zenon
by COPA-DATA

zenon Treiber Handbuch

BACnet32

v.8.20

© 2020 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.

Inhaltsverzeichnis

1 Willkommen bei der COPA-DATA Hilfe 5

2 BACnet32 5

3 BACnet32 - Datenblatt 6

4 Treiber-Historie 7

5 Voraussetzungen 8

5.1 PC 8

6 Konfiguration 8

6.1 Anlegen eines Treibers 9

6.2 Einstellungen im Treiberdialog 12

6.2.1 Allgemein 13

6.2.2 Treiberdialog BACnet-Einstellungen 17

6.2.3 Treiberdialog Diagnose-Einstellungen 18

6.2.4 Treiberdialog IP-Adressen 19

7 Variablen anlegen 20

7.1 Variablen im Editor anlegen 20

7.2 Adressierung 23

7.3 Treiberobjekte und Datentypen 24

7.3.1 Treiberobjekte 24

7.3.2 Zuordnung der Datentypen 28

7.4 Variablen anlegen durch Import 28

7.4.1 XML Import 28

7.4.2 DBF Import/Export 29

7.4.3 Online Import 36

7.5 Kommunikationsdetails (Treibervariablen) 36

8 Treiberspezifische Funktionen 42

9 Funktion Treiberkommandos 46

10 Fehleranalyse 52

10.1 Fehleranalyse 52

10.2 Analysetool 53

10.3 Treiberüberwachung 54
10.4 Checkliste 55

11 PICS (Protocol Implementation Conformance Statement) 56

1 Willkommen bei der COPA-DATA Hilfe

ZENON VIDEO-TUTORIALS

Praktische Beispiele für die Projektierung mit zenon finden Sie in unserem YouTube-Kanal (https://www.copadata.com/tutorial_menu). Die Tutorials sind nach Themen gruppiert und geben einen ersten Einblick in die Arbeit mit den unterschiedlichen zenon Modulen. Alle Tutorials stehen in englischer Sprache zur Verfügung.

ALLGEMEINE HILFE

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

PROJEKTUNTERSTÜTZUNG

Unterstützung bei Fragen zu konkreten eigenen Projekten erhalten Sie vom Customer Service, den Sie per E-Mail an support@copadata.com erreichen.

LIZENZEN UND MODULE

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

2 BACnet32

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

3 BACnet32 - Datenblatt

Allgemein:	
Treiberdateiname	BACnet32.exe
Treiberbezeichnung	Bacnet und DDC4000 Treiber
Steuerungs-Typen	Alle Bacnet "BACnet over IP, Annex J" konformen SPSen; Kieback + Peter DDC4000 Steuerungen
Steuerungs-Hersteller	Kieback + Peter; BACnet

Treiber unterstützt:	
Protokoll	BACnet/IP
Adressierung: Adress-basiert	Name based
Adressierung: Namens-basiert	--
Kommunikation spontan	X
Kommunikation pollend	X
Online Browsing	X
Offline Browsing	--
Echtzeitfähig	X
Blockwrite	--
Modemfähig	--
RDA numerisch	--
RDA String	--
Hysterese	--
erweiterte API	--
Unterstützung von Statusbit **WR-SUC**	--
alternative IP-Adresse	--

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

Plattformen:	
Betriebssysteme	Windows 10; Windows 7; Windows 8; Windows 8.1; Windows Server 2008 R2; Windows Server 2012; Windows Server 2012 R2; Windows Server 2016

4 Treiber-Historie

Datum	Treiberversion	Änderung
07.07.08	3600	Treiberdokumentation wurde neu erstellt
27.10.08	3800	Neue Funktionalität: einstellbare Priorität, einstellbare Leseverzögerung
17.11.08	4000	Fehlerbehebung der Dokuverweise

TREIBERVERSIONIERUNG

Mit zenon 7.10 wurde die Versionierung der Treiber verändert. Ab dieser Version gibt es eine versionsübergreifende Build-Nummer. Das ist die Zahl an der 4. Stelle der Dateiversion.
Zum Beispiel: **7.10.0.4228** bedeutet: Der Treiber ist für Version **7.10**, Service Pack **0** und hat die Build-Nummer **4228**.

Erweiterungen oder Fehlerbehebungen werden zukünftig in einem Build eingebaut und sind dann ab der nächsthöheren Build-Nummer verfügbar.

Beispiel

Eine Treibererweiterung wurde in Build **4228** implementiert. Der Treiber, den Sie im Einsatz haben, verfügt über die Build-Nummer **8322**. Da die Build-Nummer Ihres Treibers höher ist als die Build-Nummer der Erweiterung, ist die Erweiterung enthalten. Die Versionsnummer des Treiber (die ersten drei Stellen der Dateiversion) spielen dabei keine Rolle. Die Treiber sind versionsunabsabhängig

5 Voraussetzungen

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

5.1 PC

Treiberdateien: BACnet32.exe, PTP.dll, nb_link_settings.dll in das Programmverzeichnis kopieren, falls nicht bereits vorhanden.

WinPcap.dll, Installationssatz im Internet frei erhältlich unter z. B. http://www.winpcap.org/install/default.htm

Die Datei WinPcap.dll wird in das Verzeichnis system32 kopiert.

Empfohlen wird Version 3.0 oder höher

6 Konfiguration

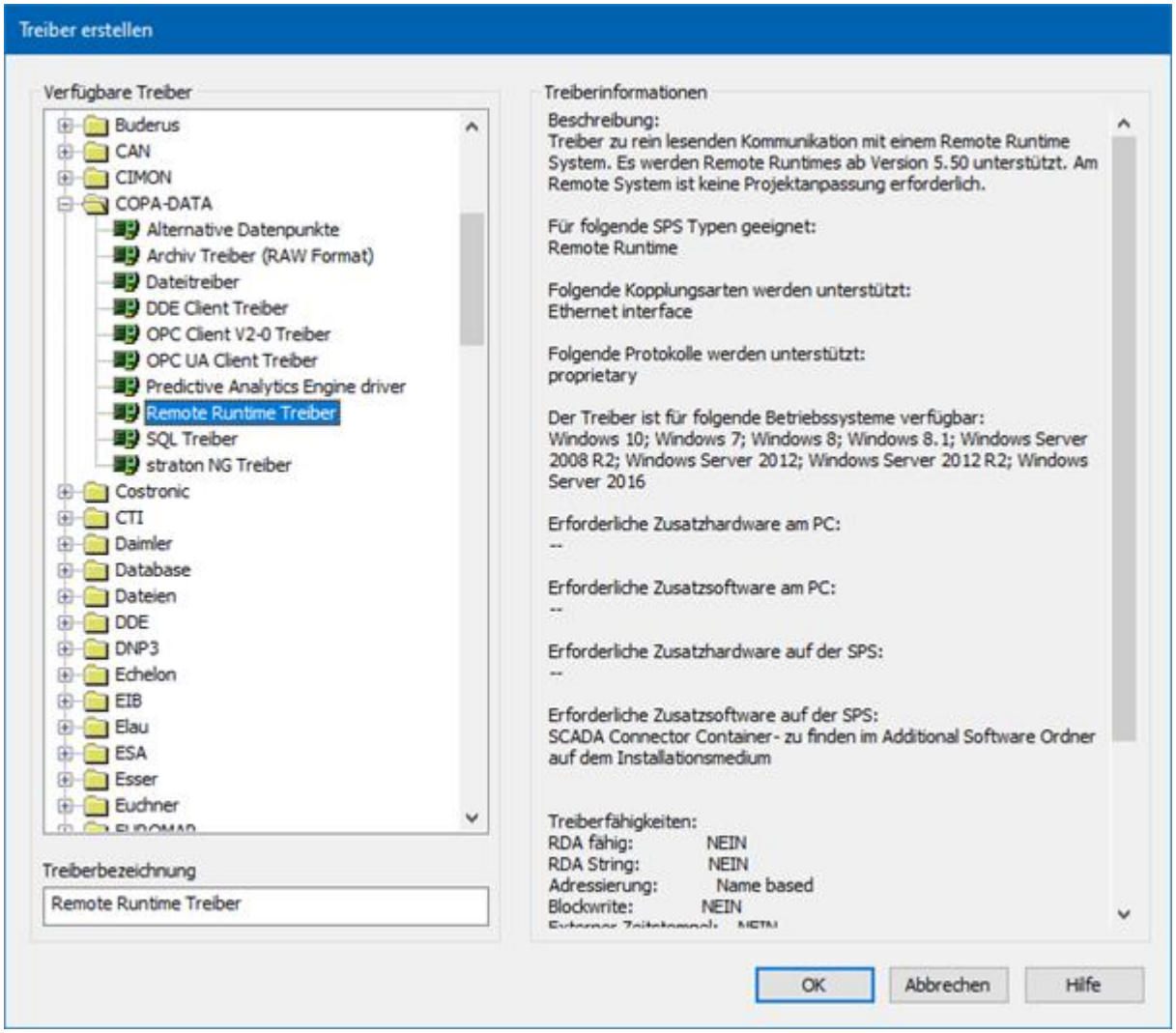
In diesem Kapitel lesen Sie, wie Sie den Treiber im Projekt anlegen und welche Einstellungen beim Treiber möglich sind.

Info

Weitere Einstellungen, die Sie für Variablen in zenon vornehmen können, finden Sie im Kapitel Variablen der Online-Hilfe.

6.1 Anlegen eines Treibers

Im Dialog **Treiber erstellen** wählen Sie aus einer Liste jenen Treiber, den Sie neu anlegen wollen.



Parameter	Beschreibung
Verfügbare Treiber	Liste aller verfügbaren Treiber. Die Darstellung erfolgt in einer Baumstruktur: *[+]* erweitert die Ordnerstruktur und zeigt die darin enthaltenen Treiber. [-] reduziert die Ordnerstruktur Default: *keine Auswahl*
Treiberbezeichnung	Eindeutige **Bezeichnung** des Treibers. Default: *leer* Das Eingabefeld wird nach Auswahl eines Treibers

Parameter	Beschreibung
	aus der Liste der verfügbaren Treiber mit der vordefinierten **Bezeichnung** vorausgefüllt.
Treiberinformationen	Weiterführende Informationen über den gewählten Treiber. Default: *leer* Nach Auswahl eines Treibers werden in diesem Bereich die Informationen zum gewählten Treiber angezeigt.

