zenon Treiber Handbuch

RemoteRT

v.8.00

©2018 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. RemoteRT 5

3. REMOTERT - Datenblatt 6

4. Treiber-Historie 8

5. Voraussetzungen 8

5.1 Installation und Ablauf 9

5.2 Connector 10

6. Konfiguration 13

6.1 Anlegen eines Treibers 14

6.2 Einstellungen im Treiberdialog 17

6.2.1 Allgemein 18

6.2.2 Remote Runtime Verbindungen 22

7. Variablen anlegen 26

7.1 Variablen im Editor anlegen 26

7.2 Adressierung 30

7.3 Treiberobjekte und Datentypen 30

7.3.1 Treiberobjekte 31

7.3.2 Zuordnung der Datentypen 31

7.4 Variablen anlegen durch Import 33

7.4.1 XML Import 33

7.4.2 DBF Import/Export 34

7.5 Kommunikationsdetails (Treibervariablen) 41

8. Polling zur Runtime 46

9. Treiberspezifische Funktionen 47

10. Funktion Treiberkommandos 47

11. Fehleranalyse 50

11.1 Analysetool 50
11.2 Checkliste 51

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 Support-Team, das 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. RemoteRT

Mit dem **Remote Runtime Treiber** (**RemoteRT.exe**) ist es möglich, Variablenwerte in einer laufenden Runtime von einer anderen Runtime auszulesen und zu übernehmen. Die Verbindung erfolgt mittels eines Connectors.

Das Lesen der Werte erfolgt blockweise. Ist kein blockweises Lesen möglich werden die Werte einzeln (ein Wert nach dem anderen) gelesen.

Der Treiber unterscheidet sich mit seinem Connector-Konzept grundsätzlich von anderen zenon Treibern. Das Anfordern der Daten aus der Quell-Runtime entspricht in etwa dem Teachen einer Rezeptur.

Der Treiber adressiert ausschließlich namensbasiert über die **Symbolische Adresse**.

Info

Eine Anwendung des **Remote Runtime Treibers** ist die Verbindungen zu älteren Runtimes, ohne die bestehende Projektierung zu ändern.

3. REMOTERT - Datenblatt

Allgemein:	
Treiberdateiname	REMOTERT.exe
Treiberbezeichnung	Remote Runtime Treiber
Steuerungs-Typen	Remote Runtime
Steuerungs-Hersteller	zenon system driver; COPA-DATA;

Treiber unterstützt:	
Protokoll	proprietary;
Adressierung: Adress-basiert	--
Adressierung: Namens-basiert	X
Kommunikation spontan	--
Kommunikation pollend	X
Online Browsing	--
Offline Browsing	--
Echtzeitfähig	--
Blockwrite	--

Modemfähig	--
Serielles Logging	--
RDA numerisch	--
RDA String	--
Hysterese	--
erweiterte API	--
Unterstützung von Statusbit **WR-SUC**	--
alternative IP-Adresse	--

Voraussetzungen:	
Hardware PC	--
Software PC	--
Hardware Steuerung	--
Software Steuerung	SCADA Connector Container- zu finden im Additional Software Ordner auf dem Installationsmedium
Benötigt v-dll	X

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

4. Treiber-Historie

Datum	Build Nummer	Änderung
04.09.14	7.20.0.14252	Treiberdokumentation wurde neu erstellt
		Unterstützung für blockweises Lesen

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 Installation und Ablauf

INSTALLATION

Mit der Installation von zenon werden am Zielrechner alle nötigen Dateien installiert.

> **Achtung**
>
> *Damit der Treiber gestartet werden kann, muss die **zrsProvider.dll** installiert sein.*

Am Quellrechner muss der Connector Container eventuell manuell installiert werden. Dazu steht am zenon Installationsmedium ein eigenes Setup zur Verfügung. Pfad: **AdditionalSoftware\COPA-DATA SCADA Runtime Connector\setup86_connectors.exe**.

> **Info**
>
> *Der Connector Container wird beim Setup als Autostart-Applikation eingetragen. Beim Anmelden eines Benutzers wird er in dessen Kontext gestartet. Dieser muss auch dem Benutzer-Kontext entsprechen, in dem die Runtime läuft. Wird die Runtime als Dienst gestartet, muss auch der Connector Container über das **Startup Tool** als Dienst gestartet werden.*

ABLAUF

- Beteiligte Systeme:
  - Lokale Runtime (SCADA): Hier läuft der Remote Runtime Treiber.
  - Remote Runtime (SPS): Hier läuft der Connector Container.
- Adressierung:
  - Lokale Runtime (SCADA): **Symbolische Adresse**.
  - Remote Runtime (SPS): **Name** der Variablen.
- Die Variable auf der lokalen Runtime gibt dem Treiber alle Informationen, die er benötigt:
  - **Netzadresse**: Wird zur IP-Adresse und zum Projektnamen der Remote Runtime in der Treiberkonfiguration gemappt.
  - **Symbolische Adresse**: Muss dem **Name** der Variablen im zenon Projekt entsprechen, das auf der Remote Runtime läuft.

    > **Achtung**
    >
    > *Für den Remote Runtime Treiber muss am lokalen System für jede gewünschte Variable die Eigenschaft **Symbolische Adresse** konfiguriert werden. Ist diese leer, wird die Variable ignoriert.*

VERBINDUNG UND UPDATE-ZEIT

Der Treiber baut eine Verbindung zum Connector Container am jeweiligen Zielrechner zum TCP Port 50778 auf. Dieser Port muss ansprechbar und in der Firewall frei gegeben sein.

In jedem Updatezyklus gibt es pro **Netzadresse** einen Leseversuch. Je Leseversuch wird eine Verbindung aufgebaut, die Anfrage abgehandelt und die Verbindung geschlossen.

