COPA-DATA
do it your way

zenon Treiber Handbuch

SNMP32

v.7.60

©2017 Ing. Punzenberger COPA-DATA GmbH

Alle Rechte vorbehalten.

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

Inhalt

1. Willkommen bei der COPA-DATA Hilfe 5

2. SNMP32 6

3. SNMP32 - Datenblatt 7

4. Treiber-Historie 8

5. Voraussetzungen 9

5.1 PC 9

5.2 Steuerung 10

6. Konfiguration 10

6.1 Anlegen eines Treibers 11

6.2 Einstellungen im Treiberdialog 14

6.2.1 Allgemein 15

6.2.2 Konfiguration 19

6.2.3 SNMP Agents 21

6.2.4 Offline MIB-Liste 26

6.2.5 Beispielkonfiguration 34

7. Variablen anlegen 36

7.1 Variablen im Editor anlegen 36

7.2 Adressierung 40

7.3 Treiberobjekte und Datentypen 41

7.3.1 Treiberobjekte 41

7.3.2 Zuordnung der Datentypen 42

7.4 Variablen anlegen durch Import 43

7.4.1 XML Import 44

7.4.2 Variablen vom Treiber importieren 45

7.4.3 DBF Import/Export 50

7.5 Kommunikationsdetails (Treibervariablen) 56

8. Treiberspezifische Funktionen 61

9. Treiberkommandos 62

10. Fehleranalyse 64

10.1 Analysetool 64

10.2 Fehlerbehandlung 65

10.3 Checkliste 68

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 bitte per E-Mail an documentation@copadata.com (mailto:documentation@copadata.com).

PROJEKTUNTERSTÜTZUNG

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

LIZENZEN UND MODULE

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

2. SNMP32

BEGRIFFSDEFINITIONEN

SNMP	Simple Network Management Protocol Dieses Protokoll wird für die Fernverwaltung, Diagnose und den Schutz von Netzwerken und Hosts verwendet. SNMP lässt sich für das Management von Geräten verwenden, die einen sogenannten SNMP-Agent ausführen.
SNMP-Agent	Dient als sogenannter Provider, d.H: er stellt die Informationen eines Gerätes anderen SNMP-Management-Workstations (in unserem Fall zenon mit dem SNMP-Treiber) zur Verfügung. Diese fragen die SNMP-Agents zyklisch ab, um Informationen über die entsprechenden Geräteeigenschaften zu erhalten.
SNMP-Objekt	Bestandteile eines Gerätes, auf die der SNMP-Agent zugreifen, oder die ein SNMP-Agent modifizieren kann, werden als SNMP-Objekte bezeichnet.
MIB	Management Information Bases Stellt eine logische Datenbank dar, die eine Gruppe/Sammlung von SNMP-Objekten enthält. Da für verschiedene Gerätetypen und Protokolle unterschiedliche Netzwerkverwaltungsdienste verwendet werden können, besitzt jeder Dienst eine eigene MIB
OID	Objekt Identifier Stellt eine spezifische Detailinformation eines SNMP-Objektes dar.
TCP/IP	Transmission Control Protocol / Internet Protocol Dabei handelt es sich um einen in vier Schichten aufgebauten Satz von herstellerneutralen, häufig verwendeten Anwendungs- und Transportprotokollen. Wird vom zenon SNMP Treiber verwendet um Netzwerk-Informationen über SNMP auszulesen.
ICMP	Internet Control Message Protocol Dieses Protokoll ist für Fehler und Kontrollmeldungen an die beteiligten Rechner während des Übertragungsprozesses verantwortlich.
Ping	Dient zur Überprüfung der Erreichbarkeit eines anderen Computers.
TRAP	In SNMP wird damit eine Meldung bezeichnet, die ein Agent an ein Verwaltungssystem sendet. Damit wird das Auftreten eines Ereignisses auf dem Host angezeigt, auf dem der Agent ausgeführt wird. Der SNMP-Dienst kann beispielsweise so konfiguriert werden, dass er bei Empfang einer Informationsanforderung, die weder den korrekten Communitynamen noch einen akzeptierten Hostnamen für den Dienst enthält, ein Trap sendet.

3. SNMP32 - Datenblatt

Allgemein:	
Treiberdateiname	SNMP32.exe
Treiberbezeichnung	SNMP Treiber
Steuerungs-Typen	SNMP fähige Geräte
Steuerungs-Hersteller	SNMP;

Treiber unterstützt:	
Protokoll	SNMPv1; SNMPv2c;
Adressierung: Adress-basiert	X
Adressierung: Namens-basiert	X
Kommunikation spontan	X
Kommunikation pollend	X
Online Browsing	X
Offline Browsing	--
Echtzeitfähig	--
Blockwrite	--
Modemfähig	--
Serielles Logging	--
RDA numerisch	--
RDA String	--
Hysterese	X
erweiterte API	--
Unterstützung von Statusbit **WR-SUC**	X
alternative IP-Adresse	--

Voraussetzungen:	
Hardware PC	Standard Netzwerkkarte
Software PC	- Der Windows "SNMP Trap" Dienst muss installiert und gestartet sein- UDP traffic für port 161 und port 162 sollte an einer etwaigen Firewall freigeschalten sein (oder Anwendung snmptrap.exe)
Hardware Steuerung	--
Software Steuerung	SNMP-Agent (Server)
Benötigt v-dll	--

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

4. Treiber-Historie

Datum	Treiberversion	Änderung
07.07.08	1400	Treiberdokumentation wurde neu erstellt

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

HARDWARE

- Netzwerkkarte.

SOFTWARE

- TCP/IP-Protokoll
- Installierter und laufender SNMP-Trap Dienst.
  Hinweis: Standardmäßig wird der Dienst von Windows nicht gestartet.
- UDP Port 161 zum Senden und Empfangen offen (für andere Variablen). Das gilt auch für die Steuerung und alle Geräte im Netzwerk, über die der Datenverkehr läuft.
- UDP Port 162 zum Empfangen offen (für Traps). Das gilt auch für alle Geräte im Netzwerk, über die der Datenverkehr läuft.-Auf der Steuerung muss der Port zum Senden geöffnet sein.

