzen.

Achtung

dBase unterstützt keine Strukturen oder Arrays (komplexe Variablen).

DBF-Dateien müssen:

- in der Benennung dem 8.3 DOS Format für Dateinamen entsprechen (8 alphanumerische Zeichen für Name, 3 Zeichen Erweiterung, keine Leerzeichen)
- nahe am Stammverzeichnis (Root) abgelegt werden

STRUKTUR

Bezeichnung	Typ	Feldgröße	Bemerkung
KANALNAME	Char	128	Variablenname. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
KANAL_R	C	128	Ursprünglicher Name einer Variablen, der durch den Eintrag unter VARIABLENNAME ersetzt werden soll (Feld/Spalte muss manuell angelegt werden). Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
KANAL_D	Log	1	Variable wird bei Eintrag 1 gelöscht (Feld/Spalte muss manuell angelegt werden).
TAGNR	C	128	Kennung. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
EINHEIT	C	11	Technische Maßeinheit
DATENART	C	3	Datentyp (z. B. Bit, Byte, Wort, ...) entspricht dem Datentyp.
KANALTYP	C	3	Speicherbereich in der SPS (z. B. Merkerbereich, Datenbereich, ...) entspricht Treiber-Objekttyp.
HWKANAL	Num	3	Bus-Adresse
BAUSTEIN	N	3	Datenbaustein-Adresse (nur bei Variablen aus den Datenbereich der SPS)
ADRESSE	N	5	Offset

BITADR	N	2	Für Bit-Variablen: Bitadresse Für Byte-Variablen: 0=niederwertig, 8=höherwertig Für String-Variablen: Stringlänge (max. 63 Zeichen)
ARRAYSIZE	N	16	Anzahl der Variablen im Array für Index-Variablen ACHTUNG: Nur die erste Variable steht voll zur Verfügung. Alle folgenden sind nur über VBA oder den Rezeptgruppen Manager zugänglich
LES_SCHR	L	1	Lese-Schreib-Berechtigung 0: Sollwert setzen ist nicht erlaubt 1: Sollwert setzen ist erlaubt
MIT_ZEIT	L	1	Zeitstempelung in zenon (nur wenn vom Treiber unterstützt)
OBJEKT	N	2	Treiberspezifische ID-Nummer des Primitivobjekts setzt sich zusammen aus TREIBER-OBJEKTTYP und DATENTYP
SIGMIN	Float	16	Rohwertsignal minimal (Signalauflösung)
SIGMAX	F	16	Rohwertsignal maximal (Signalauflösung)
ANZMIN	F	16	technischer Wert minimal (Messbereich)
ANZMAX	F	16	technischer Wert maximal (Messbereich)
ANZKOMMA	N	1	Anzahl der Nachkommastellen für die Darstellung der Werte (Messbereich)
UPDATERATE	F	19	Updaterate für Mathematikvariablen (in sec, eine Dezimalstelle möglich) bei allen anderen Variablen nicht verwendet
MEMTIEFE	N	7	Nur aus Kompatibilitätsgründen vorhanden
HDRATE	F	19	HD-Updaterate für hist. Werte (in sec, eine Dezimalstelle möglich)
HDTIEFE	N	7	HD-Eintragtiefe für hist. Werte (Anzahl)
NACHSORT	L	1	HD-Werte als nachsortierte Werte
DRRATE	F	19	Aktualisierung an die Ausgabe (für zenon DDE-Server, in sec, eine Kommastelle möglich)
HYST_PLUS	F	16	Positive Hysterese; ausgehend vom Messbereich
HYST_MINUS	F	16	Negative Hyterese; ausgehend vom Messbereich
PRIOR	N	16	Priorität der Variable

REAMATRIZE	C	32	Name der zugeordnete Reaktionsmatrix
ERSATZWERT	F	16	Ersatzwert; ausgehend vom Messbereich
SOLLMIN	F	16	Sollwertgrenze Minimum; ausgehend vom Messbereich
SOLLMAX	F	16	Sollwertgrenze Maximum; ausgehend vom Messbereich
VOMSTANDBY	L	1	Variable vom Standby-Server anfordern; der Wert der Variable wird im redundanten Netzwerkbetrieb nicht vom Server sondern vom Standby-Server angefordert
RESOURCE	C	128	Betriebsmittelkennung. Freier String für Export und Anzeige in Listen. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
ADJWVBA	L	1	Nichtlineare Wertanpassung: 0: Nichtlineare Wertanpassung wird verwendet 1: Nichtlineare Wertanpassung wird nicht verwendet
ADJZENON	C	128	Verknüpftes VBA-Makro zum Lesen der Variablenwerte für die nichtlineare Wertanpassung.
ADJWVBA	C	128	Verknüpftes VBA-Makro zum Schreiben der Variablenwerte für die nichtlineare Wertanpassung.
ZWREMA	N	16	Verknüpfte Zählwert-Rema.
MAXGRAD	N	16	Maximaler Gradient für die Zählwert-Rema.

Achtung

Beim Import müssen Treiberobjekttyp und Datentyp in der DBF-Datei an den Zieltreiber angepasst werden, damit Variablen importiert werden.