Abhängig von der Grundlast des Quellsystems und der Anzahl der angeforderten Variablen bzw. dem Zyklus der abgefragten Werte kann es zu einer erheblichen Systemlast am Quellsystem kommen.

Empfehlung: Die **Update Zeit Global** sollte größer als `1000 ms` sein.

Hinweis: Die pollende Treiberverbindung mit Update-Zeiten von 1 Sekunde und mehr benötigt entsprechend Zeit. Planen Sie langsame Reaktionen ein.

5.2 Connector

Ein Connector stellt das Bindeglied zwischen Datenquelle dar. Connectoren können bestehen aus:

- Connector-Stub
- Connector-Container
- Connector-Plugin

Es wird unterschieden zwischen:

- SQL-basierende Connectoren: Diese werden direkt im Connector-Stub ausgeführt.
- C++ DLLs: Diese funktionieren als Plugins für den Connector-Container.

Externe Laufzeitdaten werden vom Quellsystem per TCP-Verbindung abgefragt. Diese Verbindung wird zwischen dem Connector-Stub beim SQL Server bzw. dem Connector-Container für Plugins aufgebaut.

VORAUSSETZUNGEN

Für den Einsatz von Connectoren müssen folgende Voraussetzungen erfüllt werden:

- In der zenon Runtime muss der Event-Mechanismus der COM Schnittstelle aktiviert sein. Dazu muss in der **zenon6.ini** folgender Eintrag gesetzt sein:
  ```
  [VBA]
  EVENT=1
  ```
- Der Port 50778 muss frei und offen sein (z. B. in der Firewallkonfiguration).

Info

Connectoren können nicht mehrfach gestartet werden.

Der Connector wird beendet, wenn beim Herstellen der Netzwerkverbindung eines folgenden Ereignisse auftritt:

- Fehler beim Erzeugen des Sockets
- Fehler beim Öffnen des Listening-Ports
- Fehler beim Starten des Listenings
- Fehler beim Annehmen einer Client-Verbindung

BEGRENZUNGEN UND PERFORMANCE

Für den SCADA Runtime Connector gilt:

- Timeout: ist unabhängig vom Report-Timeout.
  Default: 5 Minuten (konfigurierbar)
- Variablen: Nur Variablen, die in den Metadaten angeführt werden, werden abgefragt
- String-Variable: maximal 4000 Zeichen

Die Performance eines Connectors hängt ab von der:

- Performance des Analyzer Servers
- Performance des Runtime Servers
- Last des Runtime Servers (Connector läuft mit niedriger Priorität)
- Netzwerkleistung und Netzwerkauslastung

REGELN FÜR ZEITFILTER

Zeitfilter werden wie folgt angewandt:

- Zeitstempel:
  - Startzeit: eingestellte Filterzeit plus eine Millisekunde
  - Endezeit: eingestellte Filterzeit
- Verdichtungsarchive: Wert am Beginn eines Intervalls repräsentiert immer den Verdichtungswert des vorhergegangenen Intervalls.
- Zeitstempel in Basisarchiven: Nachlauf im Millisekundenbereich zum Archivzyklus.

Beispiel Zeitstempel:

- Filter `10:00` bis `11:00` Uhr: wird als `10:00:001` bis `11:00:000` interpretiert.

- Filter `1 Tag`: beginnt eine Millisekunde nach Mitternacht.

CONNECTOR-STUB

Der Connector-Stub ist eine DLL, die von den **Table Valued User Defined Functions** im SQL Server benutzt wird, um Daten vom Connector-Container auf einem Quellrechner abzufragen. Sie

- liest notwendige Metadaten (Projektnamen, Server, Standby, usw.) aus dem SQL Server ein
- baut bei Queries eine TCP-Verbindung zum Connector-Container auf
- sendet eine Anfrage
- empfängt die Antwort darauf

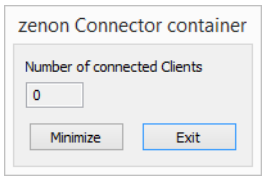
Es können mehrere Queries aus verschiedenen aufrufenden Threads parallel durchgeführt werden.Der Connector-Stub ist in der Lage, seine TCP Verbindung zu einem alternativen Quellrechner aufzubauen, falls der primäre Quellrechner nicht erreichbar ist. Die Namen des primären und des alternativen Quellrechners werden in der Projecttabelle in den Spalten `SERVER` und `STANDBY` eingetragen.

CONNECTOR-CONTAINER

Der Connector-Container ist eine Anwendung (EXE), die am Quellsystem läuft und die Connector-Plugins (DLLs) lädt und ausführt. Der Connector-Container ist ein normaler Anwenderprozess (kein Dienst o.ä.), der üblicherweise mit der Applikation gestartet wird, welche die Daten liefern soll. Der Connector-Container öffnet einen TCP-Port und wartet auf Query-Anfragen vom Connector-Stub, worauf er das angefragten Connector-Plugin lädt und die zur Anfrage passende Zugriffsfunktion aufruft. Die Rückgabedaten werden dann an den Connector-Stub zurück gesendet. Es können mehrere Queries von verschiedenen TCP-Verbindungen parallel durchgeführt werden, wenn das Quellsystem das unterstützt.

Im Normalbetrieb ist der Connector-Container als Icon im Task-Tray eingetragen und hat kein Hauptfenster. Zusätzliche Statusinformationen können dort über einen Statusdialog angezeigt werden.

DIALOG



Option	Beschreibung
Number of connectes Clients	Zeigt die Zahl der verbundenen Clients an.
Minimize	Minimiert den Dialog in den Infobereich der Taskleiste.
Exit	Beendet den Connector-Container.