5.2 Steuerung

HARDWARE

- SNMP-fähiger Netzwerkteilnehmer.

SOFTWARE

- laufender SNMP-Dienst und entsprechender SNMP-Agent.
- TCP/IP Protokoll.
- UDP Port 161 zum Senden und Empfangen offen.
- UDP Port 162 zum Senden offen (für Traps).

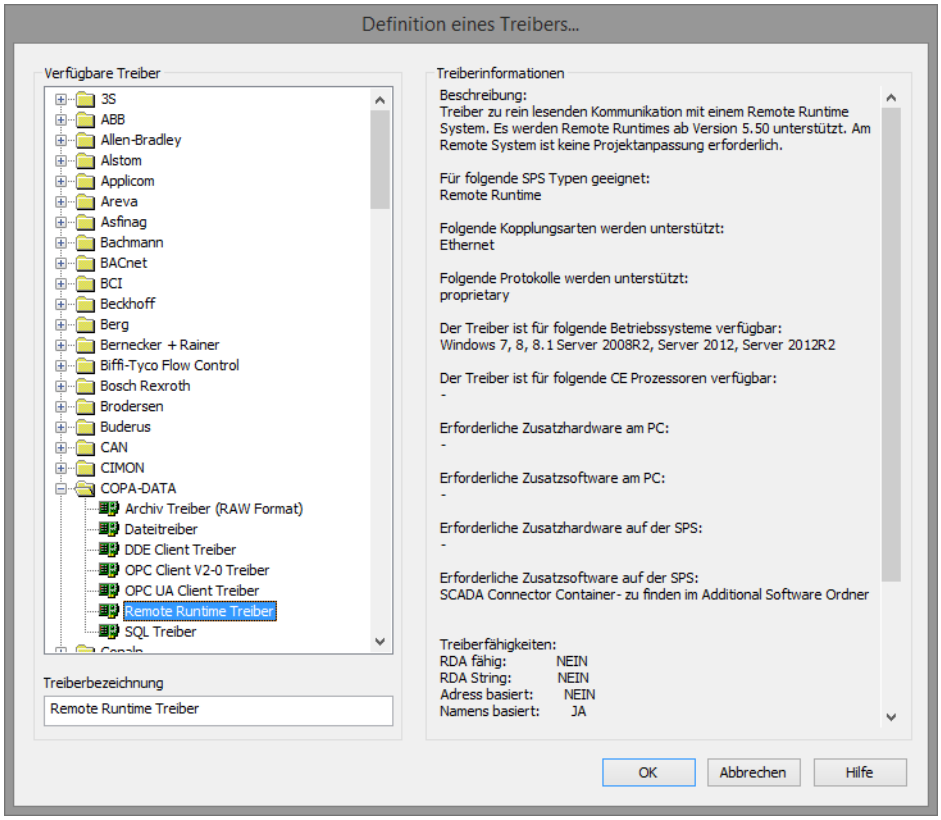
6. Konfiguration

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].`

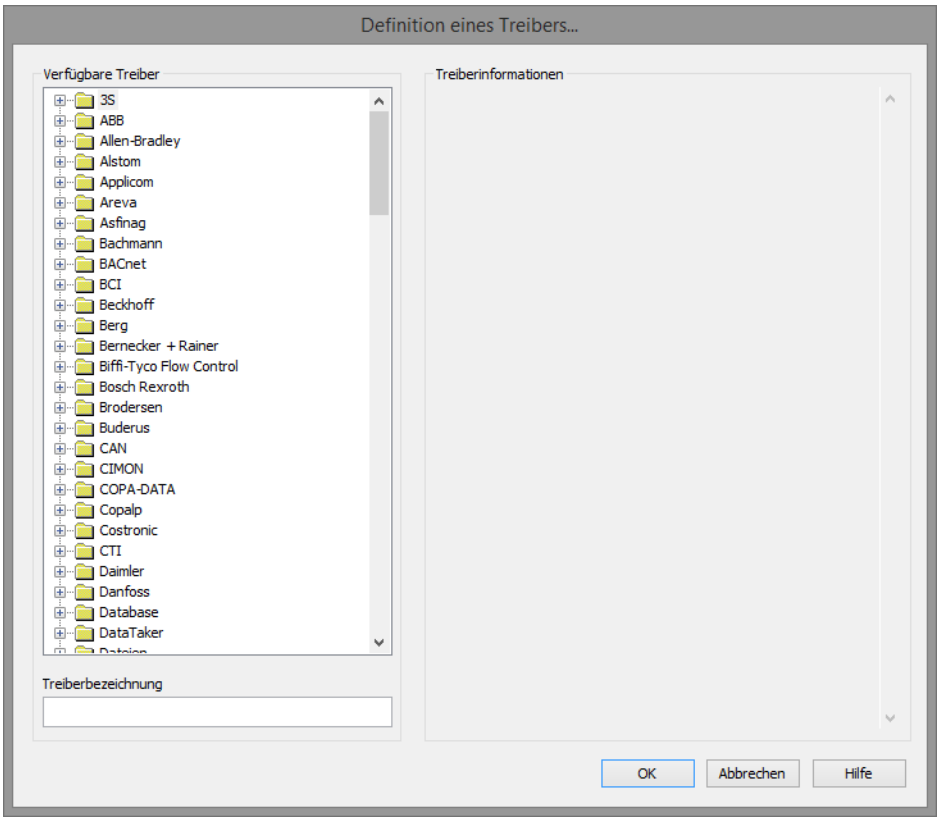
TREIBER NEU ANLEGEN

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:

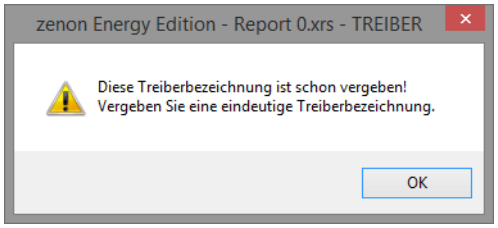
   - Die **Treiberbezeichnung** muss eindeutig sein.
     Wird ein Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
     Dies wird durch Klick auf die Schaltfläche **OK** evaluiert. Ist die Treiber im Projekt bereits vorhanden wird dies mit einem Warndialog angezeigt.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



<CD_PRODUCNTAME> PROJEKT

Bei neu angelegten Projekten werden die folgenden Treiber automatisch angelegt:

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

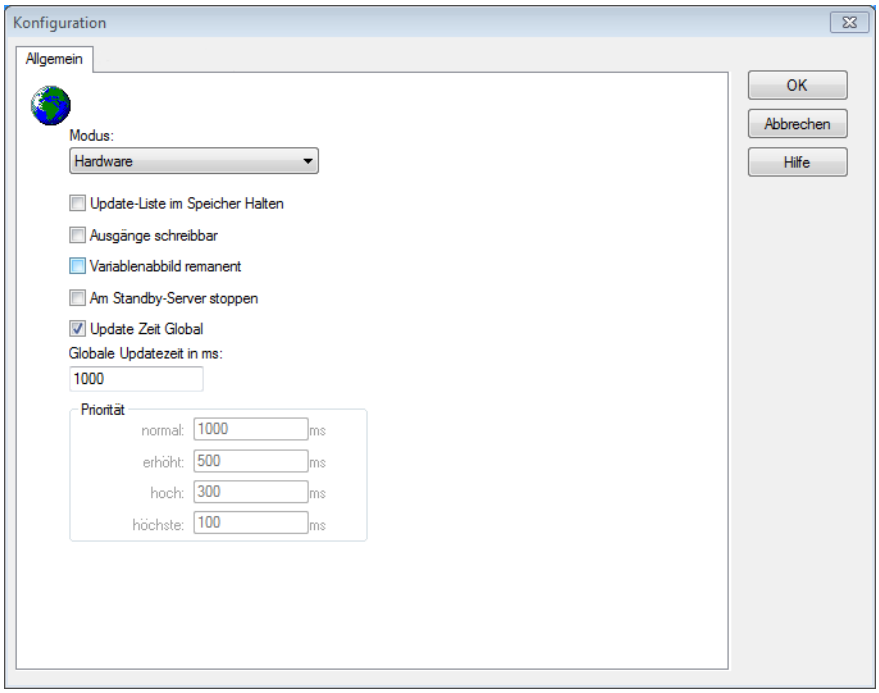
> **Info**
>
> In einem zenon Projekt müssen nur die benötigten Treiber vorhanden sein. Treiber können bei Bedarf zu einem späteren Zeitpunkt hinzugefügt werden.

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 **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** (**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. 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	`Aktiv`: Die eingestellte **Globale Update Zeit** in `ms` wird für alle Variablen im Projekt verwendet. Die bei den Variablen eingestellte Priorität wird nicht verwendet. `Inaktiv`: Die eingestellten Prioritäten werden für die einzelnen Variablen verwendet.
Priorität	Hier werden die Pollingzeiten der einzelnen 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. Zum Beispiel unterstützen spontan kommunizierende Treiber dies nicht.

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

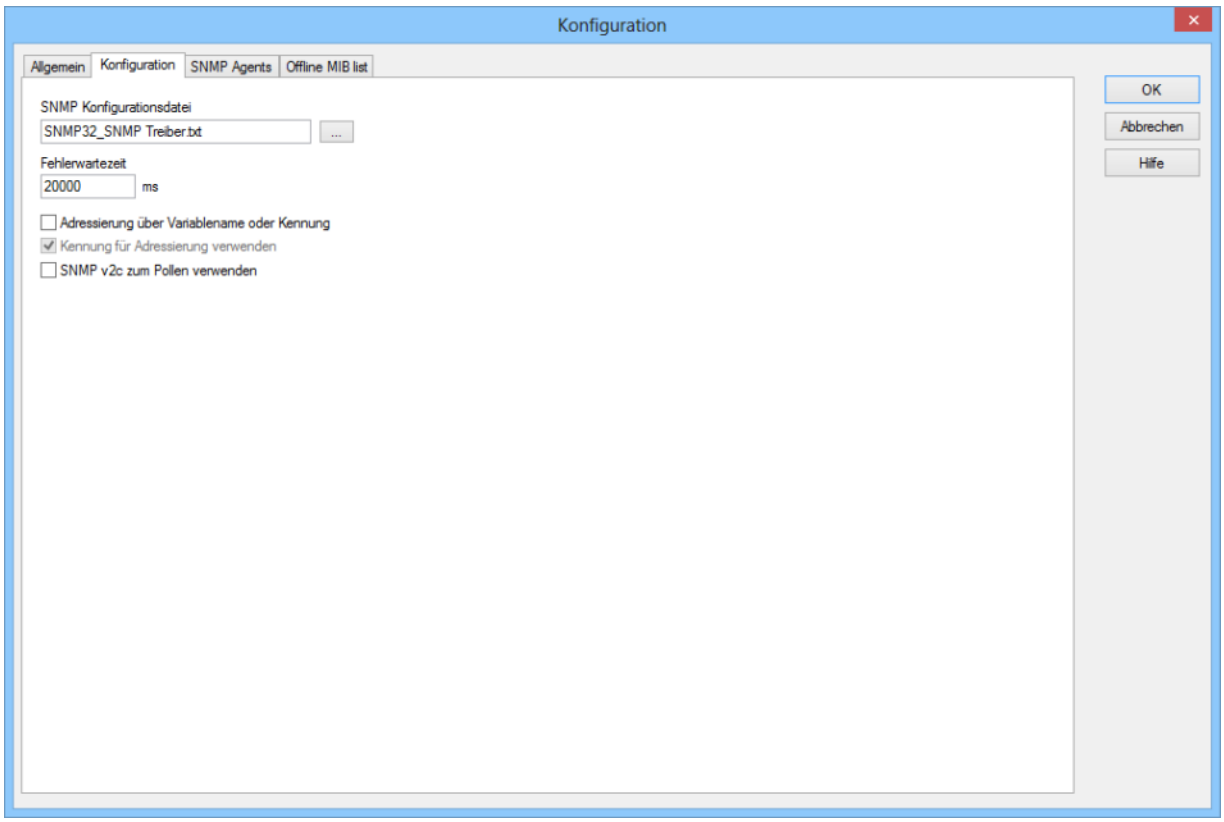
	Dialog.
Hilfe	Öffnet die Online-Hilfe.