DIALOG BEENDEN

Option	Beschreibung
OK	Übernimmt alle Einstellungen und öffnet den Treiberkonfigurationsdialog des ausgewählten Treibers.
Abbrechen	Verwirft alle Änderungen und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

> **Info**
>
> Die Inhalte dieses Dialogs sind in der Datei Treiber_[Sprachkürzel].xml gespeichert. Sie finden diese Datei im Ordner *C:\ProgramData\COPA-DATA\zenon[Versionsnummer]*.

TREIBER NEU ANLEGEN

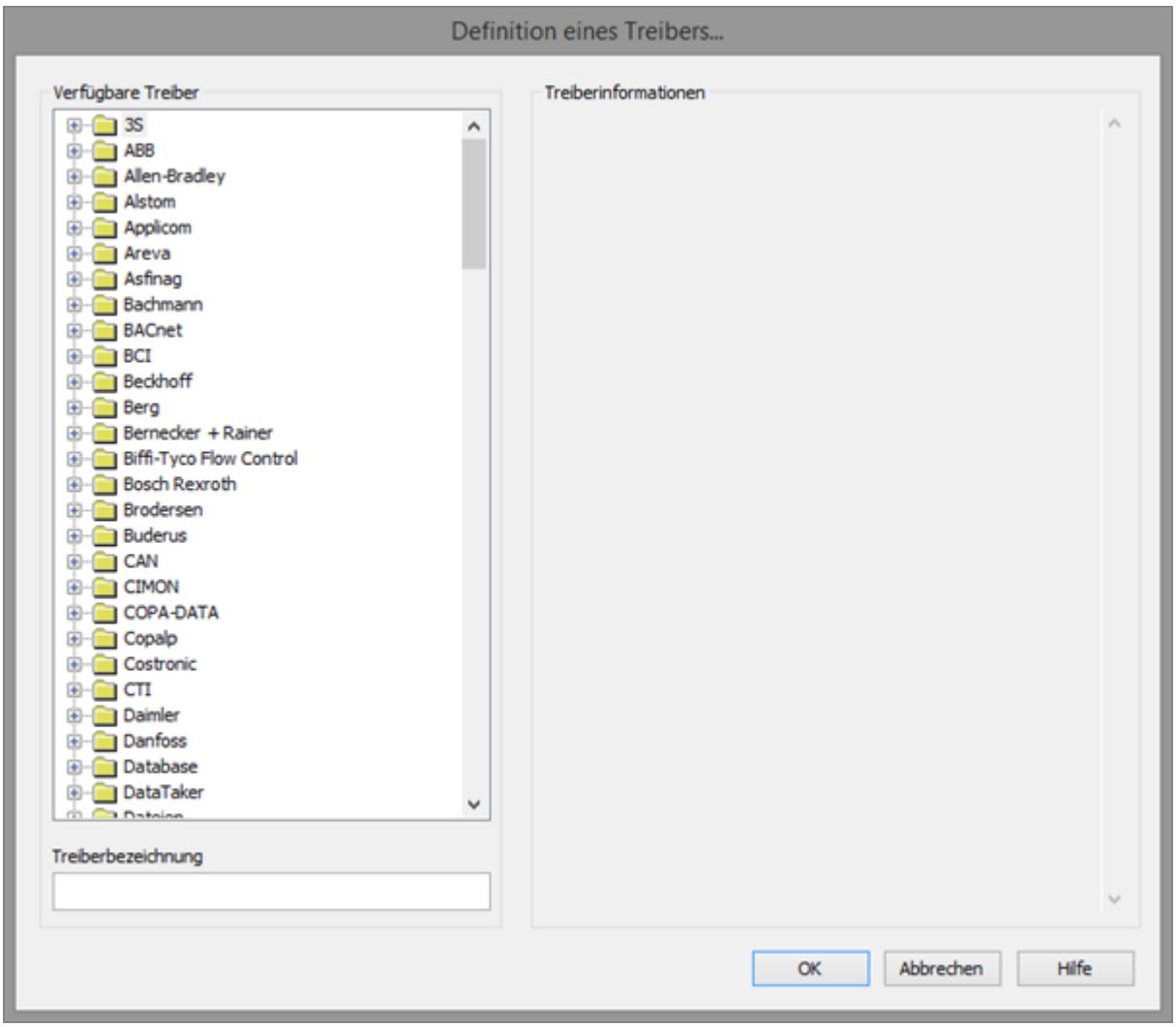
Um einen neuen Treiber anzulegen:

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

   Optional: Wählen Sie die Schaltfläche **Treiber neu** aus der Symbolleiste der Detailansicht der **Variablen**.

   Der Dialog **Treiber erstellen** wird geöffnet.

2. Der Dialog bietet eine Auflistung aller verfügbaren Treiber an.



3. Wählen Sie den gewünschten Treiber und benennen Sie diesen im Eingabefeld **Treiberbezeichnung**.
   Dieses Eingabefeld entspricht der Eigenschaft **Bezeichnung**. Per Default wird der Name des ausgewählten Treibers in diesem Eingabefeld automatisch eingefügt.
   Für die **Treiberbezeichnung** gilt:

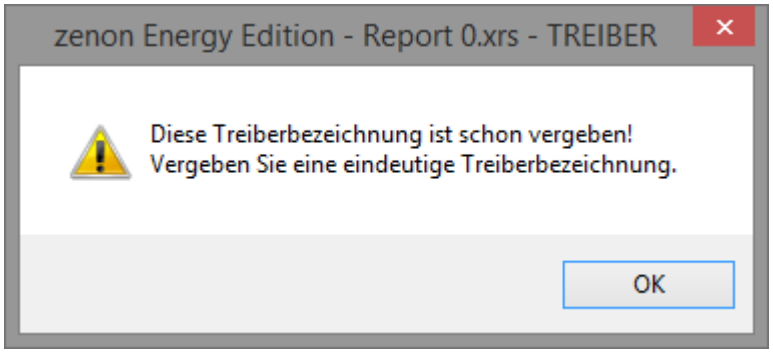
   - Die **Treiberbezeichnung** muss eindeutig sein.
     Wird ein Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
     Dies wird durch Klick auf die Schaltfläche **OK** evaluiert. Ist die Treiber im Projekt bereits vorhanden wird dies mit einem Warndialog angezeigt.
   - 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 Klick auf die Schaltfläche **OK**.
   Der Konfigurationsdialog des ausgewählten Treibers wird geöffnet.

Hinweis: Treibernamen sind nicht sprachumschaltbar. Sie werden später immer in der Sprache angezeigt, in der sie angelegt wurden, unabhängig von der Sprache des Editors. Das gilt auch für Treiberobjekttypen.

DIALOG TREIBERBEZEICHNUNG BEREITS VORHANDEN

Ist ein Treiber bereits im Projekt vorhanden wird dies in einem Dialog angezeigt. Mit Klick auf die Schaltfläche **OK** wird der Warndialog geschlossen. Der Treiber kann korrekt benannt werden.



ZENON PROJEKT

Bei neu angelegten Projekten werden die folgenden Treiber automatisch angelegt:

- **Intern**
- **MathDr32**
- **SysDrv**

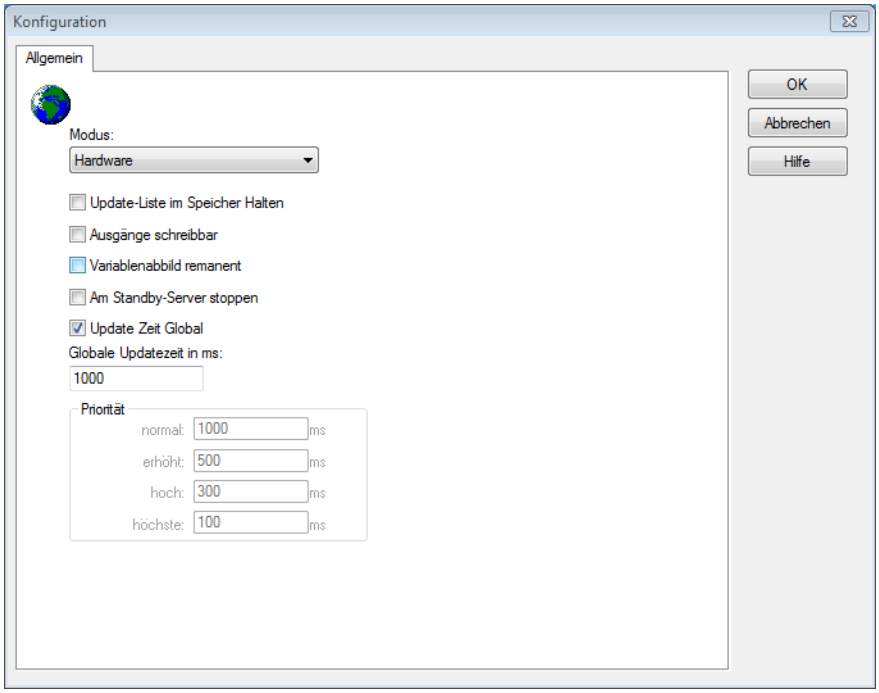
> **Info**
>
> In einem zenon Projekt müssen nur die benötigten Treiber vorhanden sein. Treiber können bei Bedarf zu einem späteren Zeitpunkt hinzugefügt werden.

6.2 Einstellungen im Treiberdialog

Folgende Einstellungen können Sie beim Treiber vornehmen:

6.2.1 Allgemein

Beim Anlegen eines Treibers wird der Konfigurationsdialog geöffnet. Um den Dialog später zum Bearbeiten zu öffnen, führen Sie einen Doppelklick auf den Treiber in der Liste aus oder klicken Sie auf die Eigenschaft **Konfiguration**.



Option	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 oder 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.