NEUSTART

Wurde der Connector-Container beendet, kann er erneut gestartet werden durch:

- Neustart des Rechners.
- Manuellen Start.
  - Ab Windows 8: *Task-Manager -> Registerkarte -> Autostart -> Connector-Container -> Dateipfad öffnen -> Doppelklick auf* **zrsConnector.exe**.
  - Andere Betriebssysteme: *Dateipfad öffnen -> Doppelklick auf* **zrsConnector.exe**.
    Pfad 32-Bit: `%Program Files (x86)%\Common Files\COPA-DATA\Connectors`

CONNECTOR-PLUGIN

Connector-Plugins stellen die Verbindung zwischen Stub und Container her.

Das Connector-Plugin **SCADA Runtime** koppelt über die zenon API auf die zenon Runtime und kann von dort Laufzeitdaten abfragen. Historische Schichtdaten und Rezepte können nicht abgefragt werden.
Referenzen:

- Projektreferenz: zenon Projektname
- Variablenreferenz: zenon Variablenname

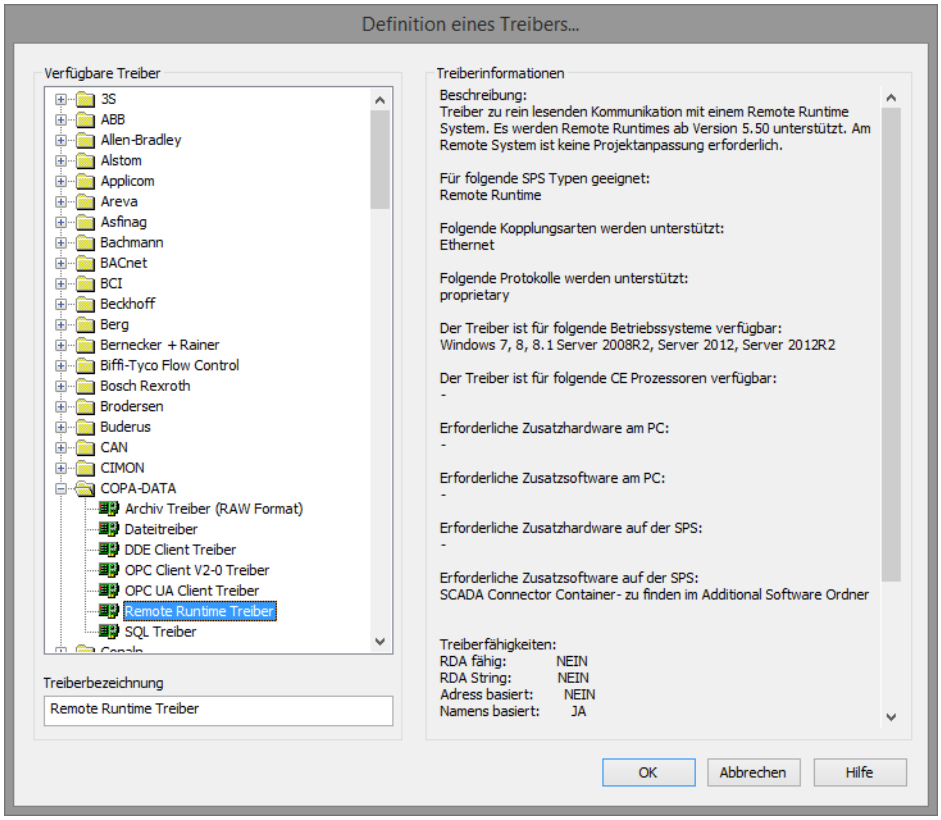
6. Konfiguration

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

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 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].`

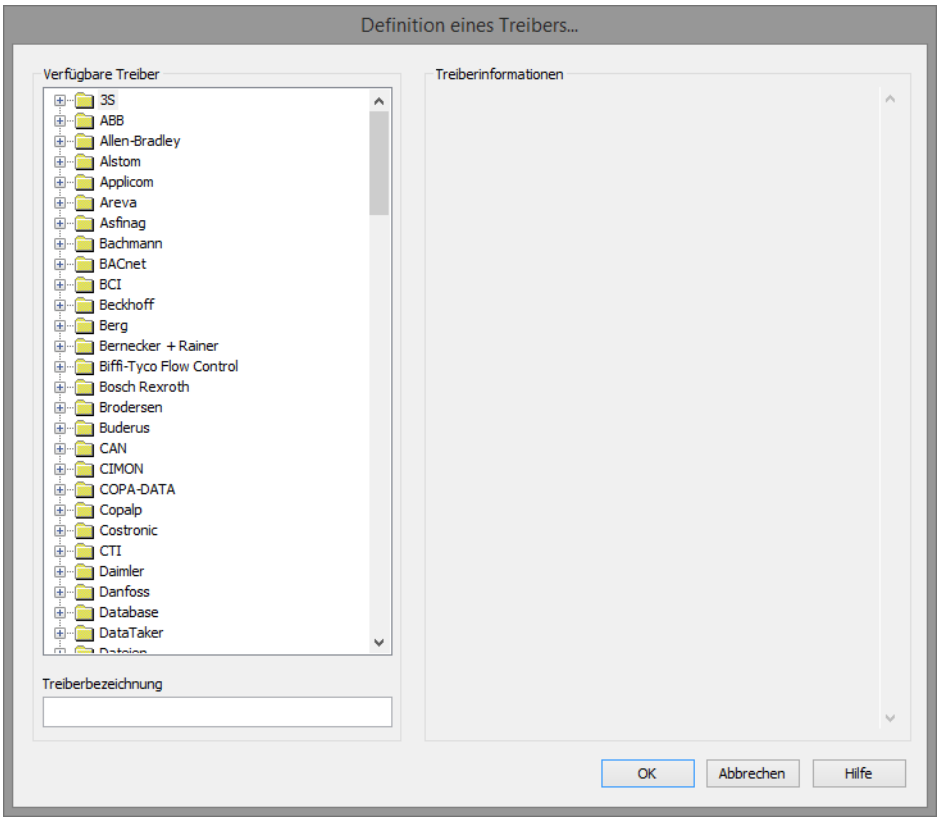
TREIBER NEU ANLEGEN

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:

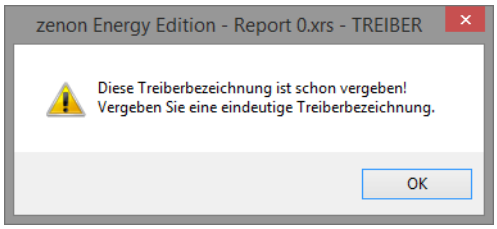
- Die **Treiberbezeichnung** muss eindeutig sein.
  Wird ein Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
  Dies wird durch Klick auf die Schaltfläche **OK** evaluiert. Ist die Treiber im Projekt bereits vorhanden wird dies mit einem Warndialog angezeigt.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

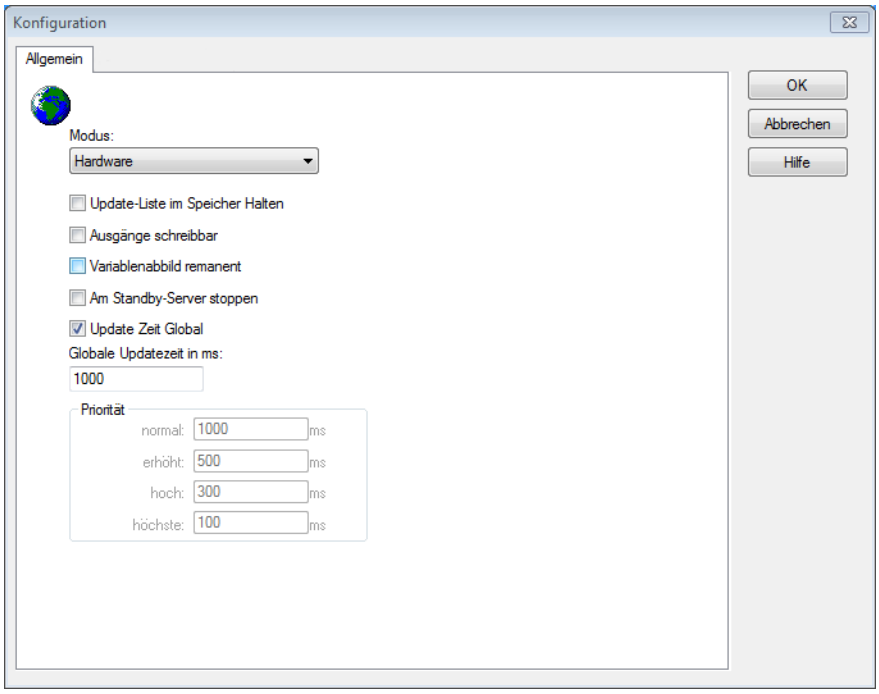
> **Info**
>
> In einem zenon Projekt müssen nur die benötigten Treiber vorhanden sein. Treiber können bei Bedarf zu einem späteren Zeitpunkt hinzugefügt werden.

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. 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. 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 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.

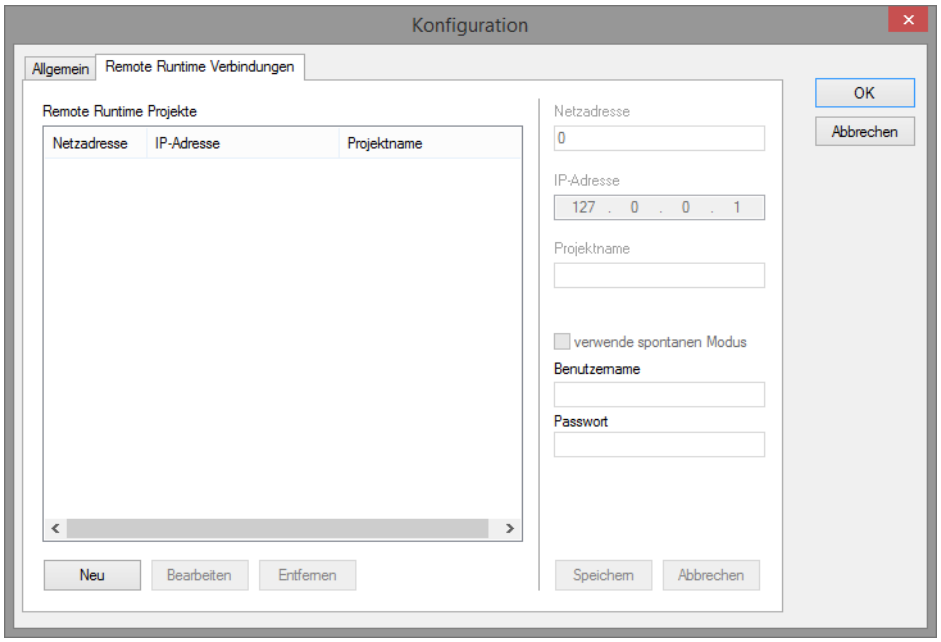
UPDATE ZEIT SPONTANE TREIBER

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 Remote Runtime Verbindungen

Konfiguration der Verbindungen zur Remote Runtime:



LISTE PROJEKTIERTER VERBINDUNGEN

Parameter	Beschreibung
Remote Runtime Projekte	Liste definierter Verbindungen. Pro Verbindung werden die **Netzadresse**, die **IP-Adresse** und der **Projektname** angezeigt. Die Projektierung erfolgt in den Eingabefeldern im Eingabebereich. Auswahl einer Verbindung zum Bearbeiten oder Löschen durch Klick auf den Eintrag. Maximum: `256 Verbindungen`
Neu	Schaltfläche zum Anlegen einer neuen Verbindung. Klick auf Schaltfläche ermöglicht Eingabe im Eingabebereich. Nur aktiv, wenn der Dialog nicht im Bearbeitungsmodus ist und weniger als 256 Verbindungen in der Liste enthalten sind.
Bearbeiten	Schaltfläche für den Wechsel in den Bearbeitungsmodus. Klick auf Schaltfläche ermöglicht die Bearbietung einer bestehenden Verbindung im Eingabebereich. Nur aktiv, wenn der Dialog nicht im Bearbeitungsmodus ist und in der Liste eine Verbindung ausgewählt ist.
Löschen	Schaltfläche zum Löschen einer bestehenden Verbindung. Klick löscht die ausgewählte Verbindung aus der Liste. Nur aktiv, wenn der Dialog nicht im Bearbeitungsmodus ist und in der Liste eine Verbindung ausgewählt ist. Achtung: Gewählte Verbindung wird ohne Rückfrage gelöscht.

VERBINDUNG ANLEGEN ODER BEARBEITEN

Diese Eigenschaften werden durch Klick auf die Schalftläche **Neu** oder **Bearbeiten** freigeschaltet.

Parameter	Beschreibung
Netzadresse	zenon **Netzadresse** der Verbindung. Nur aktiv, wenn der Dialog im Bearbeitungsmodus ist. Default: Niedrigste freie Netzadresse
IP-Adresse	IP-Adresse der Verbindung. Nur aktiv, wenn der Dialog im Bearbeitungsmodus ist. Default: Niedrigste freie Netzadresse
Projektname	Eingabe des Projektnamen für die Verbindung. Name wird automatisch in Großbuchstaben angezeigt. Nur aktiv, wenn der Dialog im Bearbeitungsmodus ist. Default: `127.0.0.1`
Spontaner Modus	Checkbox für die aktivierung spontaner Kommunikation. ▸ `aktiv`: der Treiber empfängt Werte spontan (= bei Wertänderung) Das Lesen der Werte erfolgt blockweise. Ist kein blockweises Lesen möglich werden die Werte einzeln (ein Wert nach dem anderen) gelesen. ▸ `inaktiv`: der Treiber empfängt Werte pollend. Default: `nicht aktiviert`
Benutzername	Eingabefeld für den Benutzernamen für verschlüsselte Kommunikation. **Hinweis:** Nur dann aktiv, wenn Spontaner Modus aktiviert ist.
Passwort	Eingabefeld für das Passwort für verschlüsselte Kommunikation. **Hinweis:** Nur dann aktiv, wenn Spontaner Modus aktiviert ist.
Speichern	Speichert der Konfiguration der Netzadresse. Bei Klick auf die Schaltfläche werden die Eingaben überprüft. Bei erfolgreicher Validierung werden die Änderungen übernommen und in der Liste der Verbindungen angezeigt. Validierungen: ▸ Netzadresse: Positive Ganzzahl kleiner als 256. ▸ IP-Adresse: Formal gültige IP-Adresse. ▸ Projektname: nicht leer. Nur aktiv, wenn der Dialog im Bearbeitungsmodus ist.
Abbrechen	Verwirft alle Änderungen und beendet den Bearbeitungsmodus. Nur aktiv, wenn der Dialog im Bearbeitungsmodus ist.

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.

Info

Die verschlüsselte Kommunikation für Connectoren wird im **Startup Tool** in der Registerkarte **Network configuration** mit der Eigenschaft **Encrypt Runtime Connector communication** konfiguriert.

PROJEKTIERUNG EINER VERBINDUNG

NEUE VERBINDUNG ANLEGEN

1. Klicken Sie auf die Schaltfläche **Neu**.
   Die Eingabefelder des Eingabebereichs können konfiguriert werden.
2. Tragen Sie die Verbindungsdetails ein.
3. Klicken Sie auf **Speichern**.
   Die Verbindung wird in der Liste der konfigurierten Verbindungen angezeigt.

VERBINDUNG BEARBEITEN

1. Wählen Sie in der Verbindungsliste die gewünschte Verbindung.
2. Klicken Sie auf die Schaltfläche **Bearbeiten**.
   Die Eingabefelder des Eingabebereichs können konfiguriert werden.
3. Ändern Sie die Verbindungsparameter.
4. Klicken Sie auf **Speichern**.

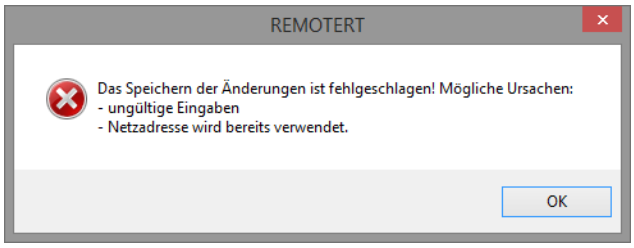
VERBINDUNG LÖSCHEN

1. Wählen Sie in der Verbindungsliste die gewünschte Verbindung.
2. Klicken Sie auf die Schaltfläche **Löschen**.
3. Die Verbindung wird aus der Liste gelöscht.
   Achtung: Die Verbindung wird ohne zusätzliche Abfrage sofort gelöscht.

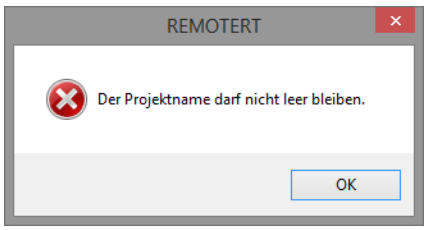
WARNDIALOGE

Bei einer Fehleingabe wird ein Warndialog angezeigt.