UPDATE ZEIT ZYKLISCHE TREIBER

Für zyklische Treiber gilt:

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

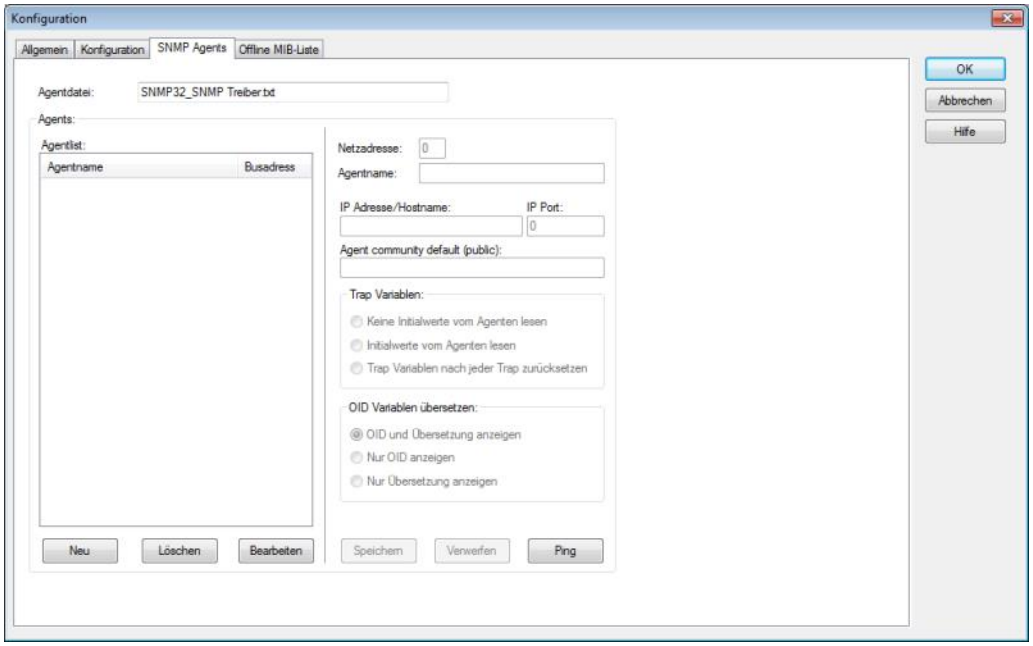
6.2.2 Konfiguration



Parameter	Beschreibung
SNMP Konfigurationsdatei	Datei, in der die Einstellungen abgespeichert werden. Sie befindet sich im zenon Projektverzeichnis.
Fehlerwartezeit (ms)	Die Zeit in Millisekunden, die gewartet wird, bis ein Fehler als solcher gewertet wird. D.h.: Sind SNMP-Agents, z. B. durch eine kurzzeitige Auslastung des Netzwerks nicht erreichbar, wird diese Zeitspanne abgewartet, bevor der Treiber die Variablen als gestört kennzeichnet. Erreicht der Treiber die Agents in dieser Zeit, übergeht er die Kurzzeit-Störung ohne Meldung. Hinweis: Gilt nicht für Traps, wenn bei der Agentkonfiguration kein Initialwert gelesen wird. Diese Variablen bleiben dann ohne Status leer, auch wenn keine Verbindung zum Agent besteht.
Adressierung über Variablenname oder Kennung	`Aktiv`: Die Adressierung erfolgt über den Namen der Variablen oder über die Kennung. Die Eigenschaften **browse agen**t und **receive online traps** stehen dann nicht zur Verfügung. `Inaktiv`: Adressierung erfolgt über Offset in Zuordnungsdatei. Achtung: Die Art der Adressierung ist wichtig für den Variablenimport (auf Seite 45). Nachträgliche Änderungen der Adressierung von **Variablenname** oder **Kennung** auf **Offset** kann zu Kommunikationsproblemen führen.
Kennung für Adressierung verwenden	Kennung wird verwendet. Steht nur zur Verfügung, wenn "**Adressierung über Variablenname oder Kennung**" aktiviert wurde.
SNMP v2c zum Pollen verwenden	`Aktiv`: Für alle SNMP-Queries (GET, GETNEXT, SET) dieser Treiberinstanz wird **SNMP v2c** verwendet: ▸ für das Browsen der Agents in der Offline-Mib-Liste ▸ zur Runtime `Inaktiv`: **SNMP v1** wird verwendet. Hinweis: Die Treiberinstanz kann unabhängig von dieser Einstellung immer **v1** und **v2c** TRAPs empfangen. Tipp: Es kann eine einzige Treiberinstanz für v2c und v1 erstellt werden. Wenn Sie aber jedem SNMP Agent eine eigene Treiberinstanz geben (**v1** oder **v2c**), wirkt sich das Warten auf das Timeout eines ausgefallenen Agents nicht auf die Kommunikation mit den anderen Agents aus.

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.