Option	Beschreibung
	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.
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 dieser Status aktiv ist: ▸ Benutzerstatus *M1* (*0*) bis *M8* (*7*) ▸ *REVISION(9)* ▸ *AUS(20)* ▸ *ERSATZWERT(27)* Das Variablenabbild wird immer gespeichert wenn: ▸ die Variable vom Objekttyp

Option	Beschreibung
	Kommunikationsdetails 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**, sondern einen leeren Wert. Damit wird verhindert, dass beim Hochstufen zum Server nicht relevante Werte in AML , CEL und Archiv erzeugt werden. Default: *inaktiv* **Hinweis:** Nicht verfügbar, wenn CE Terminal als Datenserver dient. Weitere Informationen dazu erhalten Sie im Handbuch zenon Operator im Kapitel CE Terminal als Datenserver.
Update Zeit Global	Einstellung für globale Update-Zeiten in Millisekunden: ▸ *Aktiv*: Die eingestellte **Globale Update Zeit** wird für alle Variablen im Projekt verwendet. Die bei den Variablen eingestellte Priorität wird nicht

Option	Beschreibung
	verwendet. ▸ *Inaktiv*: Die eingestellten Prioritäten werden für die einzelnen Variablen verwendet. **Ausnahmen:** Spontane Treiber ignorieren diese Option. Sie nutzen in der Regel die kürzest mögliche Update Zeit. Details siehe Abschnitt **Update Zeit spontane Treiber**.
Priorität	Hier werden die Pollingzeiten der einzelnen Prioritätsklassen eingestellt. Alle Variablen mit der entsprechenden Priorität werden in der eingestellten Zeit gepollt. Die Zuordnung der Variablen erfolgt separat bei jeder Variablen über die Einstellungen in den Variableneigenschaften. Mit den Prioritätsklassen kann die Kommunikation der einzelnen Variablen auf die Wichtigkeit oder benötigte Aktualität abgestuft werden. Daraus ergibt sich eine verbesserte Verteilung der Kommunikationslast. **Achtung:** Prioritätsklassen werden nicht von jedem Treiber unterstützt, z.B. von spontan kommunizierenden zenon Treibern.

DIALOG BEENDEN

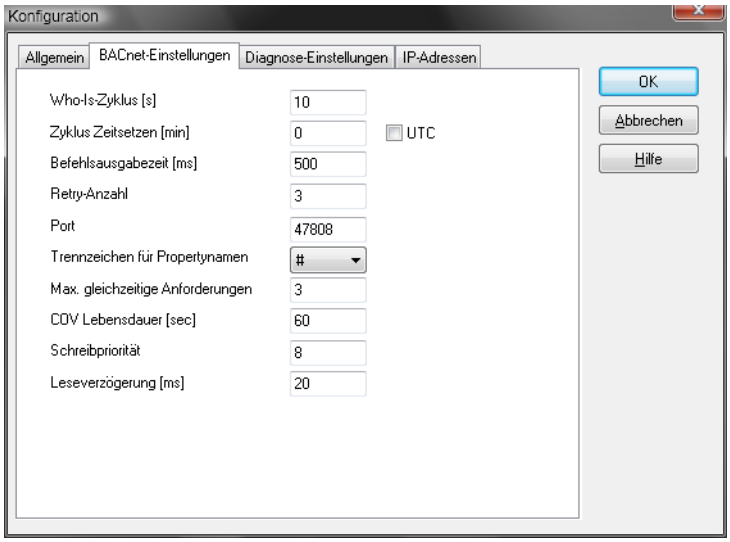
Option	Beschreibung
OK	Übernimmt alle Änderungen in allen Registerkarten und schließt den Dialog.
Abbrechen	Verwirft alle Änderungen in allen Registerkarten und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

UPDATE ZEIT SPONTANE TREIBER

Bei spontanen Treibern wird beim **Sollwert Setzen**, **Advisen** von Variablen und bei **Requests** sofort ein Lesezyklus ausgelöst - unabhängig von der eingestellten Update Zeit. Damit wird sicher gestellt, dass der Wert nach dem Schreiben in der Visualisierung sofort zur Verfügung steht. In der Regel beträgt die Updatezeit 100 ms.

Spontane Treiber sind **ArchDrv**, **BiffiDCM**, **BrTcp32**, **DNP3**, **Esser32**, **FipDrv32**, **FpcDrv32**, **IEC850**, **IEC870**, **IEC870_103**, **Otis**, **RTK9000**, **S7DCOS**, **SAIA_Slave**, **STRATON32** und **Trend32**.

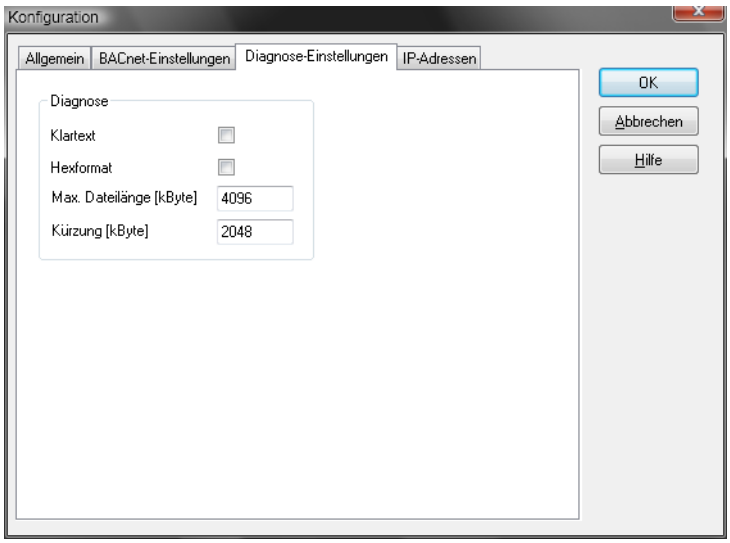
6.2.2 Treiberdialog BACnet-Einstellungen



Parameter	Beschreibung
Who-Is-Zyklus [s]	Zeitintervall der Aussendung des Who-Is-Services
Zyklus Zeitsetzen[min]	Zykluszeit für Zeitsetzen [min], Default: *0* (kein Zeitsetzen)
Befehlsausgabezeit [ms]	Befehlsausgabezeit für Sendestrategie 1 (Einzelbefehl) [ms]
Retry-Anzahl	Max. Anzahl der Aussendung des „Who-Has"-Services pro Objekt
Port	Einstellung des UDP-Ports, Angabe dezimal. (Default für BACNet 47808)
Trennzeichen für Property Namen	
Anzahl gleichzeitige Anfragen	Maximale Anzahl gleichzeitiger COV-Anmeldepakete. Wird für BACNet Geräte verwendet, die nur eine bestimmte Anzahl von Paketen buffern und abarbeiten können.
COV Lebensdauer[sec]	Nach dieser Zeit werden angelegte COV Subscriptions ungültig und danach vom Treiber neu angefordert (siehe BACnet Standard)
Schreibe-Priorität	Legt die Schreibepriorität für Variablen fest. (siehe BACnet Standard). Je niedriger der Wert desto höher die Priorität ACHTUNG: Werte unter "8" werden häufig von der Steuerung selbst verwendet und sollten daher nur mit

Parameter	Beschreibung
	Vorsicht verwendet werden.
Lese-Verzögerung[ms]	Konfiguriebare Verzögerung zwischen 2 Pollinganfragen. Dies wird für Geräte mit Performanceproblemen benötigt.

6.2.3 Treiberdialog Diagnose-Einstellungen

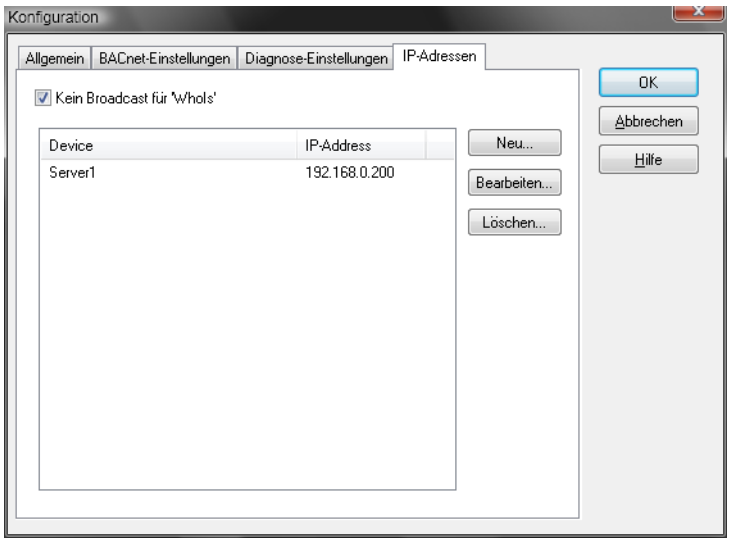


Parameter	Beschreibung
Klartext	Protokollierung des BACnet-Telegramms im Klartext
Hexformat	Protokollierung des BACnet-Telegramms im Hexformat
Max. Dateilänge [kByte]	Max. Dateilänge des Log-Files
Kürzung [kByte]	Kürzung des Log-Files, wenn max. Dateilänge erreicht

Die Ausgabe erfolgt in die Datei <Runtimepfad>\RT\FILES\zenon\custom\drivers\BACnet.log

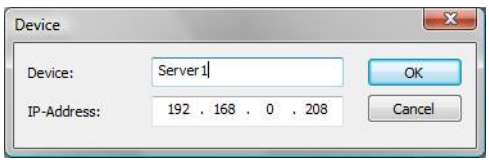
Bzw. für Editorkommunikation (browsing) in die Datei:
<SQLpfad>\<ProjektGUID>\FILES\zenon\custom\drivers\BACnet.log.