GRENZWERTDEFINITION

Grenzwertdefinition für Grenzwert 1 bis 4, bzw. Zustand 1 bis 4:

Bezeichnung	Typ	Feldgröße	Bemerkung
AKTIV1	L	1	Grenzwert aktiv (pro Grenzwert vorhanden)
GRENZWERT1	F	20	technischer Wert oder ID-Nummer der verknüpften Variable für einen dynamischen Grenzwert (siehe VARIABLEx) (wenn unter VARIABLEx 1 steht und hier −1, wird die bestehende Variablenzuordnung nicht überschrieben)
SCHWWERT1	F	16	Schwellwert für den Grenzwert
HYSTERESE1	F	14	wird nicht verwendet
BLINKEN1	L	1	Blinkattribut setzen
BTB1	L	1	Protokollierung in CEL
ALARM1	L	1	Alarm
DRUCKEN1	L	1	Druckerausgabe (bei CEL oder Alarm)
QUITTIER1	L	1	quittierpflichtig
LOESCHE1	L	1	löschpflichtig
VARIABLE1	L	1	dyn. Grenzwertverknüpfung der Grenzwert wird nicht durch einen absoluten Wert (siehe Feld GRENZWERTx) festgelegt.
FUNC1	L	1	Funktionsverknüpfung
ASK_FUNC1	L	1	Ausführung über die Alarmmeldeliste
FUNC_NR1	N	10	ID-Nummer der verknüpften Funktion (steht hier -1, so wird die bestehende Funktion beim Import nicht überschrieben)
A_GRUPPE1	N	10	Alarm/Ereignis-Gruppe
A_KLASSE1	N	10	Alarm/Ereignis-Klasse
MIN_MAX1	C	3	Minimum, Maximum
FARBE1	N	10	Farbe als Windowskodierung
GRENZTXT1	C	66	Grenzwerttext
A_DELAY1	N	10	Zeitverzögerung
INVISIBLE1	L	1	Unsichtbar

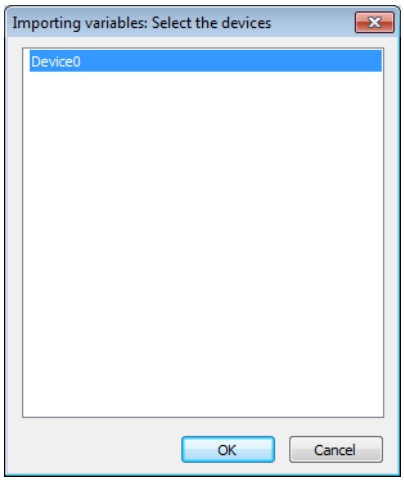
BEZEICHNUNGEN IN DER SPALTE BEMERKUNG BEZIEHEN SICH AUF DIE IN DEN DIALOGBOXEN ZUR DEFINITION VON VARIABLEN VERWENDETEN BEGRIFFE. BEI UNKLARHEITEN, SIEHE KAPITEL VARIABLENDEFINITION.

7.4.3 Online-Import

Über den Online-Import können Sie automatisch Variablen für die Eigenschaften aller Objekte eines bzw. mehrerer Geräte erstellen. Die Geräte müssen dafür in der Treiberkonfiguration (auf Seite 17) angelegt sein.

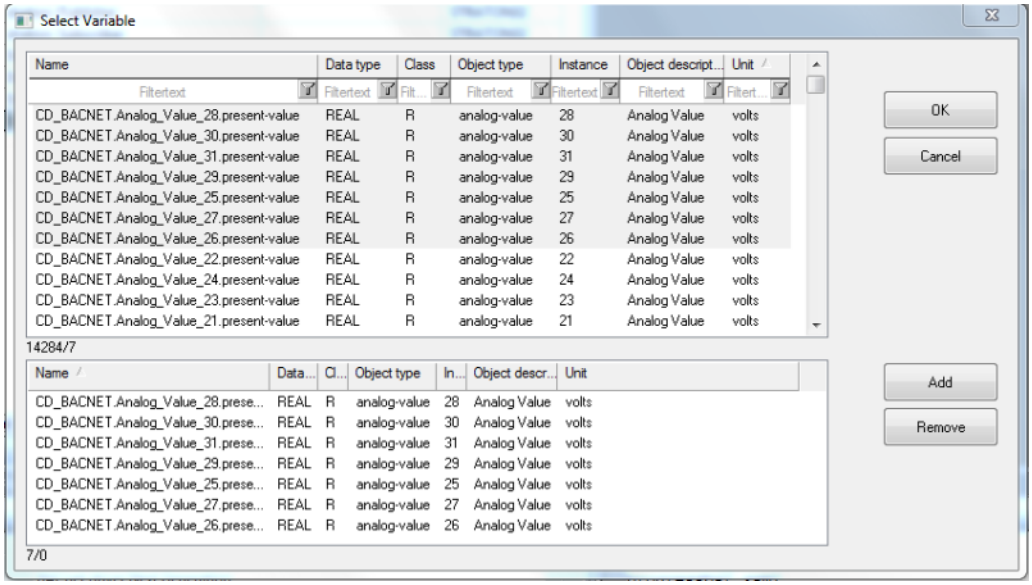
Um Daten online zu importieren:

1. führen Sie einen rechten Mausklick auf den Treiber aus
2. wählen Sie im Kontextmenü **`Variablen vom Treiber importieren...`**
3. das Fenster für die Auswahl des Geräts wird geöffnet



4. wählen Sie das gewünschte Gerät
5. klicken Sie auf OK

6. der Dialog für den Variablenimport wird geöffnet



7. wählen Sie die gewünschten Werte aus

es kann gefiltert werden nach:

- **Name**
- **Data type**
- **Class**
- **Object type**
- **Instance**
- **Object description**
- **Unit**

8. mit der Schaltfläche **Add** übernehmen Sie ausgewählte Elemente in die Importliste mit **Remove** entfernen Sie Elemente aus der Importliste

9. bestätigen Sie mit Klick auf OK die Auswahl

10. die Variablen werden importiert

UNITS

Während des Imports werden Units ausgelesen für:

- Analoge Value

- Analoge Input
- Analoge Output
- Loop Object

Die `Unit ID` wird in einen lesbaren String konvertiert. Dieser String wird als Variablen-Unit gespeichert. Units enthalten das Prefix @, um Sprachumschaltung zu ermöglichen. Units entsprechen dem BACnet Standard 135-2001.