6.2.3 SNMP Agents



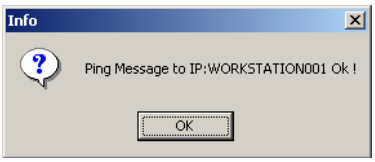
Parameter	Beschreibung
Agentfile	Die Datei in der die Konfigurationen der einzelnen SNMP-Agents gespeichert werden. Diese Datei entspricht dem SNMP Configurationfile, und kann daher in diesem Dialogfenster nicht verändert werden.
Netzadresse	Die zenon interne Netz Adresse der einzelnen Agents. Entspricht der Netzadresse in der Variablendefinition. Sie können diese Netzadresse z.B: zur logischen Gruppierung Ihrer Variablen verwenden.
Agentname	Der frei wählbare Name der einzelnen SNMP-Agents.
IP Adresse/Hostname	Die IP-Adresse der einzelnen Netzwerkteilnehmer/Agents. Es kann entweder die IP-Adresse oder der Computername (Voraussetzung dabei ist eine funktionierende Namensauflösung) verwendet werden. Wenn Traps verwendet werden, dürfen keinen Agents mit gleicher IP-Addresse konfiguriert sein.
IP Port	Portadresse Default: ▸ SNMP: 161 ▸ Traps: 162
Agent community default (public)	Die Zugriffskennung der Agents. Ist standardmäßig immer „public" (nur lesen). Ein Community-Name fungiert als Kennwort, das für einen oder mehrere SNMP-Hosts freigegeben ist. Akzeptierte Community-Namen werden lediglich zur Authentifizierung eingehender Nachrichten verwendet. Dies kann in den SNMP-Dienst-Einstellungen des Agents konfiguriert werden.
Trap Variablen	Einstellungen für Traps.
▸ **Keine Initialwerte vom Agent lesen**	Für Traps wird keinen Initialwert gelesen. Wert und Status der Variable bleiben so lange leer, bis ein Trap empfangen wird. **Empfehlung:** Wählen Sie diese Option, wenn der Agent Traps versendet, die nicht mittels GET gelesen werden können und damit nicht in der MIBliste enthalten sind.

Initialwerte vom Agent lesen	Startwerte für Traps werden vom Agent gelesen. Startwert für die Trap-Variablen wird vom Agent mittels "GET" gelesen. Sie erhalten den letzten gültigen Wert vom letzten Trap bzw. den Initialwert bei Runtimestart. Wurde der Startwert erfolgreich gelesen, wird die Trap-Variable nicht mehr zyklisch gelesen, sondern nur noch spontan. **Empfehlung:** Wählen Sie diese Option, wenn Traps vom Agent einmal mittels GET gelesen werden, um dann spontane Meldungen (Traps) zu erwarten.
Trap Variablen nach jeder Trap zurücksetzen	Für Traps wird keinen Initialwert gelesen. Wert und Status der Variable bleiben so lange leer, bis ein Trap empfangen wird. Wird ein Trap empfangen, wird der Wert an die Runtime gesendet, und anschließend sofort mit einem leeren String bzw. Wert 0 überschrieben. Trap-Wert kann durch Reaktionsmatrix erkannt werden und in AML und CEL geschrieben werden Ist bei aktiver Option der Wert der Trap-Variablen in der Runtime 0 oder ein leerer String mit Status **Spontan**, wurde bereits ein Trap empfangen. **Empfehlung:** Wählen Sie diese Option, um in der Runtime auf Traps zu reagieren, die immer den gleichen Wert haben, um einen CEL-Eintrag oder Alarm zu erzeugen.

OID Variablen übersetzen	Einstellungen für OID Variablen. Ermöglicht separate Konfiguration der Übersetzung für jeden Agenten. Hinweis: Nur verfügbar ab Version 6.50. Alle anderen Versionen werden automatisch mit dem Standard **Nur OID anzeigen** behandelt. **Voraussetzungen:** Die folgenden Eigenschaften definieren, wie die OID-Übersetzung auf eingehende Werte für Variablen vom SNMP-Treiber angewandt wird. Dazu müssen folgende Bedingungen erfüllt sein: ▸ Die Variable muss vom Treiberobjekttyp **SNMP Variable** oder **SNMP Trap** sein. ▸ Der eingehende Wert muss vom SNMP-Datentyp **OID** sein. ▸ Individuell installierte MIBS müssen sich in folgendem Pfad befinden: `%ProgramData%\COPA-DATA\zenon7.60\CommunicationProfiles\SNMP-MIBS` Eingehende Werte, die diese Bedingungen nicht erfüllen, werden unbehandelt weitergeleitet.
OID und Übersetzung anzeigen	`Aktiv`: OID und Übersetzung werden als Wert übermittelt. Die OID wird in einen beschreibenden Text übersetzt und der Wert wird im folgenden Format an die Runtime weitergeleitet: **[OID] ([Text])** Beispiel: **.1.3.6.1.6.3.1.1.5.3 (IF-MIB::linkDown)**. Wenn die Übersetzung nicht erfolgreich ist, wird nur die OID als Wert an die Runtime übermittelt.
Nur OID anzeigen	`Aktiv`: Nur OID wird als Wert übermittelt. Hinweis: Standard für alle Versionen vor zenon 6.50.
Nur Übersetzung anzeigen	`Aktiv`: Nur Übersetzung wird als Wert übermittelt. Die OID wird in einen beschreibenden Text übersetzt und nur der übersetzte Text wird an die Runtime weitergeleitet. Wenn die Übersetzung nicht erfolgreich ist, wird die OID als Wert an die Runtime übermittelt.

Neu	Neuen Agents definieren.
Löschen	Agent löschen.
Bearbeiten	Agent bearbeiten.
Speichern	Speichert die getroffenen Agent-Einstellungen.
Verwerfen	Verwirft die getroffenen Agent-Einstellungen.
Ping	Testet eine projektierten **Agent IP Adress**.