6.2.4 Treiberdialog IP-Adressen



Parameter	Beschreibung
Kein Broadcast für 'Whols'	Ist diese Option aktiviert, werden ‚Whols' Nachrichten nur an die BACNet Stationen in der Liste darunter gesendet. ACHTUNG: Diese Funktion ist im BACNet Standard nicht vorgesehen. Wir geben daher keine Garantie dafür dass dies mit allen BACNet Steuerungen funktioniert
Neu...	Anlegen einer neuen Station: Device: Angabe des Device-Namen (z. B.Gerätename im BACNet) IP-Adresse: IP Adresse des Zielgerätes
Bearbeiten...	Zum Bearbeiten der markierten Adresse
Löschen...	Zum Löschen der markierten Adresse

DIALOG ZUM ANLEGEN BZW. BEARBEITEN VON VERBINDUNGEN



Device	Frei wählbarer Name der Verbindung
IP-Adresse	IP Adresse des BACnet Gerätes

7 Variablen anlegen

So werden Variablen im zenon Editor angelegt:

7.1 Variablen im Editor anlegen

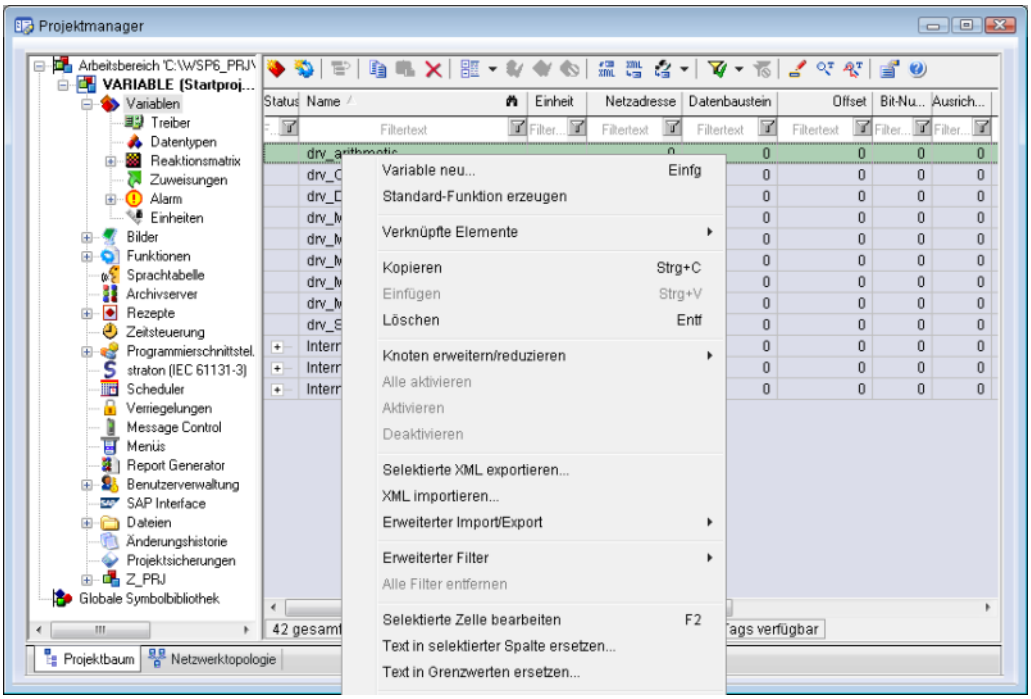
Variablen können angelegt werden:

- als einfache Variable
- in Arrays
- als Struktur-Variablen

DIALOG VARIABLE

Um eine neue Variable zu erstellen, gleich welchen Typs:

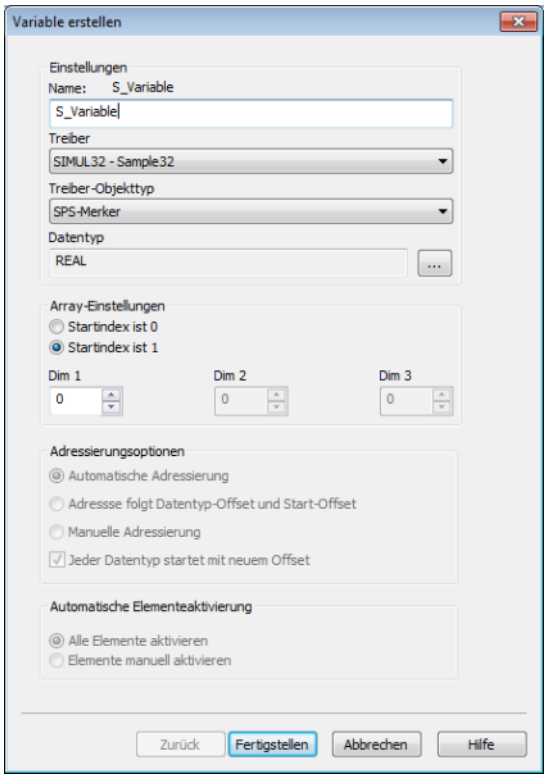
1. Wählen Sie im Knoten **Variablen** im Kontextmenü den Befehl **Variable neu**.



   Der Dialog zur Konfiguration der Variable wird geöffnet.

2. Konfigurieren Sie die Variable.
3. Welche Einstellungen möglich sind, hängt ab vom Typ der Variablen.

DIALOG VARIABLE ERSTELLEN



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:** Die Zeichen **#** und **@** sind für Variablennamen nicht erlaubt. Bei Verwendung nicht zugelassener Zeichen kann die Variablenerstellung nicht abgeschlossen werden, die Schaltfläche **Fertigstellen** bleibt inaktiv. **Hinweis:** Manche Treiber erlauben die Adressierung auch über die Eigenschaft **Symbolische Adresse**.
Treiber	Wählen Sie aus der Dropdownliste den gewünschten Treiber. **Hinweis:** Sollte im Projekt noch kein Treiber angelegt sein, wird automatisch der Treiber für interne Variable (**Intern.exe**) geladen.
Treiberobjekttyp	Wählen Sie aus der Dropdownliste den passenden Treiberobjekttyp aus.
Datentyp	Wählen Sie den gewünschten Datentyp. Klick auf die Schaltfläche

Eigenschaft	Beschreibung
	... 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.

SYMBOLISCHE ADRESSE

Die Eigenschaft **Symbolische Adresse** kann für die Adressierung alternativ zu **Name** oder **Kennung** der Variablen verwendet werden. Die Auswahl erfolgt im Treiberdialog, die Konfiguration in der Variableneigenschaft. Beim Import von Variablen unterstützter Treiber wird die Eigenschaft automatisch eingetragen.

Maximale Länge: *1024* Zeichen.

Folgende Treiber unterstützen die **Symbolische Adresse**:

- **3S_V3**
- **AzureDrv**
- **BACnetNG**
- **IEC850**
- **KabaDPServer**
- **OPCUA32**
- **Phoenix32**
- **POZYTON**
- **RemoteRT**
- **S7TIA**
- **SEL**
- **SnmpNg32**
- **PA_Drv**
- **EUROMAP63**

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

Die Adressierung der Variablen des BACnet-Treibers erfolgt über den eindeutigen Namen der Variablen. Die Datenpunkte des BACnet-Treibers sind durch den Objekt-Namen eindeutig referenziert. Dem Objekt-Namen des BACnet-Objekts ist der Objekt-Name des Deviceobjekts vorangestellt, getrennt durch einen Punkt. (z. B. BPS10.d10a002), Bei Schedule-Objekten wird der Name der Property durch einen Punkt getrennt angehängt. Die virtuellen Datenpunkte vom Typ BACnet-Komponentenstatus spiegeln den Zustand der Verbindung wider. Sie sind eindeutig durch den Objektnamen des Deviceobjekte gekennzeichnet.

EIGENSCHAFTEN ZUR EINDEUTIGEN ADRESSIERUNG VON VARIABLEN

Eigenschaft	Beschreibung
Name	Wird für die eindeutige Adressierung verwendet. Format: **device-name.object-name[.property]**
Kennung	Frei vergebbare Kennung. Z. B. für **Betriebsmittelkennung**, Kommentar usw. Kann für die eindeutige Adressierung verwendet werden. Format: **device-name.object-name[.property]**
Netzadresse	Wird für diesen Treiber nicht verwendet.
Datenbaustein	Wird für diesen Treiber nicht verwendet.
Offset	Wird für diesen Treiber nicht verwendet.
Ausrichtung	Wird für diesen Treiber nicht verwendet.
Bitnummer	Wird für diesen Treiber nicht verwendet.
Stringlänge	Nur verfügbar bei String-Variablen. Maximale Anzahl von Zeichen, die die Variable aufnehmen kann.
Treiber Anbindung/Treiberobjekttyp	Objekttyp der Variablen. Wird abhängig vom verwendeten Treiber beim Erstellen der Variablen ausgewählt und kann hier geändert werden.

Eigenschaft	Beschreibung
Treiber Anbindung/Datentyp	Datentyp der Variablen. Wird beim Erstellen der Variablen ausgewählt und kann hier geändert werden. **ACHTUNG:** Wenn der Datentyp nachträglich geändert wird, müssen alle anderen Eigenschaften der Variablen überprüft bzw. angepasst werden.
Treiber Anbindung/Priorität	Ordnet einer Variablen eine Priorität für die Updatezeit zu.

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	Kommentar
Ausgang	11	X	X	*BOOL, REAL, SINT, USINT*	
Eingang	10	X	X	*BOOL, REAL, SINT, USINT*	
SPS-Merker	8	X	X	*BOOL, REAL, SINT, USINT*	
Kommunikationsdetails	35	X	X	*BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING*	Variablen für die statische Analyse der Kommunikation. Werte werden nur zwischen Treiber und Runtime übertragen, nicht zur SPS! **Hinweis**: Die Adressierung und das Verhalten ist bei den

Treiberobjekttyp	Kanaltyp	Lesen	Schreiben	Unterstützte Datentypen	Kommentar
					meisten zenon Treibern gleich. Weitere Informationen dazu finden Sie im Kapitel Kommunikationsdetails (Treibervariablen) (auf Seite 36).