- Fehlerhafte Eingabe:



- Kein **Projektname** eingegeben:



7. Variablen anlegen

So werden Variablen im zenon Editor angelegt:

7.1 Variablen im Editor anlegen

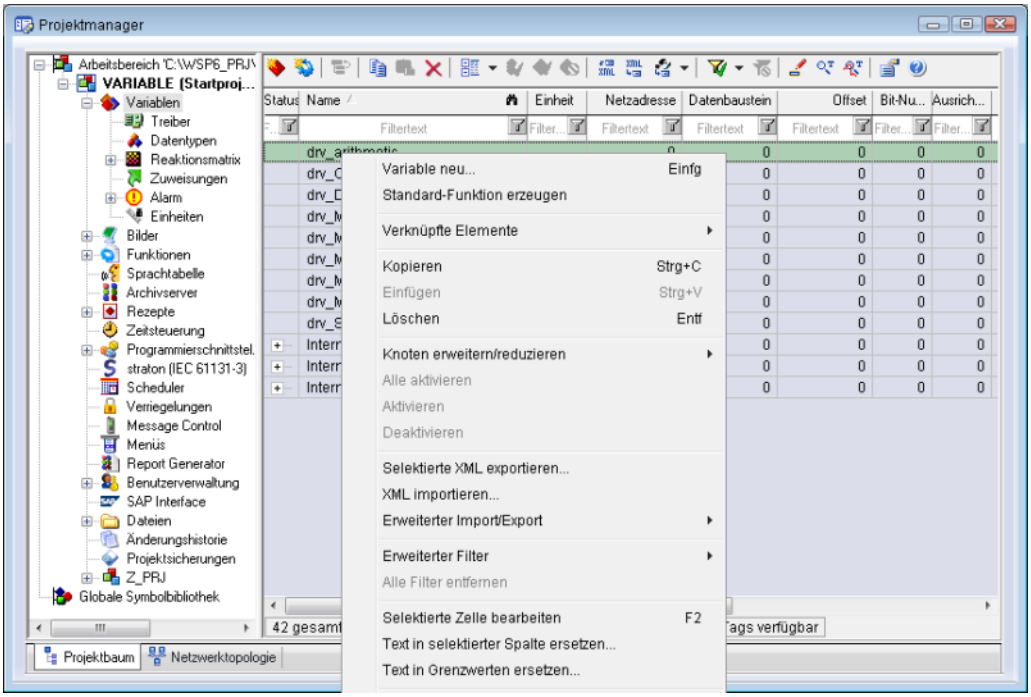
Variablen können angelegt werden:

- 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:

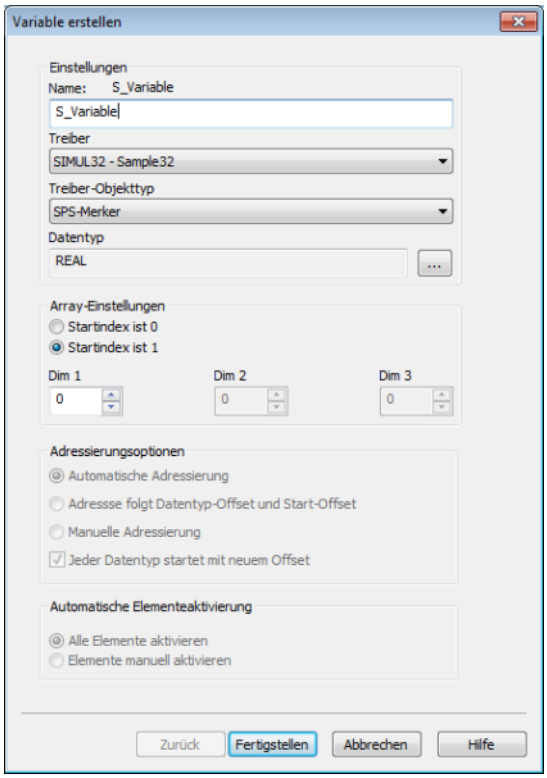
1. Wählen Sie im Knoten **Variablen** im Kontextmenü den Befehl **Variable neu**.



Der Dialog zur Konfiguration der Variable wird geöffnet.

2. Konfigurieren Sie die Variable.

3. 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: 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** (**Main.chm::/Intern.chm::/Intern.htm**)) geladen.
Treiberobjekttyp (cti.chm::/28685.htm)	Wählen Sie aus der Dropdownliste den passenden Treiberobjekttyp 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.

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.

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	Gruppe mit allgemeinen Eigenschaften.
Name	Frei vergebbarer Name. Achtung: Je zenon Projekt muss der Name eindeutig sein.
Kennung	Frei vergebbare Kennung. Z. B. für **Betriebsmittelkennung**, Kommentar usw.
Adressierung	Der Name der Variablen im Projekt auf der Remote Runtime wird ausschließlich aus der Eigenschaft **Symbolische Adresse** gelesen.
Netzadresse	Netzadresse der Variablen. Diese Adresse bezieht sich auf die Netzadresse der Verbindungsprojektierung im Treiber. Damit wird ausgewählt auf welchem Zielsystem 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.
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. Für den Remote Runtime Treiber stehen nur `Treibervariable` und `Remote-Runtime-Variable` zur Verfügung.
Treiber Anbindung/Datentyp	Datentyp der Variablen. Wird beim Erstellen der Variablen ausgewählt und kann hier geändert werden. Für den Remote Runtime Treiber stehen zur Verfügung: ▸ `BOOL` ▸ `LREAL` ▸ `WSTRING` ACHTUNG: Wenn der Datentyp nachträglich geändert wird, müssen alle anderen Eigenschaften der Variablen überprüft bzw. angepasst werden.