Unterstützte Unit-Gruppen:

- Area
- Currency
- Electrical
- Energy
- Enthalpy
- Entropy
- Frequency
- Humidity
- Length
- Light
- Mass
- Mass Flow
- Power
- Pressure
- Temperature
- Time
- Velocity
- Volume
- Volumetric Flow

7.5 Treibervariablen

Das Treiberkit implementiert eine Reihe von Treibervariablen. Diese sind unterteilt in:

- Information
- Konfiguration
- Statistik und
- Fehlermeldungen

Die Definitionen der im Treiberkit implementierten Variablen sind in der Importdatei **`drvvar.dbf`** (auf der CD im Verzeichnis: `CD_Laufwerk:/Predefined/Variables`) verfügbar und können von dort importiert werden.

Hinweis: Variablennamen müssen in zenon einzigartig sein. Soll nach einem Import der Treibervariablen aus **`drvvar.dbf`** ein erneuter Import durchgeführt werden, müssen die zuvor importierten Variablen umbenannt werden.

Info

Nicht jeder Treiber unterstützt alle Treibervariablen.

Zum Beispiel werden:

- Variablen für Modem-Informationen nur von modemfähigen Treibern unterstützt
- Treibervariablen für den Polling-Zyklus nur für rein pollenden Treibern
- verbindungsbezogene Informationen wie ErrorMSG nur von Treibern, die zu einem Zeitpunkt nur eine Verbindung bearbeiten

INFORMATION

Name aus Import	Typ	Offset	Erklärung
MainVersion	UINT	0	Haupt-Versionsnummer des Treibers.
SubVersion	UINT	1	Sub-Versionsnummer des Treibers.
BuildVersion	UINT	29	Build-Versionsnummer des Treibers.
RTMajor	UINT	49	zenon Hauptversionsnummer
RTMinor	UINT	50	zenon Sub-Versionsnummer
RTSp	UINT	51	zenon Servicepack-Nummer
RTBuild	UINT	52	zenon Buildnummer
LineStateIdle	BOOL	24.0	TRUE, wenn die Modemleitung belegt ist.
LineStateOffering	BOOL	24.1	TRUE, wenn ein Anruf rein kommt.
LineStateAccepted	BOOL	24.2	Der Anruf wird angenommen.
LineStateDialtone	BOOL	24.3	Rufton wurde erkannt.
LineStateDialing	BOOL	24.4	Wahl aktiv.
LineStateRingBack	BOOL	24.5	Während Verbindungsaufbau.
LineStateBusy	BOOL	24.6	Zielstation besetzt.
LineStateSpecialInfo	BOOL	24.7	Spezielle Statusinformation empfangen.
LineStateConnected	BOOL	24.8	Verbindung hergestellt.
LineStateProceeding	BOOL	24.9	Wahl ausgeführt.
LineStateOnHold	BOOL	24.10	Verbindung in Halten.
LineStateConferenced	BOOL	24.11	Verbindung im Konferenzmodus.
LineStateOnHoldPendConf	BOOL	24.12	Verbindung in Halten für Konferenz.
LineStateOnHoldPendTransfer	BOOL	24.13	Verbindung in Halten für Transfer.
LineStateDisconnected	BOOL	24.14	Verbindung beendet.
LineStateUnknow	BOOL	24.15	Verbindungszustand nicht bekannt.
ModemStatus	UDINT	24	Aktueller Modemstatus.
TreiberStop	BOOL	28	Treiber gestoppt

			Bei `Treiberstop`, hat die Variable den Wert `TRUE` und ein **OFF**-Bit. Nach dem Treiberstart, hat die Variable den Wert `FALSE` und kein **OFF**-Bit.
SimulRTState	UDINT	60	Informiert über Status der Runtime bei Treibersimulation.

KONFIGURATION

Name aus Import	Typ	Offset	Erklärung
ReconnectInRead	BOOL	27	Wenn TRUE, dann wird beim Lesen automatisch ein Neuaufbau der Verbindung durchgeführt.
ApplyCom	BOOL	36	Änderungen an den Einstellungen der seriellen Schnittstelle zuweisen. Das Schreiben auf diese Variable hat unmittelbar den Aufruf der Methode SrvDrvVarApplyCom zur Folge (aktuell ohne weitere Funktion).
ApplyModem	BOOL	37	Änderungen an den Modemeinstellungen zuweisen. Das Schreiben auf diese Variable hat unmittelbar den Aufruf der Methode SrvDrvVarApplyModem zur Folge. Diese schließt die aktuelle Verbindung und öffnet eine neue entsprechend den Einstellungen **`PhoneNumberSet`** und **`ModemHwAdrSet`**.
PhoneNumberSet	STRING	38	Telefonnummer, welche verwendet werden soll.
ModemHwAdrSet	DINT	39	Hardwareadresse, welche zu der Telefonnummer gehört.
GlobalUpdate	UDINT	3	Updatezeit in Millisekunden (ms).
BGlobalUpdaten	BOOL	4	TRUE, wenn die Updatezeit global ist.
TreiberSimul	BOOL	5	TRUE, wenn der Treiber in Simulation ist.