Ergebnis eines erfolgreichen Ping-Versuchs mit Namensauflösung:

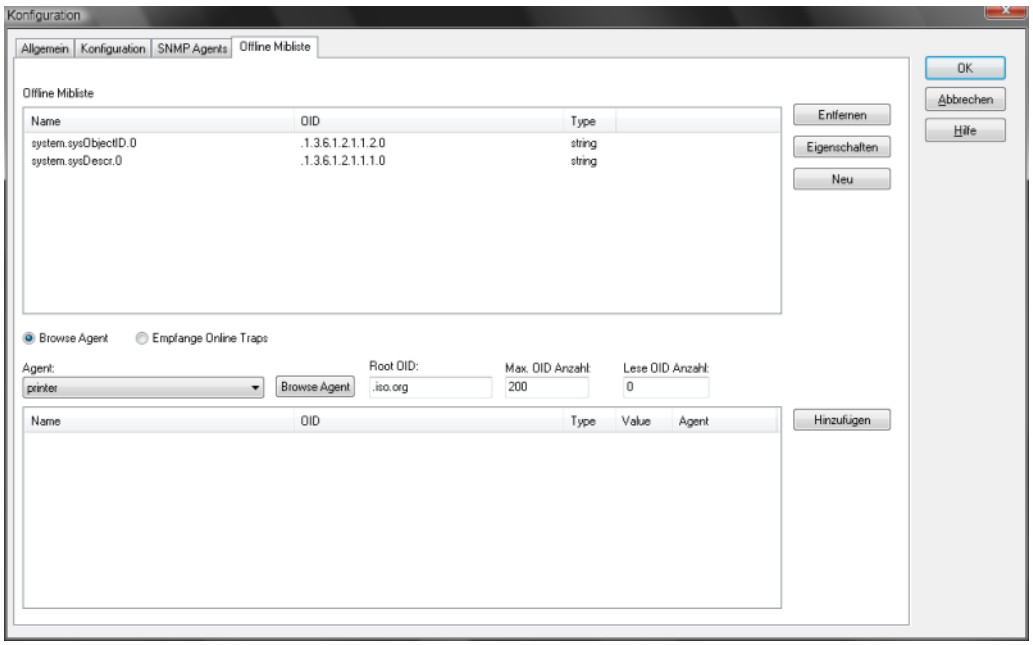


Erhalten Sie keine positive Rückmeldung auf einen Ping, konnte entweder der Name nicht richtig aufgelöst werden. In diesem Fall versuchen Sie einen Ping mit der IP-Adresse. Schlägt auch dieser fehl, konnte der Netzwerkpartner/Agent nicht erreicht werden.

Info

Maximale Anzahl der Verbindungen: 256 (0-255).

6.2.4 Offline MIB-Liste



Parameter	Beschreibung
Offline MIB-Liste	Beinhaltet alle Eigenschaften (=SNMP-Objekte) aller SNMP-Agents, die in der Variablendefinition verfügbar/auswählbar sein sollen. SNMP-Objekte, die nicht in diese Liste übernommen wurden, können später nicht als Variablen definiert werden.
Entfernen	Löscht ausgewählte OID aus der Liste.
Eigenschaften	Zeigt Eigenschaften des ausgewählten Eintrags an.
Neu	Ermöglicht das manuelle Erstellen eines neuen Eintrags (auf Seite 29).
Browse Agent	`Aktiv`: Agent wird nach OIDs durchsucht. Der ausgewählte Agent wird mittels **GETNEXT** so lange nach OIDS durchsucht, bis entweder die definierte maximale Anzahl an OIDs empfangen wurde oder der Agent auf GET mit **endofMIBview** antwortet als Kennzeichnung, dass es keine weitere OIDs mehr gibt. Hinweis: Für manche SNMP Agents kann konfiguriert werden, wie viele SNMP Pakete pro Sekunde beantwortet werden. Eine zu geringe Zahl kann sich auf das Auslesen der OIDs mittels **Browse Agent** auswirken.
Empfange Online Traps	`Aktiv`: Über den Windows Dienst **SNMP Trap** werden vom Treiber die SNMP Trap Meldungen von verschiedenen Agents empfangen und angezeigt. Dazu muss der Agent nicht zwingend konfiguriert sein, auch Traps von nicht konfigurierten Agents werden aufgelistet. Falls die Trapmeldung von einem bereits konfigurierten Agent kommt, wird der Name des Agents in der Spalte "**Agent**" angezeigt. Damit der Projektierungsrechner Traps empfängt, muss: ▸ die Firewall UDP Traffic auf Port `161` und `162` zulassen ▸ die Firewall den Prozess **snmptrap.exe** zulassen ▸ im Agent der Projektierungsrechner als Ziel für SNMP Traps definiert sein Hinweis: Erhalten Sie eine Fehlermeldung mit Bezug auf **0x2a** oder **0x64**, läuft möglicherweise der Windows Dienst **SNMP Trap** nicht.
Agent	Auswahl des Agents aus Dropdownliste, der durch den Treiber mit dem Befehl "GETNEXT" nach unterstützten Informationen durchsucht werden und dessen MIB-Objekte angezeigt werden sollen. Diese Liste beinhaltet alle Agents, die in der Registerkarte **SNMP Agents** (auf Seite 21) angelegt wurden.
Browse Agent	Durchsucht ausgewählten Agent.
Root OID	Hier ist die Definition "**.iso.org**" oder "**.1.3**" zu verwenden.
Max. OID Anzahl	Die Anzahl der OIDs die vom Browser ausgelesen werden sollen.
Lese OID Anzahl	Statusanzeige der bis zu diesem Zeitpunkt ausgelesenen OIDs.
Hinzufügen	Fügt ausgewählte OIDs in **Offline MIB-Liste** hinzu. Von hier aus können die OIDs zu verschiedenen Agents über Treiber-Online Import

	automatisch erstellt werden.