Legende:

X: wird unterstützt

--: wird nicht unterstützt

TREIBER OBJEKTTYPEN UND UNTERSTÜTZTE IEC DATENTYPEN FÜR PROZESSVARIABLEN IN ZENON

Treiber Objekttypen	Untersützte Datentypen	Lesen	Schreiben	BACnet Object Type
Eingang	*BOOL*	X	--	Binary Input
	REAL	X	--	Analog Input
	SINT, USINT	X	--	Multistate Input
Ausgang	*BOOL*	--	X	Binary Output
	REAL	--	X	Analog Output
	SINT, USINT	--	X	Multistate Output
SPS-Merker	*BOOL*	X	X	Binary Value
	REAL	X	X	Analog Value Schedule - Present Value
	USINT	X	X	Komponenten Status
	*STRING **	X	X	Schedule - Effective Period

Treiber Objekttypen	Untersützte Datentypen	Lesen	Schreiben	BACnet Object Type
				Schedule - Weekly Schedule Schedule - Exception Schedule

* Die einzelnen Properties von Schedule-Objekten besitzen einen komplexen und variablen Aufbau (Siehe BACnet Standard). Daher können sie nur als Strings an zenon übergeben werden. Diese Strings haben je nach Property verschiedene Formate (Leerzeichen können beim Sollwertsetzen weggelassen werden):

EFFECTIVE PERIOD

Parameter	Beschreibung
Effective Period:	{ <Start-Datum> - <End-Datum> }
Datum:	<Jahr – 1900>.<Monat (1..12)>.<Tag (1..31)>.<Wochentag (1..7, 1 = Montag)>

WEEKLY SCHEDULE

Parameter	Beschreibung	
Weekly Schedule:	{ Array [1..7] of <Daily Schedule> }	
Daily Schedule:	{ List of <Time Value> }	
Time Value:	{ <Zeit>, [<Typ>] <Wert> }	
Zeit:	<Stunde>:<Minute>:<Sekunde>:<1/100 Sek.>	
Typ:	1	BOOLEAN
	2	UNSIGNED
	3	SIGNED
	4	REAL
	5	DOUBLE
	6	OCTET_STRING
	7	CHARACTER_STRING

Parameter	Beschreibung	
	8	BIT_STRING
	9	ENUMERATED
	10	DATE
	11	TIME
	12	OBJECT_IDENTIFIER

EXCEPTION SCHEDULE

Parameter	Beschreibung
Exception Schedule:	{ Array [1..n] of <Special Event> }
Special Event:	{ <Period> { List of <Time Value>* } <Event Priority> }
Period:	{ [0] <Calendar Entry> } oder { [1] <Calendar Reference> }
Calendar Reference:	Instanz-Nr. des referenzierten Calendar-Objekts
Calendar Entry:	[0] <Datum>** oder [1] <Start-Datum>** - <End-Datum>** oder [2] <WeekNDay>
WeekNDay:	<Monat (1..12)>.<Woche im Monat (1..6)>.<Wochentag (1..7)>
Event Priority:	1..16, 1 = höchste, 16 = niedrigste Priorität

* siehe Weekly Schedule, ** siehe Effective Period

KANALTYP

Der Begriff "**Kanaltyp**" ist die interne numerische Bezeichnung des Treiberobjekttyps. Diese wird auch für den erweiterten DBF Import/Export der Variablen verwendet.

"**Kanaltyp**" wird für den erweiterten CSV Import/Export der Variablen in der Spalte "**HWObjectType**" verwendet.

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.

BEISPIELE FÜR ALLE MÖGLICHEN IEC DATENTYPEN

SPS	zenon	Wertebereich
8 Bit signed	SINT	-128 bis 127
8 Bit unsignd	USINT	0 bis 255
32 Bit Fliesskomma	REAL	± 3.4E ± 38
Boolean	BOOL	0, 1

DATENART

Der Begriff **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 im Abschnitt Variablen.

7.4.1 XML Import

Beim XML- Import von Variablen oder Datentypen werden diese erst einem Treiber zugeordnet und dann analysiert. Vor dem Import entscheidet der Benutzer, ob und wie das jeweilige Element (Variable oder Datentyp) importiert werden soll:

- *Importieren*:
  Das Element wird neu importiert.
- *Überschreiben*:
  Das Element wird importiert und überschreibt ein bereits vorhandenes Element.

- *Nicht importieren*:
  Das Element wird nicht importiert.

Hinweis: Beim Import werden die Aktionen und deren Dauer in einem Fortschrittsbalken angezeigt. In der folgenden Dokumentation wird der Import von Variablen beschrieben. Datentypen werden analog dazu importiert.

VORAUSSETZUNGEN

Beim Import gelten folgende Bedingungen:

- **Abwärtskompatibilität**

  Beim XML Import/Export ist keine Abwärtskompatibilität gegeben. Daten aus älteren zenon Versionen können übernommen werden. Die Übergabe von Daten aus neueren Versionen an ältere wird nicht unterstützt.

- **Konsistenz**

  Die zu importierende XML-Datei muss konsistent sein. Beim Import der Datei erfolgt keine Plausibilitätsprüfung. Weisen die importierten Daten Fehler auf, kann es zu unerwünschten Effekten im Projekt kommen.

  Dies muss vor allem auch beachtet werden, wenn in einer XML-Datei nicht alle Eigenschaften vorhanden sind und diese dann durch Default-Werte ersetzt werden. Z. B.: Eine binäre Variable hat einen Grenzwert von *300*.

- **Struktur-Datentypen**

  Struktur-Datentypen müssen über die gleiche Anzahl von Strukturelementen verfügen. Beispiel: Ein Strukturdatentyp im Projekt hat 3 Strukturelemente. Ein gleichnamiger Datentyp in der XML-Datei hat 4 Strukturelemente. Dann wird keine der auf diesem Datentyp basierenden Variablen der Datei in das Projekt importiert.

Tipp

Weitere Informationen zum XML-Import finden Sie im Handbuch **Import - Export**, im Kapitel **XML-Import**.

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 in der Dropdownliste von **Erweiterter Export/Import** ... den Befehl **dBase importieren**.
3. Folgen Sie den Anweisungen des 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 den Anweisungen des 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.

Bezeichnung	Typ	Feldgröße	Bemerkung
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 Treiberobjekttyp.
HWKANAL	Num	3	Netzadresse
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)

Bezeichnung	Typ	Feldgröße	Bemerkung
OBJEKT	N	2	Treiberspezifische ID-Nummer des Primitivobjekts setzt sich zusammen aus TREIBER-OBJEKTTYP und DATENTYP
SIGMIN	Floa t	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

Bezeichnung	Typ	Feldgröße	Bemerkung
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*, oder Zustand *1* bis *4*:

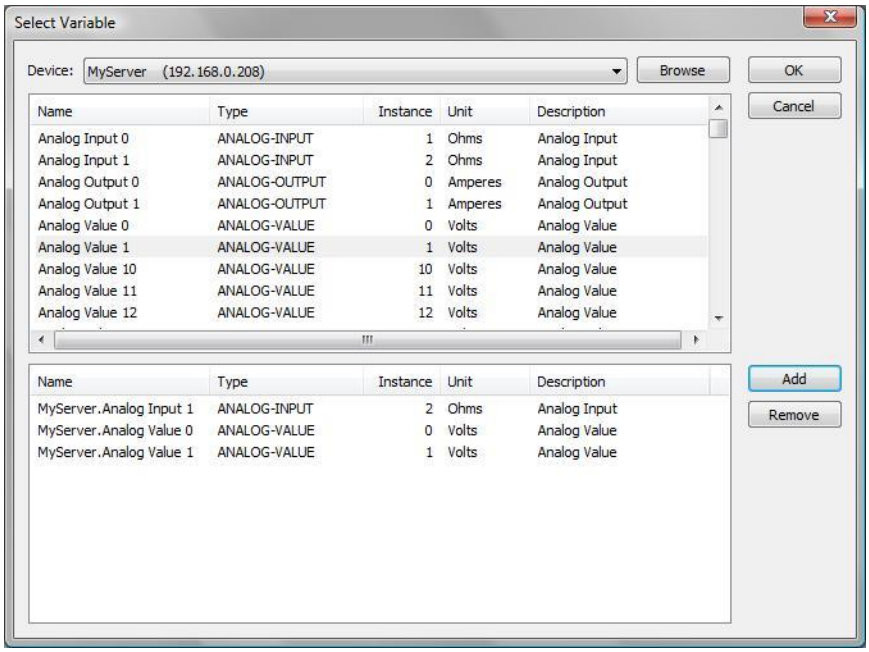
Bezeichnung	Typ	Feldgröße	Bemerkung
AKTIV1	L	1	Grenzwert aktiv (pro Grenzwert vorhanden)
GRENZWERT1	F	20	technischer Wert oder ID-Nummer der verknüpften Variable für einen dynamischen Grenzwert (siehe VARIABLEx) (wenn unter VARIABLEx *1* steht und hier *-1*, wird die bestehende Variablenzuordnung nicht überschrieben)
SCHWWERT1	F	16	Schwellwert für den Grenzwert
HYSTERESE1	F	14	wird nicht verwendet
BLINKEN1	L	1	Blinkattribut setzen

Bezeichnung	Typ	Feldgröße	Bemerkung
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

Variablen werden durch den Treiber Online Import angelegt. Sie finden den Aufruf in der Treiberliste im Kontextmenü des Treibers.



Parameter	Beschreibung
Device	Auswahl des zu browsenden Devices
Button "Browse"	Liest die Variablen des Devices aus
Button "Add"	Fügt die ausgewählte Variable zur Auswahl hinzu
Button "Remove"	Entfernt die ausgewählte Variable aus der Auswahl
Button "OK"	Übernimmt die ausgewählten Variablen in die Variablenliste
Button "Cancel"	Abbruch des Imports

7.5 Kommunikationsdetails (Treibervariablen)

Das Treiberkit implementiert eine Reihe von Treibervariablen, welche in dem Treiberobjekttyp *Kommunikationsdetails* zusammengefasst sind. Diese sind unterteilt in:

- Information
- Konfiguration
- Statistik und
- Fehlermeldungen

Die Definitionen der im Treiberkit implementierten Variablen sind in der Importdatei **DRVVAR.DBF** verfügbar und können von dort importiert werden.
Pfad zur Datei: *%ProgramData%\COPA-DATA\zenon<Versionsnummer>\PredefinedVariables*

Hinweis: Variablennamen müssen in zenon einzigartig sein. Soll nach einem Import der Variablen vom Treiberobjekttyp *Kommunikationsdetails* aus **DRVVAR.DBF** ein erneuter Import durchgeführt werden, müssen die zuvor importierten Variablen umbenannt werden.

Info

Nicht jeder Treiber unterstützt alle Treibervariablen des Treiberobjekttyps *Kommunikationsdetails*.

Zum Beispiel:

- Variablen für Modem-Informationen werden nur von modemfähigen Treibern unterstützt.
- Treibervariablen für den Polling-Zyklus stehen nur für rein pollende Treiber zur Verfügung.
- Verbindungsbezogene Informationen wie **ErrorMSG** werden nur von Treibern unterstützt, 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 Service Pack-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.