TreiberProzab	BOOL	6	TRUE, wenn das Prozessabbild gehalten werden soll.
ModemActive	BOOL	7	TRUE, wenn das Modem bei diesem Treiber aktiv ist.
Device	STRING	8	Name der seriellen Schnittstelle oder Name des Modem.
ComPort	UINT	9	Nummer der seriellen Schnittstelle.
Baudrate	UDINT	10	Baudrate der seriellen Schnittstelle.
Parity	SINT	11	Parität der seriellen Schnittstelle.
ByteSize	USINT	14	Bitanzahl pro Zeichen der seriellen Schnittstelle. Wert = 0, wenn der Treiber keine serielle Kommunikation herstellen kann.
StopBit	USINT	13	Anzahl der Stoppbits der seriellen Schnittstelle.
Autoconnect	BOOL	16	TRUE, wenn die Modemverbindung automatisch beim Lesen/Schreiben aufgebaut werden soll.
PhoneNumber	STRING	17	Aktuelle Telefonnummer.
ModemHwAdr	DINT	21	Hardwareadresse zur aktuellen Telefonnummer.
RxIdleTime	UINT	18	Wenn länger als diese Zeit in Sekunden (s) erfolgreich kein Datenverkehr stattfindet, wird die Modemverbindung beendet.
WriteTimeout	UDINT	19	Maximale Schreibdauer bei einer Modemverbindung in Millisekunden (ms).
RingCountSet	UDINT	20	So oft läutet ein hereinkommender Anruf, bevor dieser angenommen wird.

ReCallIdleTime	UINT	53	Wartezeit zwischen Anrufen in Sekunden (s).
ConnectTimeout	UINT	54	Zeit in Sekunden (s) für Verbindungsaufbau.

STATISTIK

Name aus Import	Typ	Offset	Erklärung
MaxWriteTime	UDINT	31	Längste Zeit in Millisekunden (ms), die zum Schreiben benötigt wird.
MinWriteTime	UDINT	32	Kürzeste Zeit in Millisekunden (ms), die zum Schreiben benötigt wird.
MaxBlkReadTime	UDINT	40	Längste Zeit in Millisekunden (ms), die zum Lesen eines Datenblocks benötigt wird.
MinBlkReadTime	UDINT	41	Kürzeste Zeit in Millisekunden (ms), die zum Lesen eines Datenblocks benötigt wird.
WriteErrorCount	UDINT	33	Anzahl der Schreibfehler.
ReadSucceedCount	UDINT	35	Anzahl der erfolgreichen Leseversuche.
MaxCycleTime	UDINT	22	Längste Zeit in Millisekunden (ms), die zum Lesen aller angeforderten Daten benötigt wurde.
MinCycleTime	UDINT	23	Kürzeste Zeit in Millisekunden (ms), die zum Lesen aller angeforderten Daten benötigt wurde.
WriteCount	UDINT	26	Anzahl der Schreibversuche.
ReadErrorCount	UDINT	34	Anzahl der fehlerhaften Leseversuche.
MaxUpdateTimeNormal	UDINT	56	Zeit seit letzter Aktualisierung der Prioritätsgruppe `Normal` in Millisekunden (ms).
MaxUpdateTimeHigher	UDINT	57	Zeit seit letzter Aktualisierung der Prioritätsgruppe `Höher` in Millisekunden (ms).
MaxUpdateTimeHigh	UDINT	58	Zeit seit letzter Aktualisierung der Prioritätsgruppe `Hoch` in Millisekunden (ms).

MaxUpdateTimeHighest	UDINT	59	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Höchste** in Millisekunden (ms).
PokeFinish	BOOL	55	Geht für eine Abfrage auf 1, wenn alle anstehenden Pokes ausgeführt wurden.

FEHLERMELDUNGEN

Name aus Import	Typ	Offset	Erklärung
ErrorTimeDW	UDINT	2	Zeit (in Sekunden seit 1.1.1970), wann der letzte Fehler auftrat.
ErrorTimeS	STRING	2	Zeit (in Sekunden seit 1.1.1970), wann der letzte Fehler als String auftrat.
RdErrPrimObj	UDINT	42	Nummer des PrimObjektes, als der letzte Lesefehler verursacht wurde.
RdErrStationsName	STRING	43	Name der Station, als der letzte Lesefehler verursacht wurde.
RdErrBlockCount	UINT	44	Anzahl der zu lesenden Blöcke, als der letzte Lesefehler verursacht wurde.
RdErrHwAdresse	DINT	45	Hardwareadresse, als der letzte Lesefehler verursacht wurde.
RdErrDatablockNo	UDINT	46	Bausteinnummer, als der letzte Lesefehler verursacht wurde.
RdErrMarkerNo	UDINT	47	Merkernummer, als der letzte Lesefehler verursacht wurde.
RdErrSize	UDINT	48	Blockgröße, als der letzte Lesefehler verursacht wurde.
DrvError	USINT	25	Fehlermeldung als Nummer.
DrvErrorMsg	STRING	30	Fehlermeldung als Klartext.
ErrorFile	STRING	15	Name der Fehlerprotokolldatei.

8. Treiberspezifische Funktionen

Dieser Treiber unterstützt folgende Funktionen:

- Treiberkommandos (auf Seite 79)

- Zugriffsverfahren (spontanes und pollendes Lesen) (auf Seite 73)
- Abbildung der BACnet Status-Flags auf die Status-Bits der Variablen (auf Seite 73)
- Abbildung von BACnet Datentypen auf Stringvariablen (auf Seite 74)
- Device Management (auf Seite 78)

8.1 Zugriffsverfahren (spontanes und pollendes Lesen)

Abhängig von der eingestellten Priorität wird die Variable entweder über COV-Subscriptions spontan gelesen (Priorität Normal) oder gepollt (alle anderen Prioritäten. Für alle Prioritäten außer `Normal` entspricht das eingestellte Update-Intervall dem Pollingzyklus.