Achtung

Der SNMP-Treiber sucht auf Basis der Netzadresse erst den Agent und danach die zur OID gehörende Variable. Das bedeutet:

- Pro Agent darf nur eine Variable für eine OID definiert werden. Existieren für einen Agenten mehrere Variablen mit der gleichen OID, ist undefiniert, welche der Variablen bei Ankunft eines Traps aktualisiert wird.
- Mehrere Variablen mit der selben OID aber unterschiedlichen Agenten sind erlaubt.

Eine ähnliche Einschränkung gilt für die Agents. Für Trapvariablen wird über die Absender IP-Addresse der Agent zugeordnet. Mehreren Agents mit der gleiche IP-Addresse, oder mehrere Treiber mit Agents mit der gleiche IP-Addresse sind nicht unterstützt. Es ist undefiniert welcher Agent bzw. Treiber den Trapwert empfängt.

UNTERSCHIEDE MIB EDITOR UND ONLINE VARIBALEN-IMPORT

Werden Variablen Offline über den MIB Editor (auf Seite 26) angelegt oder über Online Variablenimport (auf Seite 45) angelegt, ergeben sich einige Unterschiede:

VARIABLEN UND AGENT

- Werden Traps für die **Offline MIB-Liste** empfangen, werden Traps von allen IP-Adressen angezeigt. Beim Hinzufügen der empfangenen Traps zur **Offline MIB-Liste** werden Informationen zum Agent nicht gespeichert.
- Werden Variablen vom Treiber importiert, muss darauf geachtet werden, die korrekten Variablen für den korrekten Agent als Trap zu erstellen.

KONFIGURIERTE UND UNKONFIGURIERTE AGENTS IM ONLINE-IMPORT

Wenn Traps über den Import-Dialog empfangen werden, werden SMNP-Traps von allen IP-Adressen angezeigt.

- Wenn ein Trap von einem Agent kommt, der bereits konfiguriert ist, wird der Agent beim Hinzufügen des Traps angepasst: die Net-Adresse wird korrekt gesetzt.
- Kommt der Trap von einem im Treiber nicht konfigurierten Agent (Source IP-Adresse wird in der Liste empfangener Traps angezeigt) und die Variable wird hinzugefügt, wird die Net Adresse 0 verwendet. Diese Net Adresse könnte allerdings bereits von einem anderen Agent verwendet werden und so zu Problemen führen.

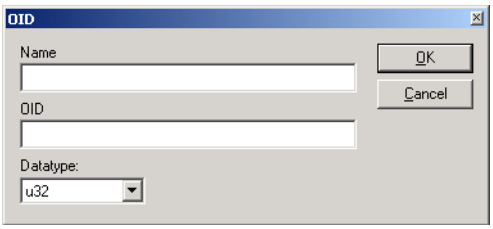
ADRESSIERUNG

Beim Browsen eines Agents oder beim Empfangen von Traps im Treiber Online-Import, werden Variablen nicht zur **Offline MIB-Liste** hinzugefügt und empfangen keinen Offset.

Wird die Art der Adressierung in der Treiberkonfiguration (auf Seite 19) von **Variablenname** oder **Kennung** auf **Offset** geändert, werden diese Variablen nicht mehr kommunizieren. Wählen Sie daher die Art der Kommunikation bevor Sie Variablen erstellen.

OID manuell anlegen

Wenn zum Zeitpunkt des Projektierens keine Verbindung zum gewünschten Netzwerkteilnehmer besteht, kann die OID auch ohne Browser Agent angelegt werden.



Parameter	Beschreibung
Name	Hier kann ein frei wählbarer Name verwendet werden.
OID	Die genaue Bezeichnung der OID. Bitte beachten Sie, dass hier die vollständige und korrekte Bezeichnung eingegeben wird, da der zenon SNMP-Treiber die Daten sonst nicht auslesen kann.
Datatype	Integer oder String, abhängig vom anzusprechenden SNMP-Datentyps.

OID-Variablen übersetzen

Die OID-Übersetzung steht zur Verfügung für Variablen von den Treiberobjekttypen:

- SNMP Trap
- SNMP Variable

Die Funktion zur OID-Übersetzung ist nur bei Treibern für zenon 6.50 und höher verfügbar.Die Liste aller Inhalte einer Trap wird in jeder Treiberversion unterstützt

Info

Hinweis zum Copyright

Die Funktion zur OID-Übersetzung wurde aus der Net-SNMP Open Source Library in der Version 5.6.1.1 extrahiert.

Copyright 1989, 1991, 1992 by Carnegie Mellon University. Derivative Work - 1996, 1998-2000.

Copyright 1996, 1998-2000 The Regents of the University of California. All Rights Reserved.

KONFIGURATION

Überblick über notwendigen Konfigurationen und der daraus resultierenden Ergebnisse:

1. Treiberkonfiguration:
   a) Ein Treiber vom Typ **SNMP Treiber** wird im Editor angelegt.
   b) Am Treiber werden die benötigten Agenten konfiguriert (auf Seite 21).
   c) Bei den Agenten wird die gewünschte OID-Übersetzung eingestellt.
   d) In der Offline MIB-Liste (auf Seite 26) werden die Agenten gebrowst und die gewünschten OIDs zur Offline MIB-Liste hinzugefügt.
   e) Danach wird auf **Empfange Online Traps** umgeschaltet, um die Liste aller Inhalte der Traps in die Offline MIB-Liste einzufügen. Nachdem von den Agenten Traps empfangen wurden, werden die notwendigen OIDs aus den Traps zur Liste hinzugefügt. Die Treiberkonfiguration ist damit abgeschlossen.
2. Projektierung im Editor:
   a) Im Editor werden die Variablen vom Treiber importiert, wobei für jeden Agenten die benötigten Variablen aus der Offline MIB-Liste eingefügt werden.
   b) Die Variablen werden mit Elementen auf Bildern verknüpft, damit sie zur Laufzeit betrachtet werden können.
   c) Die Runtime-Dateien werden erzeugt und die Runtime wird gestartet.
3. Verhalten zur Runtime:
   a) Die Runtime startet den SNMP Treiber.
   b) Die SNMP-Treiberinstanz initialisiert die OID-Übersetzung.
   c) Nachdem die Variablen am Treiber angemeldet wurden, werden die ersten Werte gelesen: Für alle Variablen der Treiberobjekttypen **Pingstatus**, **SNMP Counter**, **SNMP Variable** und **SNMP Traps** wenn **Initialwerte für Traps lesen** aktiv ist.