Name aus Import	Typ	Offset	Erklärung
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.
ConnectionStates	*STRING*	61	Interner Verbindungsstatus des Treibers zur SPS. Verbindungszustände: ▸ *0:* Verbindung OK ▸ *1:* Verbindung gestört ▸ *2:* Verbindung simuliert Formatierung: **<Netzadresse>:<Verbindungszustand>;...;...**

Name aus Import	Typ	Offset	Erklärung
			; Eine Verbindung ist erst nach dem ersten Anmelden einer Variablen bekannt. Damit eine Verbindung im String enthalten ist, muss einmal eine Variable dieser Verbindung angemeldet worden sein. Der Zustand einer Verbindung wird nur aktualisiert, wenn eine Variable der Verbindung angemeldet ist. Ansonsten wird nicht mit der entsprechenden Steuerung kommuniziert.

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.

Name aus Import	Typ	Offset	Erklärung
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

Name aus Import	Typ	Offset	Erklärung
			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:

PROTOKOLL

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.

VERBINDUNG

Der BACnet-Standard bietet fünf Optionen auf den Data-Link- und Physikalischen Schichten des (auf 4 Schichten reduzierten) OSI-Schichten-Models. Der Treiber unterstützt nur die Option ISO 8802-3, bekannt als „Ethernet". Diese Option (zusammen mit den übrigen Schichten des BACnet-Protokolls) wird auch als „BACnet/IP, Annex J" bezeichnet.

BACnet Layers				
BACnet Application Layer				
BACnet Network Layer				
ISO 8802-2 Type1		MS/TP	PTP	LonTa
ISO 8802-3	ARCNET	EIA-485 (RS485)	EIA-232 (RS232)	

Die Kommunikation dieser Option setzt in der Data-Link-Schicht auf ISO 8802-2 Type1 (Ethernet) auf. Auf der PC-Seite ist also eine Ethernetkarte mit Verbindung zu einem TCP/IP-Netzwerk notwendig.

BEGRIFFSERKLÄRUNG

Begriff	Beschreibung
BAS	BACnet-Automatisierungsstation
BAZ	Befehlsausgabezeit
COV	Change of Value
DA	Datenart
DCS	Double Command State bei TK 46
KT	Kanaltyp
OB	Objekt
RPM	ReadPropertyMultiple

ALLGEMEINE FUNKTIONSBESCHREIBUNG

Die BACnet-Automatisierungsstation (BAS) wird im BACnet-Treiber als Device-Objekt abgebildet. Die BAS verhält sich als Server. Zum Peer-to-Peer Datenaustausch meldet sich der BACnet-Treiber als Client an. Jeder Datenpunkt der BAS wird als ein BACnet-Objekt modelliert. Zur Datenaktualisierung kann sowohl die COV-Subscription als auch der ReadPropertyMultiple (RPM)-Service verwendet. Die Zykluszeit für die Resubscription sowie die Pollintervalle für die RPM-Services können über 4 Prioritätsgruppen parametriert werden (siehe Konfiguration BACnet-Treiber).

Eine Zeitsynchronisierung vom BACnet-Treiber in Richtung BAS kann parametriert werden.

VERBINDUNGSAUFNAHME UND DATENAUSTAUSCH

Der BACnet-Treiber als Client sendet den „Who-Is"-Service als Broadcast an die Subnet-Mask. Die vorhandenen BAS antworten mit einem „I-Am". Die Treiber-Variable vom Typ „Komponentenstatus", die dem Namen des Deviceobjekts entspricht, wird auf 0 (OK) gesetzt.

Der Client erfragt mit dem Service „Who-Has(Object_Name)" die BACnet-Objekte des jeweiligen Servers. Dieser teilt dem Client mit dem Service „I-Have" die Objekte, einschl. der Properties Object_Instance und Objekt_Type mit. Unbeantwortete „Who-Has"-Services werden entsprechend der parametrierten Retryanzahl wiederholt. Wenn BACnet-Objekte existieren, die auch dann nicht mit dem „I-Have"-Service antworten, wird der Komponentenstatus der jeweiligen BAS auf 1 (Initialisierungsfehler) gesetzt. Die betreffenden Objektnamen werden in das Log-File geschrieben.

Der Client startet dann die COV-Subscriptions bzw. RPM-Services für die vorhandenen Objekte. Bei Wert (Present_Value)- oder Statusänderung (Status_Flags) der COV-Objekte wird der Client per COV-Notification informiert, worauf er mit dem ReadProperty-Service den Zeitstempel (Event_Time_Stamps) vom Server für das entsprechende Objekt anfordert. Durch die RPM-Services werden die Properties Present_Value, Status_Flags und Event_Time_Stamps zyklisch vom Server angefordert.

Sendedaten (Present_Value, Status_Flags, Event_Time_Stamps) werden mit dem WritePropertyMultiple-Service zum Server gesendet. Falls eine dieser Properties auf dem Server nicht beschrieben werden kann, quittiert dies der Server mit einer Fehlermeldung („Rec_error: WritePropertyMultiple").

WERT UND STATUS SPEICHERN

Das Speichern von Wert, Status und Zeitstempel der Output-Variablen erfolgt spontan. Somit wird ein Datenverlust bei Ausfall des Rechners vermieden. Das Einlesen der gespeicherten Informationen erfolgt beim Neustart.

AUSFALLÜBERWACHUNG

Um die Präsenz des Servers zu überprüfen, sendet der Client zyklisch „Who-Is"-Telegramme. Die Zykluszeit wird im Dialog Treiber-Konfiguration-BACnet-Einstellungen parametriert.

Wenn der Server innerhalb eines bestimmten Zeitfensters nicht mehr mit „I-Am" antwortet, gilt die Schnittstelle als ausgefallen. Der zugehörige virtuelle Datenpunkt vom Typ BACnet-Komponentenstatus wird auf 64 (NOK) gesetzt. Alle Prozessdatenpunkte dieser BAS werden auf gestört (Invalid) gesetzt.

Den Wiederaufbau der Verbindung meldet der Server mit einem „I-Am". Der weitere Verlauf ist identisch mit dem bei der Verbindungsaufnahme.

ZUGRIFFSVERFAHREN

Die Werte der Variablen können wahlweise spontan oder durch Polling gelesen werden. Die Auswahl erfolgt über die Eigenschaft „Priorität" der einzelnen Variablen
Wenn die Hardware das Lesen der Zeitstempel zulässt, wird den Variablen der ausgelesene Zeitstempel zugewiesen, ansonsten erhalten die Variablen den Zeitstempel vom Treiber.

Polling	**Wird die Eigenschaft „Priorität" der Variablen auf einen der Werte „Erhöht", „Hoch" oder „Höchste" gesetzt, so werden die Variablenwerte in dem in der Treiberkonfiguration, Seite „Allgemein", für die betreffende Prioritätsstufe eingestellten Intervall eingelesen. Dafür wird der BACnet-Service ReadPropertyMulti (liest die Objektproperties „present value", „status flags" und „event time stamps") verwendet.**
Spontan	Durch setzen der Variableneigenschaft „Priorität" auf den Wert „Normal" werden die Variablenwerte über den BACnet-Service COV („Change of Value") gelesen. Bei der ersten Anforderung des Variablenwertes wird eine so genannte „COV Subscription" durchgeführt, d.h. das entsprechende Objekt wird dazu aufgefordert, bei jeder Wertänderung ein entsprechendes COV-Notification-Telegramm zu senden. Die COV Subscription gilt für die Zeitspanne, die in der Treiberkonfiguration für die Prioritätsstufe „Normal" angegeben wird. Nach dieser Zeit hört das Gerät auf, Wertänderungen mitzuteilen. Wird in diesem Fall der Variablenwert weiterhin benötigt, führt der Treiber eine neuerliche COV Subscription durch.

Beim COVNotification-Empfang von BACnet-Objekten werden die Properties Present_Value und Status_Flags (FAULT <=> IBit) ausgelesen und mit dem Service ReadProperty der Event_Time_Stamps angefordert. Falls die empfangene Zeit ungültig ist (0 oder 255), stempelt der BACnet-Treiber mit der aktuellen Rechnerzeit nach.

RÜCKNAHME VON WERTEN AUS DEM PRIORITÄTENARRAY:

Für einige Objekttypen besteht die Möglichkeit, die von zenon gesetzten Werte wieder aus dem BACnet Prioritätenarray der Variable auf der Steuerung zu entfernen. Dies ist jedoch nicht direkt über den Treiber möglich, deshalb wird hierfür ein VBA Script verwendet.

Beispiel für ein VBA Script:

```
Sub ResetPrioArray()
        Dim Var As Variable
        Set Var = Variables.Item("TestVar 0")
        Var.SetValueWithStatus 0, 262144, 0, 0      '262144 = Hex 40000, entspricht dem gesetzten INVALID-Bit
End Sub
```

EINSCHRÄNKUNG

Sollte der Treiber mehrfach gestartet werden, sind für jeden Treiber unterschiedliche UDP Ports einzustellen. Diese Ports müssen auf den Steuerungen (BACNet Server) auch einstellbar sein.

Priority: Der Treiber sendet Sollwerte mit der Priorität 8

9 Funktion Treiberkommandos

Die zenon Funktion **Treiberkommandos** dient dazu, Treiber über zenon zu beeinflussen.
Mit einem Treiberkommando können Sie einen Treiber:

- starten
- stoppen
- in einen bestimmten Treibermodus versetzen
- zu bestimmten Aktionen veranlassen

Hinweis: 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.