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
Kommunikationsdetails	35	X	X	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variablen für die statische Analyse der Kommunikation; wird zwischen Treiber und Runtime übertragen (nicht zur SPS). **Hinweis**: Die Adressierung und das Verhalten ist bei den meisten zenon Treibern gleich. Weitere Informationen dazu finden Sie im Kapitel Kommunikationsdetails (Treibervariablen) (auf Seite 41).
Remote Runtime Variable	64	X	--	BOOL, LREAL, WSTRING	

Legende:

x: wird unterstützt

--: wird nicht unterstützt

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.

Remote Runtime	Lokale Runtime	Datenart
BOOL	BOOL	8
-	USINT	9
-	SINT	10
-	UINT	2
-	INT	1
-	UDINT	4
-	DINT	3
-	ULINT	27
-	LINT	26
-	REAL	5
LREAL	LREAL	6
-	STRING	12
WSTRING	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.

Der Remote Runtime Treiber führt keine Typ-Konvertierungen durch. Die folgende Aufstellung zeigt, wie der Connector-Container Daten für einzelne zenon Datentypen liefert:

- BOOL:
  - Numerische Werte:

    0: False

    1: True i
  - Textwerte: leer
- Numerische Datentypen:
  - Numerischer Wert: Wert
  - Textwert: leer
- String Datentypen:
  - Numerischer Wert: leer
  - Textwert: Text

Darauf basierend bestimmt der Treiber den Wert je Datentyp aus den vom Connector Client gelieferten Daten wie folgt:

- BOOL: `True` wenn Feld für numerischen Wert nicht `0` ist, sonst `False`.
- LREAL: Wert des Feldes für numerische Werte.
- WSTRING: Wert des Feldes für Textwerte.

Wenn die Datentypen auf der eigenen Runtime mit jenen der Remote Runtime nicht übereinstimmen, ergeben sich Fehler bei der Wertberechnung:

Remote\Ziel	BOOL	LREAL	WSTRING
`BOOL` auf Remote Runtime	Keine Fehler	`0` wenn `False` auf Remote Runtime. `1` wenn `True` auf Remote Runtime	Immer leer.
`Numerisch` auf Remote Runtime	`False` wenn genau `0` auf Remote Runtime. Sonst `True`.	Eventuell Rundungsfehler, da Wert auf LREAL gemappt wird (limitierte Präzision).	Immer leer.
`Text` auf Remote Runtime	Immer `False`.	Immer `0`.	Eventuell wird der String gekürzt.

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

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.

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** (**main.chm::/13046.htm**).*

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 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 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)
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, 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
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.

7.5 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** (auf der Installationsmedium im Ordner `\Predefined\Variables`) verfügbar und können von dort importiert werden.

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 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 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.
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>;…;…; 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.
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. Polling zur Runtime

Beim Polling werden immer alle angemeldeten Variablen einer Netzadresse in einer Anfrage auf einmal geholt.

Ablauf beim Pollen:

1. Der Treiber triggert ein Polling für eine Netzadresse, abhängig von globaler Updatezeit, Updatezeit der höchstprioren Variablen der Netzadresse und Fehlerwartezeit nach einem fehlgeschlagenen Polling.
2. In der Treiberkonfiguration wird die Verbindungsdefinition der Netzadresse gesucht. Ist diese nicht auffindbar, wird das Polling mit einem Fehler beendet.
3. Im Treiber werden alle derzeit angemeldeten Variablen dieser Netzadresse geholt. Sind derzeit keine Variablen angemeldet, wird das Polling ohne weitere Aktion als erfolgreich beendet.
4. Die projektierte IP-Adresse wird zu einer für den Connector Client verwertbaren Struktur aufgelöst. Tritt dabei ein Fehler auf, wird das Polling mit einem Fehler beendet.
5. Die aktuellen Variablenwerte aller zu pollenden Variablen werden über den Connector Client mit der zuvor geholten IP-Addressstruktur und dem in der Verbindung vergebenen Projektnamen abgefragt. Der Variablenname auf der Remote Runtime entspricht am lokalen System dem Eintrag in der Eigenschaft **Symbolische Adresse** der Variablen.
   Tritt dabei ein Fehler auf, wird das Polling mit einem Fehler beendet. Kann eine Variable an der Remote Runtime nicht abgefragt werden kann, gilt das nicht als Fehler.
6. Die erhaltenen Werte werden je Variable vom Treiber zur Runtime geschickt:
   a) Ist ein Wert gekommen, wird er gemäß dem auf der eigenen Runtime eingestellten Datentyp zur Runtime geschickt.
   b) Ist kein Wert gekommen, wird die einzelne Variable auf `I-Bit` gesetzt.

9. Treiberspezifische Funktionen

Dieser Treiber unterstützt folgende Funktionen:

- Import von Variablenwerten einer entfernten zenon Runtime auf eine lokale zenon Runtime.

Achtung: Um die Remote Runtime nicht zu überlasten, wird eine globale Updatezeit von mindestens einer Sekunde empfohlen. Das Pollen läuft langsamer als von anderen Treibern gewohnt ab.

Fehlerwartezeit: `20 Sekunden`

10. 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

Achtung: Die zenon Funktion **Treiberkommandos** ist nicht ident mit Treiberkommandos, die bei Energy-Treibern zur Runtime ausgeführt werden können!