Achtung: Normalerweise können nur die `PRESENT-VALU`E und die `STATUS-FLAGS` Eigenschaft über COV-Subscription d.h. spontan gelesen werden. Da sich eine COV-Subscription immer nur auf ein Objekt bezieht und nicht auf eine Eigenschaft, führt das Anlegen einer Variable, die nicht spontan gelesen werden kann, zu keinem Log-Eintrag oder einem **`INVALID-Bit`**. In diesem Fall bleibt der Wert der Variablen einfach leer.

8.2 Abbildung der BACnet Status-Flags auf die Status-Bits der Variablen

Beim Lesen der `PRESENT-VALUE` Eigenschaft wird immer auch die `STATUS-FLAGS` Eigenschaft mitgelesen. Wird über eine Variable die `PRESENT-VALUE` Eigenschaft gelesen, wird die `STATUS-FLAGS` Eigenschaft wie folgt auf die Status-Bits der Variablen abgebildet:

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

8.3 Abbildung von BACnet Datentypen auf Stringvariablen

KODIERUNG DER WERTE

Datentyp	Tag-Nummer	Beschreibung
NULL	0	-
Boolean	1	FALSE (0): "FALSE" TRUE (1): "TRUE"
Unsigned	2	Dezimalzahl ohne Vorzeichen.
Integer	3	Dezimalzahl mit Vorzeichen.
Real	4	Gleitkommazahl.
Double	5	Gleitkommazahl.
Bit string	8	Bitsequenz (Folge von "1" bzw. "0").
Octet string	6	Bytesequenz (Folge von zweistelligen Hexadezimalen Zahlen).
Character string	7	Zeichenfolge (Enthält der String neben der Zeichenfolge auch noch andere Zeichen z. B. Tagnummer, wird die Zeichenfolge innerhalb von Anführungszeichen gestellt.
Enumerated	9	Dezimalzahl ohne Vorzeichen.
Date	10	`YYY.mm.dd.tt` `YYY`: Jahr seit 1900 `mm`: Monat `dd`: Tag `tt`: Wochentag) Wert `255` in einem der Felder bedeutet: `nicht angegeben`
Time	11	`hh:mm:ss:MMM`

		hh: Stunde mm: Minute ss: Sekunde MMM: 100/Sekunde Wert 255 in einem der Felder bedeutet: nicht angegeben
ObjectID	12	xxxx yyyyy xxxx: Objekttyp yyyyy: Objektinstanznummer
WeekNDay	kontext-spezifisch	mm.ww.tt mm: Monat ww: Woche des Monats tt: Wochentag Wert 255 in einem der Felder bedeutet: nicht angegeben

LISTEN BZW. ARRAYS VON EINFACHEN DATENTYPEN

Listen bzw. Arrays von Boolean, Unsigned, Integer, Real, Double, Bit string, Octet string, Character string, Enumerated, Date, Time oder ObjectID werden als Sequenz der kodierten Werte dargestellt. Dabei wird jeder Wert innerhalb von geschwungenen Klammern '{' und '}' gestellt und die einzelnen Werte werden durch Komma(,) getrennt.

KOMPLEXE DATENTYPEN

Bei komplexen Typen (d.h. bei kontextspezifischen Tags und allen Typen, die aus mehr als einem Tag bestehen aber keine Listen bzw. Arrays von einfachen Datentypen sind) wird neben dem Wert jedes Tags auch die Tag-Nummer in den String aufgenommen. In diesem Fall besteht ein Tag dann immer aus Tag-Nummer und Wert. Wurde in der Variablenkonfiguration nicht der BACnet Datentyp "[Raw]" ausgewählt und kann der Datentyp ermittelt werden (bei allen bekannten Eigenschaften möglich), wird

der Wert wie in der obigen Tabelle dargestellt im String kodiert, ansonsten erfolgt eine Kodierung des Werts als Bytesequenz (Folge von zweistelligen Hexadezimalen Zahlen).

Format der Tag-Nummer:

Tag-Klasse	Datentyp unbekannt bzw. eingestellter Typ "[Raw]"	Datentyp bekannt, Kodierung des Wertes anhand des Datentyps
Tag-Klasse: Applikation	Tagnummer innerhalb von '%' und '%'	Tagnummer innerhalb von eckigen Klammern '[' und ']'
Tag-Klasse: Kontextspezifisch	Tagnummer innerhalb von '#' und '#'	Tagnummer innerhalb von Klammern '(' und ')'

So genannte konstruierte Tags, das sind Tags welche eine Tagfolge enthalten, werden in geschwungene Klammern '{' und '}' gestellt und bestehen aus der Tag-Nummer gefolgt von den umschlossenen Tags, die durch Kommata (,) getrennt sind. Die Tag-Nummer wird zwischen '<' und '>' gestellt.

Achtung: Für die korrekte Dekodierung/Enkodierung muss der entsprechende BACnet Datentyp in den Adresseinstellungen der Variablen eingestellt sein.