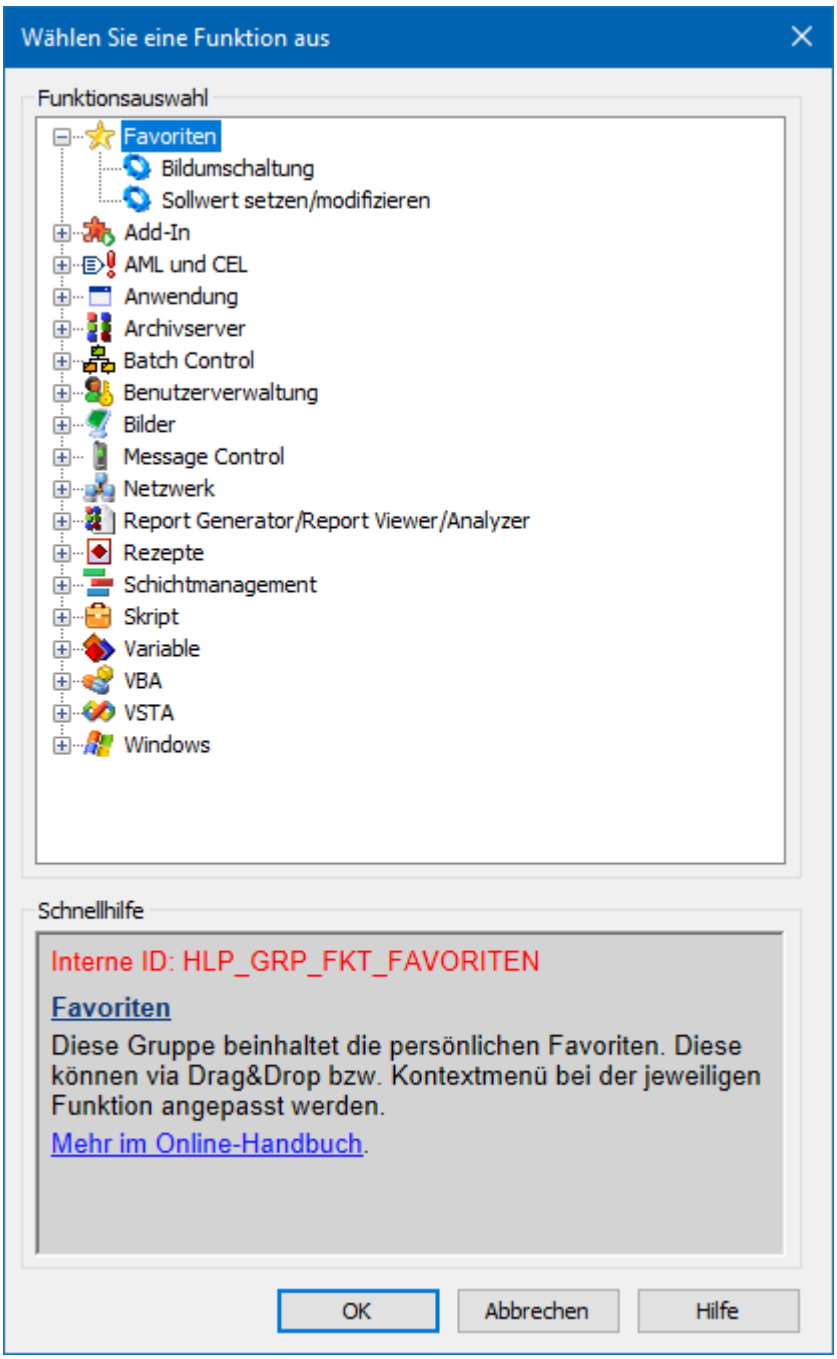
> **Achtung**
>
> Die zenon Funktion **Treiberkommandos** ist nicht ident mit den Treiberkommandos, die bei Energy-Treibern in der Runtime ausgeführt werden können!

PROJEKTIERUNG DER FUNKTION

Die Projektierung erfolgt über die Funktion **Treiberkommandos**.
Um die Funktion zu projektieren:

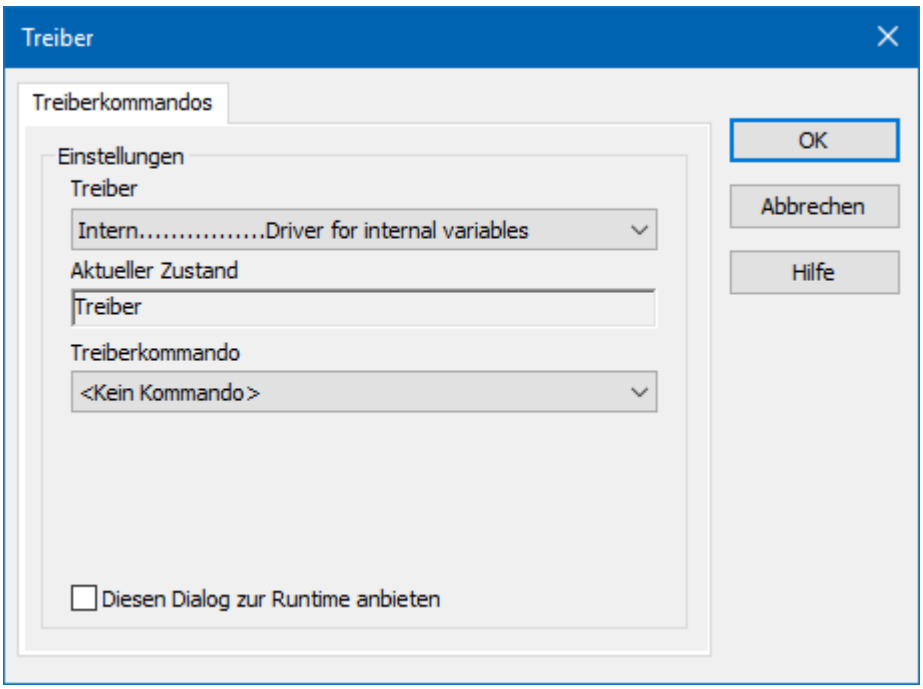
1. Legen Sie im zenon Editor eine neue Funktion an.

Der Dialog zur Auswahl einer Funktion wird geöffnet.



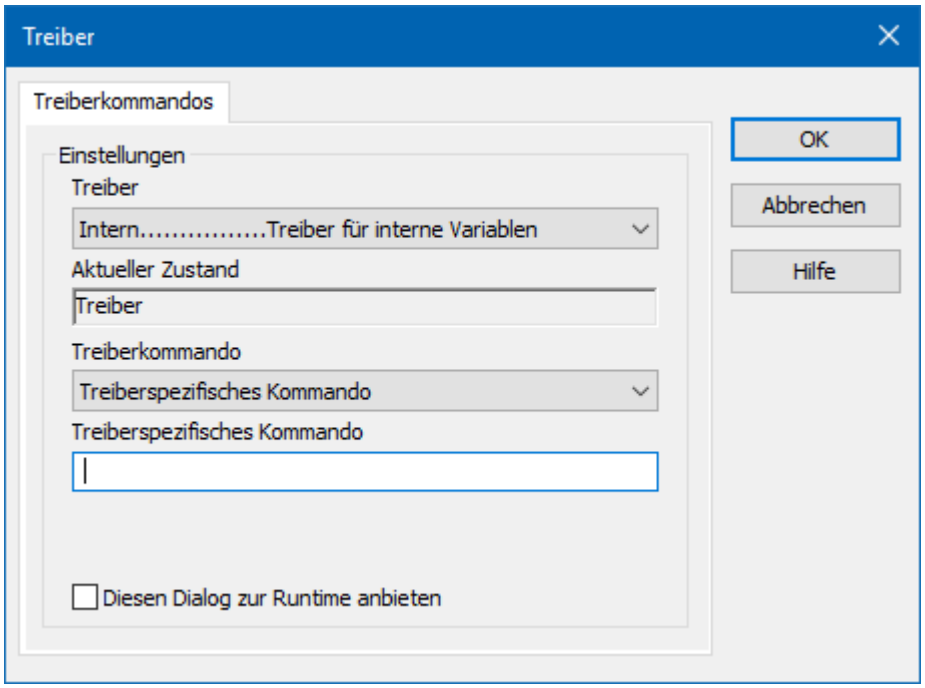
2. Navigieren Sie zum Knoten **Variable.**
3. Wählen Sie den Eintrag **Treiberkommandos**.

Der Dialog zur Konfiguration wird geöffnet.



4. Wählen Sie den gewünschten Treiber und das benötigte Kommando aus.
5. Schließen Sie den Dialog mit Klick auf **OK** und stellen Sie sicher, dass die Funktion in der Runtime ausgeführt wird.
   Beachten Sie die Hinweise im Abschnitt **Funktion Treiberkommandos im Netzwerk**.

DIALOG TREIBERKOMMANDOS



Option	Beschreibung
Treiber	Auswahl des Treibers aus der Dropdownliste. Diese enthält alle im Projekt geladenen Treibern.
Aktueller Zustand	Fixer Eintrag, der vom System gesetzt wird. In aktuellen Versionen ohne Funktion.
Treiberkommando	Auswahl des gewünschten Treiberkommandos aus Dropdownliste. Details zu den konfigurierbaren Treiberkommandos siehe Abschnitt **Verfügbare Treiberkommandos**.
Treiberspezifisches Kommando	Eingabe eines für den gewählten Treiber spezifischen Kommandos. **Hinweis:** Nur verfügbar, wenn für die Option **Treiberkommando** der Eintrag *Treiberspezifisches Kommando* gewählt wurde.
Diesen Dialog zur Runtime anbieten	Konfiguration, ob die Konfiguration in der Runtime geändert werden kann: ▸ *Aktiv*: Dieser Dialog wird in der Runtime vor dem Ausführen der Funktion geöffnet. Die Konfiguration kann damit in der Runtime vor der Ausführung noch geändert werden. ▸ *Inaktiv*: Die Editor-Konfiguration wird in der Runtime beim Ausführen der Funktion angewendet. Default: *inaktiv*

DIALOG BEENDEN

Option	Beschreibung
OK	Übernimmt Einstellungen und schließt den Dialog.
Abbrechen	Verwirft alle Änderungen und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

VERFÜGBARE TREIBERKOMMANDOS

Diese Treiberkommandos stehen - abhängig vom gewählten Treiber - zur Verfügung:

Treiberkommando	Beschreibung
Kein Kommando	Es wird kein Kommando gesendet. Damit kann auch ein bereits bestehendes Kommando aus einer projektierten Funktion entfernt werden.
Treiber starten (Onlinemodus)	Treiber wird neu initialisiert und gestartet. **Hinweis:** Wenn der Treiber bereits gestartet wurde, muss er gestoppt werden. Erst dann kann der Treiber wieder neu initialisiert und gestartet werden.
Treiber stoppen (Offlinemodus)	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 eines treiberspezifischen Kommandos. Öffnet Eingabefeld für die Eingabe eines Kommandos.
Treiber Sollwertsetzen aktivieren	Sollwert setzen auf Treiber ist möglich.
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).
Treiber in Simulationsmodus zählend	Treiber wird in den zählenden Simulationsmodus gesetzt. Alle Werte werden mit *0* initialisiert und in der eingestellten Updatezeit jeweils um *1* bis zum Maximalwert inkrementiert und beginnen dann wieder

Treiberkommando	Beschreibung
	bei *0*.
Treiber in Simulationsmodus statisch	Es wird keine Kommunikation zur Steuerung aufgebaut. Alle Werte werden mit *0* initialisiert.
Treiber in Simulationsmodus programmiert	Die Werte werden von einem frei programmierbaren Simulationsprojekt berechnet. Das Simulationsprojekt wird mit der zenon Logic Workbench erstellt und läuft in der zenon Logic Runtime ab.