> **Info**
>
> *Dieses Kapitel beschreibt Standardfunktionalitäten, die für die meisten zenon Treiber gültig sind.*
> *Aber 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.*

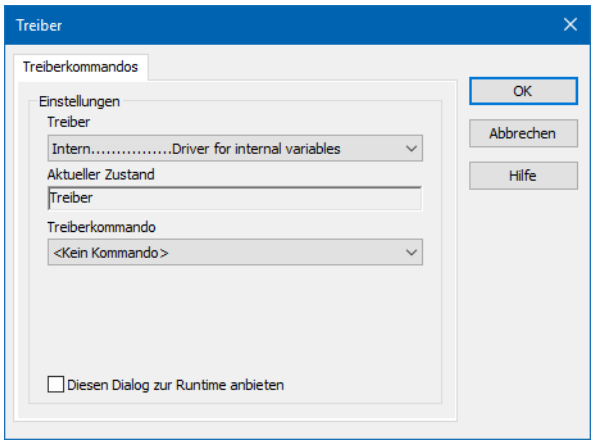
PROJEKTIERUNG DER FUNKTION

Die Projektierung erfolgt über die Funktion **Treiberkommandos**.
Dazu:

1. Legen Sie im zenon Editor eine neue Funktion an.
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 zur 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, in aktuellen Versionen ohne Funktion.
Treiberkommando	Dropdownliste zur Auswahl des Kommandos:
<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.
`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 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).
`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 bei `0`.
`Treiber in Simulationsmodus statisch`	Treiber wird in den zählenden Simulationsmodus gesetzt. Alle Werte werden mit `0` initialisiert.
`Treiber in Simulationsmodus programmiert`	Treiber wird in den zählenden Simulationsmodus gesetzt. 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.
Diesen Dialog zur Runtime anbieten	Dialog wird zur Runtime für Änderungen angeboten.

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, 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.

11. Fehleranalyse

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

11.1 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 Diagnosis Viewer (main.chm::/12464.htm), das mit zenon mitinstalliert wird. Sie finden es unter *Start/Alle Programme/zenon/Tools 8.00 -> 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 (main.chm::/12464.htm).

11.2 Checkliste

Fragen und Hinweise zur Fehlereingrenzung:

ALLGEMEINE FEHLERSUCHE

- Ist die Runtime an die Stromversorgung angeschlossen?
- Analyse mit Hilfe des Diagnosis Viewers (auf Seite 50):
  -> Welche Meldungen werden angezeigt?
- Sind die Teilnehmer im **TCP/IP**-Netz verfügbar?
- Kann die Runtime über den Befehl `Ping` erreicht werden?

  Ping: *Kommandozeile öffnen -> ping <IP-Adresse> (z. B.: ping 192.168.0.100) ->* `Eingabe-Taste` *drücken.*

Kommt eine Antwort mit Zeitangabe oder ein Timeout?

- Kann die Runtime auf dem entsprechenden Port über `Telnet` erreicht werden?

  Telnet: *Kommandozeile öffnen, telnet <IP-Adresse Port-Nummer> eingeben (z. B. für Modbus: telnet 192.168.0.100 502)* -> `Eingabe-Taste` *drücken* .

  Wird der Bildschirm schwarz, konnte eine Verbindung aufgebaut werden.

- Wurde für jede Variable, deren Wert importiert werden soll, eine **Symbolische Adresse** vergeben.
- Wurde die Netzadresse in den Adresseigenschaften der Variable korrekt eingestellt?
  - Stimmt die Adressierung mit der Konfiguration im Treiberdialog überein?
  - Entspricht die Netzadresse der Adresse der Zielstation?
- Wird in der Variable der richtige Objekttyp verwendet?

MANCHE VARIABLEN MELDEN INVALID

- INVALID Bits beziehen sich immer auf eine Netzadresse.
- Mindestens eine Variable der Netzadresse ist gestört.

WERTE WERDEN NICHT ANGEZEIGT, ZAHLENWERTE BLEIBEN LEER

Treiber läuft nicht. Überprüfen Sie die:

- Installation von zenon
- Installation des Treibers
- Installation aller Komponenten
  -> Achten Sie auf Fehlermeldungen beim Start der Runtime.

PROBLEMLÖSUNGEN

Problem	Ursache	Lösung
Der Treiber startet nicht. Es wird eine Fehlernachricht angezeigt und der Prozess sofort beendet.	Die **zrsProvider.dll** kann nicht geladen werden.	Die Container-Komponenten für zenon installieren. Bleibt das Preoblem bestehen, dann: ▶ Sicherstellen, dass die Registry-Einträge existieren. 32-Bit-Verzeichnis unter AnalyzerWrapperDir32, 64-Bit-Verzeichnis unter AnalyzerWrapperDir64 ▶ Sicherstellen, dass die zrsProvider.dll in den in den Registry-Einträgen angegebenen Pfaden existiert. 32-Bit-DLL im 32-Bit-Verzeichnis; 64-Bit-DLL im 64-Bit-Verzeichnis. ▶ Sicherstellen, dass alle Abhängigkeiten der zrsProvider.dll installiert sind (vcredist in korrekter Version; mit Dependency Walker prüfbar).
Variablen vom Remote Runtime Treiber sind auf `I-Bit`.	Welche Ursache genau zutrifft zeigen die Logeinträge im Diagnosis Viewer. Mögliche Ursachen: ▶ Netzadresse der Variablen stimmt nicht. ▶ Netzwerk-Verbindung ist nicht möglich. Z.B. wegen falscher	Überprüfen Sie: ▶ Netzwerkkonnektivität und Firewalleinstellungen. ▶ Connector Container am Zielsystem installieren und im gleichen Benutzerkontext wie die Runtime am

	IP-Adresse, Ziel-IP nicht erreichbar wegen Netzwerkausfall, Firewalleinstellungen usw. ▶ Connector Container am Zielsystem läuft nicht. ▶ Connector Container am Zielsystem läuft in einem anderen Benutzerkontext als die Runtime am Zielsystem. ▶ Projektname des Remote Runtime Projekts stimmt nicht. ▶ **Symbolische Adresse** der Variablen sind keine Variablennamen des Projekts auf der Remote Runtime.	Zielsystem starten. ▶ Projektierung überprüfen: - IP-Adresse - Projektname der Verbindungen in der Treiberkonfiguration - Symbolische Adressen Netzadressen der Variablen