FORMATIERUNGSBEISPIELE

NULL:

```
[0]
```

Boolean, BOOLEAN:

```
TRUE

FALSE

[1] TRUE

[1] FALSE
```

Datum, DATE:

```
107.11.25.7
```

```
[10] 107.11.25.7

%10% 6D071603
```

Uhrzeit, 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
```

Zeichenkette, CHARACTERSTRING:

```
BACnet

[7] "BACnet"
```

Datum, [CalendarEntry] mit [WeekNDay]:

```
(2) 255.255.1
```

[DateRange] bzw. Datums-Liste/Array:

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

Konstruierter Typ ([SpecialEvent])

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

8.4 Device Management

DM-DCC-A DEVICE COMMUNICATION CONTROL-A

BACnet Kommunikation Ein/Ausschalten

Syntax:

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

Beispiel:

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

Hinweis: Die Trennung von Schlüsselwort **DeviceCommunicationControl** und Parametern kann beliebig mittels der Zeichen **()** , und Leerzeichen erfolgen. Trennzeichen innerhalb von „" werden ignoriert.

DM-RD-A REINITIALIZE DEVICE-A

Gerät Reinitialisieren (Reset)

Syntax:

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

Beispiel:

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

Hinweis: Die Trennung von Schlüsselwort **DeviceCommunicationControl** und Parametern kann beliebig mittels der Zeichen `()`, und Leerzeichen erfolgen. Trennzeichen innerhalb von „" werden ignoriert.

DM-TS-A TIME SYNCHRONISATION-A

Zeitsynchronisationsbroadcast mit Lokalzeit

Syntax:

```
TimeSynchronisation
```

DM-UTC-A UTC TIME SYNCHRONISATION-A

Zeitsynchronisationsbroadcast mit UTC Zeit

Syntax:

```
UTCTimeSynchronisation
```

8.5 Import

Für die zenon Variable, die für `PRESENT-VALUE` erstellt wird, werden gesetzt, sofern vorhanden:

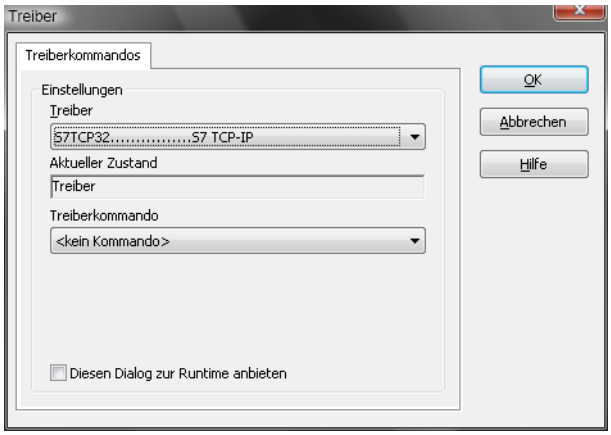
- Eigenschaft `Min/Max Wert`
- Nachkommastellen entsprechend den Eigenschaften `min-/max-pres-value` und `resolution` des BACnet Objekts

9. Treiberkommandos

Dieses Kapitel beschreibt Standardfunktionalitäten, die für die meisten zenon Treiber gültig sind. Nicht alle hier beschriebenen Funktionalitäten stehen für jeden Treiber zur Verfügung. Zum Beispiel enthält ein Treiber, der laut Datenblatt keine Modemverbindung unterstützt, auch keine Modem-Funktionalitäten.

Treiberkommandos dienen dazu, Treiber über zenon zu beeinflussen, z. B. starten und stoppen.
Die Projektierung erfolgt über die Funktion **Treiber Kommandos**. Dazu:

- legen Sie eine neue Funktion an
- wählen Sie Variablen -> Treiberkommandos
- der Dialog zur Konfiguration wird geöffnet



Parameter	Beschreibung
Treiber	Dropdownliste mit allen im Projekt geladenen Treibern.
Aktueller Zustand	Fixer Eintrag, in aktuellen Versionen ohne Funktion.
Treiberkommando	Dropdownliste zur Auswahl des Kommandos.
▸ Treiber starten (Online-Modus)	Treiber wird neu initialisiert und gestartet.
▸ Treiber stoppen (Offline-Modus)	Treiber wird angehalten, es werden keine neuen Daten angenommen. Hinweis: Ist der Treiber im Offline-Modus, erhalten alle Variablen, die für diesem Treiber angelegt wurden, den Status Abgeschaltet (OFF; Bit 20).
▸ Treiber in Simulationsmodus	Treiber wird in den Simulationsmodus gesetzt. Die Werte aller Variablen des Treibers werden vom Treiber simuliert. Es werden keine Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
▸ Treiber in Hardwaremodus	Treiber wird in den Hardwaremodus gesetzt. Für die Variablen des Treibers werden die Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
▸ Treiberspezifisches Kommando	Eingabe treiberspezifischer Kommandos. Öffnet Eingabefeld für die Eingabe eines Kommandos.

▸ Treiber Sollwertsetzen aktivieren	Sollwert setzen auf Treiber ist erlaubt.
▸ Treiber Sollwertsetzen deaktivieren	Sollwert setzen auf Treiber wird verhindert.
▸ Verbindung mit Modem aufbauen	Verbindung aufbauen (für Modem-Treiber). Öffnet Eingabefelder für Hardware-Adresse und Eingabe der zu wählenden Nummer.
▸ Verbindung mit Modem trennen	Verbindung beenden (für Modem-Treiber).
Diesen Dialog zur Runtime anbieten	Dialog wird zur Runtime für Änderungen angeboten.

TREIBERKOMMANDOS IM NETZWERK

Wenn sich der Rechner, auf dem die Funktion **Treiberkommandos** ausgeführt wird, im zenon Netzwerk befindet, werden zusätzliche Aktionen ausgeführt. Ein spezielles Netzwerkkommando wird vom Rechner zum Server des Projekts gesendet, der dann die gewünschte Aktion auf seinem Treiber durchführt. Zusätzlich sendet der Server das gleiche Treiberkommando zum Standby des Projekts. Der Standby führt die Aktion auch auf seinem Treiber aus.

Dadurch ist gewährleistet, dass Server und Standby synchronisiert sind. Dies funktioniert nur, wenn Server und Standby jeweils eine funktionierende und unabhängige Verbindung zur Hardware haben.

10. Fehleranalyse

Sollte es zu Kommunikationsproblemen kommen, bietet dieses Kapitel Hilfen, um den Fehler zu finden.

10.1 Analysetool

Alle zenon Module wie z. B. Editor, Runtime, Treiber, usw. schreiben Meldungen in eine gemeinsame Log-Datei. Um sie korrekt und übersichtlich anzuzeigen, benutzen Sie das Programm Diagnose Viewer (main.chm::/12464.htm), das mit zenon mitinstalliert wird. Sie finden es unter Start/Alle Programme/zenon/Tools 7.10 -> Diagviewer.

zenon Treiber protokollieren alle Fehler in Log-Dateien. Der Standardordner für die Log-Dateien ist der Ordner **LOG** unterhalb des Ordners `ProgramData`, zum Beispiel: `C:\ProgramData\zenon\zenon7.10\LOG` für die zenon Version 7.10. Log-Dateien sind Textdateien mit einer speziellen Struktur.

Achtung: Mit den Standardeinstellungen zeichnet ein Treiber nur Fehlerinformationen auf. Mit dem **Diagnose Viewer** kann bei den meisten Treibern die Diagnose-Ebene auf „Debug" und „Deep Debug" erweitert werden. Damit protokolliert der Treiber auch alle anderen wesentlichen Aufgaben und Ereignisse.

Im Diagnose Viewer kann man auch:

- eben erstellte Einträge live mitverfolgen
- die Aufzeichnungseinstellungen anpassen
- den Ordner, in dem die Log-Dateien gespeichert werden, ändern

Hinweise:

1. Unter Windows CE werden aus Ressourcegründen auch Fehler standardmäßig nicht protokolliert.
2. Der Diagnose Viewer zeigt alle Einträge in UTC (Koordinierter Weltzeit) an und nicht in der lokalen Zeit.
3. Der Diagnose Viewer zeigt in seiner Standardeinstellung nicht alle Spalten einer Log-Datei an. Um mehr Spalten anzuzeigen, aktivieren Sie die Eigenschaft **Add all columns with entry** im Kontextmenü der Spaltentitel.
4. Bei Verwendung von reinem **Error-Logging** befindet sich eine Problembeschreibung in der Spalte **Error text**. In anderen Diagnose-Ebenen befindet sich diese Beschreibung in der Spalte **General text**.
5. Viele Treiber zeichnen bei Kommunikationsprobleme auch Fehlernummern auf, die die SPS ihnen zuweist. Diese werden in **Error text** und/oder **Error code** und/oder **Driver error parameter(1** und **2)** angezeigt. Hinweise zur Bedeutung der Fehlercodes erhalten Sie in der Treiberdokumentation und der Protokoll/SPS-Beschreibung.
6. Stellen Sie am Ende Ihrer Tests den Diagnose-Level von **Debug** oder **Deep Debug** wieder zurück. Bei **Debug** und **Deep Debug** fallen beim Protokollieren

sehr viele Daten an, die auf der Festplatte gespeichert werden und die Leistung Ihres Systems beeinflussen können. Diese werden auch nach dem Schließen des `Diagnose Viewers` weiter aufgezeichnet.

Info

Weitere Informationen zum Diagnose Viewer finden Sie im Kapitel Diagnose Viewer (main.chm::/12464.htm).

10.2 BACnet Fehlercodes

Beschreibung der BACnet Fehlercodes, welche in Log-Meldungen dargestellt werden.

REJECT REASON

0	Unbestimmt (other)
1	Bufferüberlauf (buffer-overflow)
2	Inconsitente Parameter (inconsistent-parameters)
3	Ungültiger Parameterdatentype (invalid-parameter-data-type)