FUNKTION TREIBERKOMMANDOS IM NETZWERK

Wenn sich der Rechner, auf dem die Funktion **Treiberkommandos** ausgeführt wird, im zenon Netzwerk befindet, werden zusätzlich weitere Aktionen ausgeführt:

- Ein spezielles Netzwerkkommando wird vom Rechner zum Server des Projekts gesendet. Dieser führt dann die gewünschte Aktion auf seinem Treiber aus.
- 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 Hilfe, um den Fehler zu finden.

10.1 Fehleranalyse

MITPROTOKOLLIEREN

Es besteht die Möglichkeit, den gesamten Telegrammverkehr zwischen BACnet-Treiber und BAS mitzuprotokollieren (siehe BACnet32_DiagnosticSetting (auf Seite 18)).

Die Ausgabe erfolgt in die Datei <Runtimepfad>\RT\FILES\zenon\custom\drivers\BACnet.log

Bzw. für Editorkommunikation (browsing) in die Datei:
<SQLpfad>\<ProjektGUID>\FILES\zenon\custom\drivers\BACnet.log.

10.2 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 Diagnosis Viewer, das mit zenon mitinstalliert wird. Sie finden es unter **Start/Alle Programme/zenon/Tools 8.20 -> Diagviewer.**

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

%ProgramData%\COPA-DATA\LOG.

Achtung: Mit den Standardeinstellungen zeichnet ein Treiber nur Fehlerinformationen auf. Mit dem Diagnosis 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 Diagnosis Viewer kann man auch:

- neu erstellte Einträge in Echtzeit mitverfolgen
- die Aufzeichnungseinstellungen anpassen
- den Ordner, in dem die LOG-Dateien gespeichert werden, ändern

Hinweise:

1. Der Diagnosis Viewer zeigt alle Einträge in UTC (Koordinierter Weltzeit) an und nicht in der lokalen Zeit.
2. Der Diagnosis 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.
3. 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**.
4. 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.
5. 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 Diagnosis Viewers weiter aufgezeichnet.

> **Achtung**
>
> Unter Windows CE werden aus Ressourcegründen Fehler standardmäßig nicht protokolliert.

Weitere Informationen zum Diagnosis Viewer finden Sie im Handbuch Diagnosis Viewer.

10.3 Treiberüberwachung

Die Runtime überwacht die Verfügbarkeit des Treibers via Watchdog. Ist ein Treiber nicht mehr verfügbar, wird für alle angemeldeten Variablen des Treibers zusätzlich das Statusbit *INVALID* gesetzt.

Mögliche Ursachen für Auslösen des Watchdogs:

- Der Treiberprozess läuft nicht mehr.

  Überprüfen Sie im Task Manager ob die Treiber-Exe noch läuft.

- Betriebssystem ist mit höher priorisierten Prozessen ausgelastet.

  Überprüfen Sie die Konfiguration Ihres Systems auf zu wenig Arbeitsspeicher und zu geringe CPU-Leistung. In diesem Fall erfolgt das Rücksetzen des *INVALID* Statusbits durch den Treiber nur bei Wertänderung auf der Gegenstelle. Statische Werte behalten das *INVALID* Statusbit bis zum nächsten Start der Runtime oder des Treibers.

WATCHDOG KONFIGURATION

Für die Überwachung der Kommunikation zur Runtime wird die Verbindung zum Treiber in einem fix vorgegebenen Zeitraum von 60 Sekunden überprüft. Dieser Vorgang wird mehrmals wiederholt. Konnte nach 5 Versuchen (= innerhalb von 5 Minuten) keine valide Verbindung zum Treiber erkannt werden, wird das *INVALID*-Bit bei der angemeldeten (*advised*) Variablen gesetzt. Zusätzlich wird das *INVALID*-Bit auch gesetzt, wenn neue Variablen angemeldeten werden. Das *INVALID*-Bit wird nicht mehr zurückgesetzt.

Dafür werden entsprechende LOG-Einträge erstellt.

LOG-EINTRAG

Bei Auslösen des Watchdogs wird eine Fehlermeldung im LOG protokolliert:

Parameter	Beschreibung
Communication with driver:<drvExe>/<drvDesc>(id:<drvId>) timed out. No communication for <time> ms.	Keine Kommunikation mit Treiber innerhalb der angegeben Zeit. ▸ *<time>*: Zeitangabe in Millisekunden ▸ *<drvDesc>*: Treibername

Parameter	Beschreibung
	▸ *<drvExe>*: Treiber EXE-Name ▸ *<drvId>*: Treiber-ID im zenon Projekt
Communication with %s timed out. Invalid-Bit will be set.	Die Kommunikation zum Treiber *%s* konnte bei 5 Versuchen nicht innerhalb von 60 Sekunden aufgebaut werden. Bei der Variable wird das *INVALID*-Bit gesetzt.
Communication with %s timed out. Timeout happened %d times	Die Kommunikation zum Treiber *%s* konnte *%d* Mal nicht innerhalb von 60 Sekunden aufgebaut werden.

10.4 Checkliste

Ist die Steuerung an die Stromversorgung angeschlossen
Sind die Teilnehmer im TCP/IP Netz verfügbar
Kann die Steuerung über den PING Befehl erreicht werden
Kann die Steuerung auf dem entsprechenden Port über TELNET erreicht werden
Sind Steuerung und PC mit dem entsprechenden Kabel verbunden
Wurde der richtige COM Port ausgewählt
Stimmen die seriellen Kommunikationsparameter (Baudrate, Parität, Start/Stop Bits,...) überein
Wird der COM Port durch eine andere Anwendung blockiert
Wurde die Netzadresse sowohl im Treiberdialog als auch in den Adresseigenschaften der Variable korrekt eingestellt.
Wird in der Variable der richtige Objekttyp verwendet
Stimmt die Offset-Adressierung der Variable mit der in der Steuerung überein
Analyse mit Hilfe des DiagViewer -> Welche Meldungen werden angezeigt

11 PICS (Protocol Implementation Conformance Statement)

Date	September 21, 2005
Vendor Name	COPA-DATA GmbH
Product Name	BACnet-driver for process control system (HMI/SCADA)
Product Model Number:	Version 6.20 SP1
Applications Software Version	6.20.1
BACnet protocol Revision	2

PRODUCT DESCRIPTION

The BACnet driver allows communication and data exchange between one or more BACnet-capable devices and the SCADA-Runtime. Therefor it's required that the connected BACnet devices are operating as servers. In the BACnet driver only the Client functionality is implemented.

BACNET STANDARDIZED DEVICE PROFILE (ANNEX L):

X	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)

ADDITIONAL BACNET INTEROPERABILITY BUILDING BLOCKS SUPPORTED (ANNEX K):

DS-RP-A, DS-RPM-A, DS-WP-A, DS-WPM-A, DS-COV-A, DM-DDB-A, DM-DOB-A, DM-TS-A, DM-UTC-A, SCHED-A

SEGMENTATION CAPABILITY:

	Segmented requests supported	Window Size _____
	Segmented responses supported	Window Size _____

STANDARD OBJECT TYPES SUPPORTED:

Analog-Input, Analog-Output, Binary-Input, Binary-Output, Multi-State-Input, Multi-State-Output, Schedule, Device

OBJEKT DEFINITIONEN

1. ANALOG INPUT

Property	Client uses
Present_Value	R
Status_Flags	R
Event_Time_Stamps	R

2. ANALOG OUTPUT

Property	Client uses
Present_Value	W
Status_Flags	W
Event_Time_Stamps	W

3. ANALOG VALUE

Property	Client uses
Present_Value	R / W
Status_Flags	R / W
Event_Time_Stamps	R / W

4. BINARY INPUT

Property	Client uses
Present_Value	R
Status_Flags	R
Event_Time_Stamps	R

5. BINARY OUTPUT

Property	Client uses
Present_Value	W
Status_Flags	W
Event_Time_Stamps	W

6. BINARY VALUE

Property	Client uses
Present_Value	R / W
Status_Flags	R / W
Event_Time_Stamps	R / W

7. MULTI STATE INPUT

Property	Client uses
Present_Value	R
Status_Flags	R
Event_Time_Stamps	R

8. MULTI STATE OUTPUT

Property	Client uses
Present_Value	R
Status_Flags	R
Event_Time_Stamps	R

9. MULTI STATE VALUE

Property	Client uses
Present_Value	R / W
Status_Flags	R / W
Event_Time_Stamps	R / W

10. SCHEDULE

Property	Client uses
Effective_Period	R / W (as String)
Weekly_Schedule	R / W (as String)
Exception_Schedule	R / W (as String)

11. DEVICE

Property	Client uses
Local_Date	W
Local_Time	W

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

NETWORKING OPTIONS

	Router, Clause 6 - Routing configurations: _____
	Annex H, BACnet Tunnelling Router over IP

	Router, Clause 6 - Routing configurations: _____
	BACnet/IP Broadcast Management Device (BBMD)

CHARACTER SETS SUPPORTED

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

OPTIONAL SERVICE-PARAMETERS SUPPORTED

ReadProperty, ReadPropertyMultiple, WriteProperty, WritePropertyMultiple, SubscribeCOV, COVNotification, Who-Is, I-Am, Who-Has, I-Have, TimeSynchronization, TimeUTCSynchronization