4	Ungültiger Tag (invalid-tag)
5	Fehlender Parameter (missing-required-parameter)
6	Parameter nicht innerhalb des erlaubten Wertbereichs (parameter-out-of-range)
7	Zu viele Argumente (too-many-arguments)
8	Ungültiger Wert für Enumeration (undefined-enumeration)
9	Unbekanntes Service (unrecognized-service)

ABORT REASON

0	Unbestimmt (other)
1	Bufferüberlauf (buffer-overflow)
2	Ungültige APDU zu diesem Zeitpunkt (invalid-apdu-in-this-state)
3	Preemted bei Task mit höherer Priorität (preempted-by-higher-priority-task)
4	Segmentierung wird nicht unterstützt (segmentation-not-supported)

ERROR

ERROR CLASS

0	Gerät
1	Objekt
2	Property

3	Resource
4	Security
5	Services
6	VT

ERROR CODE

0	Unbestimmt (other)
1	Authentifizierung fehlgeschlagen (authentication-failed)
2	Konfiguration läuft (configuration-in-progress)
3	Gerät ausgelastet (device-busy)
4	Dynamische Erstellung nicht möglich (dynamic-creation-not-supported)
5	Dateizugriff verweigert (file-access-denied)
6	Inkompatible Sicherheitslevels (incompatible-security-levels)
7	Inkonsitente Parameter (inconsistent-parameters)
8	Inkonsitentes Selektionskriterium (inconsistent-selection-criterion)
9	Ungültiger Datentyp (invalid-data-type)
10	Ungültige Dateizugriffsmethode (invalid-file-access-method)
11	Ungültige Dateistartposition (invalid-file-start-position)
12	Ungültiger Operatorname (invalid-operator-name)
13	Ungültiger Parameterdatentyp (invalid-parameter-data-type)
14	Ungültiger Zeitstempel (invalid-time-stamp)
15	Erstellung des Schlüssel fehlgeschlagen (key-generation-error)
16	Fehlender Parameter missing-required-parameter)
17	Keine Objekte des spezifizierten Typs (no-objects-of-specified-type)
18	Kein Platz für Objekt (no-space-for-object)
19	Kein Platz für das Hinzufügen eines Listenelemts (no-space-to-add-list-element)
20	Kein Platz für das Schreiben einer Eigenschaft (no-space-to-write-property)

21	Keine VT-Session verfügbar (no-vt-sessions-available)
22	Property ist keine List (property-is-not-a-list)
23	Löschen des Objekts nicht erlaubt (object-deletion-not-permitted)
24	Objekt ID existiert bereits (object-identifier-already-exists)
25	Operational-problem (operational-problem)
26	Passwort Fehler (password-failure)
27	Lesezugriff verweigert (read-access-denied)
28	Security nicht unterstützt (security-not-supported)
29	Service request verweigert (service-request-denied)
30	Timeout (timeout)
31	Unbekanntes Objekt (unknown-object)
32	Unbekanntes Property (unknown-property)
33	-
34	Unbekannte VT-Klasse (unknown-vt-class)
35	Unbekannte VT-Session (unknown-vt-session)
36	Nicht unterstützter Objekttyp (unsupported-object-type)
37	Wert nicht innerhalb der erlaubten Grenzen (value-out-of-range)
38	VT-Session bereits geschloßen (vt-session-already-closed)
39	Schließen der VT-Session fehlgeschlagen (vt-session-termination-failure)
40	Schreibzugriff verweigert (write-access-denied)
41	Zeichensatz nicht unterstützt (character-set-not-supported)
42	Ungültiger Array Index (invalid-array-index)
43	COV-Subscription fehlgeschlagen (cov-subscription-failed)
44	Kein COV Property (not-cov-property)
45	Optionale Funktionalität nicht unterstützt (optional-functionality-not-supported)
46	Ungültige Konfigurationsdaten (invalid-configuration-data)

10.3 Checkliste

Überprüfen Sie bei Kommunikationsfehler:

- Ist die Steuerung an die Stromversorgung angeschlossen?
- Sind die Teilnehmer im `IP`-Netz verfügbar?
- Kann die Steuerung über den `Ping` Befehl erreicht werden?
- Wurde der Gerätename sowohl im Treiberdialog als auch bei der Variablen korrekt eingestellt.?
- Entspricht das Property-Trennzeichen der Variable dem eingestellten Trennzeichen in der Treiberkonfiguration?
- Wird in der Variable der richtige Objekttyp verwendet?
- Bei Kommunikation über COV Subscriptions: Kann das gewählte Property über COV gelesen werden?
- Analyse mit Hilfe des Diagnose Viewers: Welche Meldungen werden angezeigt?

11. PICS - Protocol Implementation Conformance Statement

Date:	December 16, 2011
Vendor Name:	Ing. Punzenberger COPA-DATA GmbH
Product Name:	BACnet NG driver -Driver for process control system (HMI/SCADA)
Product Model Number:	n.a.
Applications Software Version:	6.51.800
Firmware Revision:	n.a.
BACnet protocol Revision:	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	Writeable 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	all
Life Safety Point	+	-	all	all
Life Safety Zone	+	-	all	all
Loop	+	-	all	all
Multi-state Input	+	-	all	all
Multi-state Output	+	-	all	all
Multi-state Value	+	-	all	all
Notification Class	+	-	all	all
Program	+	-	all	all

Schedule	+	-	all	all
Trend Log	+	-	all	all

Info

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

Info

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. Kommunikation mit der Steuerung

1. Ist die Adresse nicht manuell konfiguriert, wird über einen Who-Has Broadcast der den Namen der Steuerung enthält die Object ID der Steuerung und deren Adresse ermittelt.

Ist die Adresse manuell konfiguriert, wird die Objekt ID der Steuerung über einen Who-Has Unicast mit dem Namen der Steuerung an die angegebene Adresse ermittelt.

2. Es werden mit ReadPropertyMultiple die Kommunikationsparameter der Steuerung gelesen (Maximale APDU Größe, Unterstützung für Segmentierung, Maximale Anzahl der Segmente, APDU Timeout und APDU Segment Timeout). Ist die automatische Konfiguration der Timeouts aktiviert, werden die gelesenen Timeouts für die weitere Kommunikation übernommen.

3. Beim Browsen werden mittels ReadProperty die einzelnen Items der object-list Eigenschaften des Device-Objekts gelesen.

4. Für die Objekte aller angelegten Variablen wird mittels Unicast Who-Has Requests die ObjektID über den Objektnamen ermittelt.

Dabei werden nie mehr ausstehnde Who-Has Requests zugelassen als die maximale Anzahl der gleichzeitigen Requests.

5. Zyklisch werden die COV-Subscriptions erneuert bzw. werden die gepollten Eigenschaften über ReadProperty bzw. ReadPropertyMultiple (PRESEN-VALUE mit STATUS-FLAGS) gelesen.

6. Im Fehlerfall werden alle offenen Requests aborted und die eingestellt Fehlerwartezeit gewartet, danach wird wieder mit 1) begonnen.