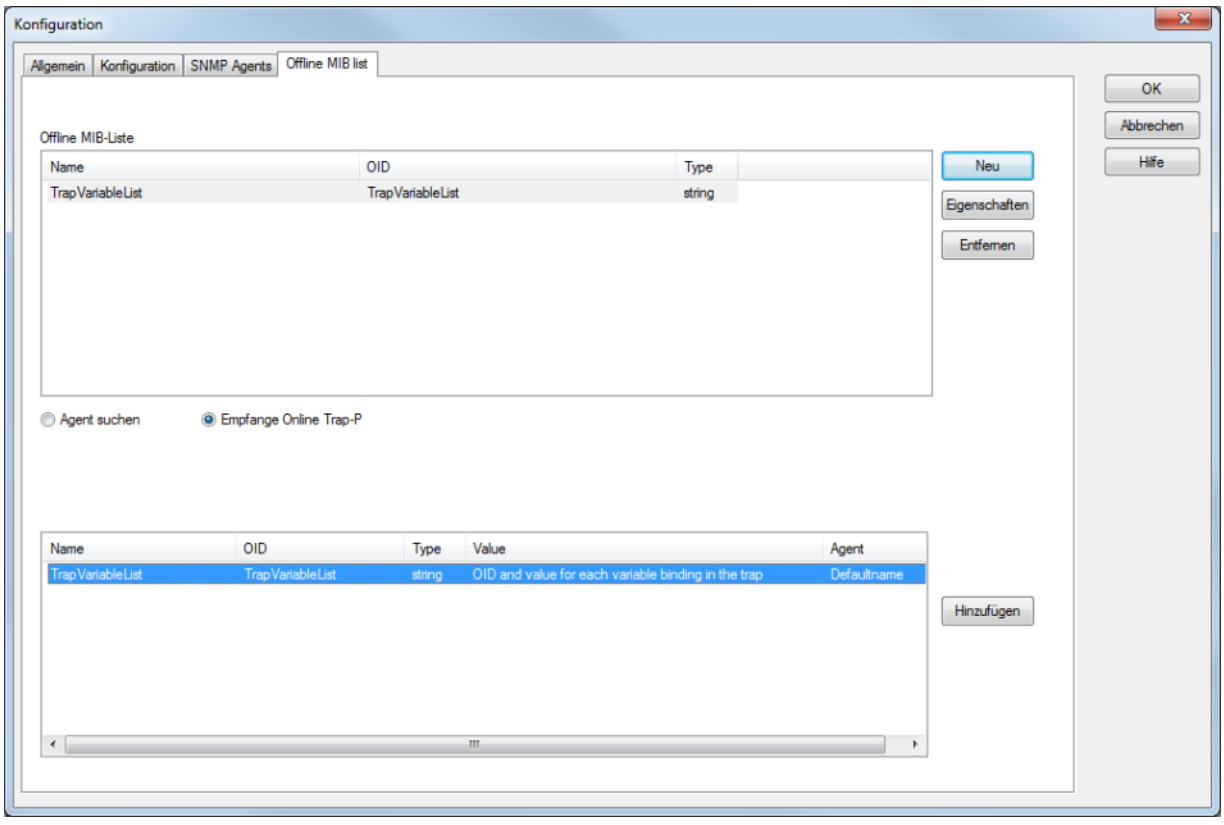
d) Bei Variablen der Treiberobjekttypen **SNMP Variable** und **SNMP Trap** werden empfangene Werte vom SNMP-Datentyp **OID** gemäß der konfigurierten OID-Übersetzung für den jeweiligen Agenten behandelt.

e) Bei jedem weiteren Polling und jeder empfangenen Trap kommt die OID-Übersetzung für Werte vom SNMP-Datentyp **OID** gemäß der jeweiligen Agenten-Konfiguration zum Einsatz. Bei Traps wird darüber hinaus noch die Liste aller Inhalte erstellt.

Stringvariable mit allen Trap-Inhalten

Die Liste aller Inhalte einer Trap wird im Dialog der Offline MIB-Liste angezeigt, wenn dieser auf Trapempfang eingestellt ist.

Bei einem Klick auf **Empfange Online Traps** wird ein Beispieleintrag für jeden konfigurierten Agenten erzeugt und in der Liste empfangener Traps angezeigt. Für jede empfangene Trap wird eine solche Liste aller Inhalte angelegt und als letzter empfangener Eintrag für diese Trap eingefügt.



Im Screenshot werden empfangene Variablen vom SNMP-Datentyp OID immer als **OID und Übersetzung** angezeigt. Als Name wird der aus der OID übersetzte beschreibende Text angezeigt.

Das Objekt kann in der **Offline-MIB-Liste** in der Liste für **Empfange Online Trap** eingefügt werden. Beim Importieren von Variablen vom Treiber aus der **Offline-MIB-Liste** kann für jeden Agenten eine

Stringvariable mit allen Trapinhalten angelegt werden. Diese muss am SNMP Treiber mit folgenden Parametern angelegt werden:

- Treiberobjekttyp: **SNMP Trap**
- Netzadresse: Index des Agenten, den sie betrifft
- Kennung: **TrapVariableList**
- Offset

Tipp: Fügen Sie den Beispieleintrag des Agenten in die Offline MIB-Liste ein und importieren Sie dann die Variable vom Treiber.

FORMAT DER STRINGVARIABLEN

Der String ist im gleichen Format wie eine INI-Datei formatiert:

Der erste Knoten **[DEFAULT]** enthält einen Eintrag **COUNT**, der die Anzahl der Variable-Bindings in einer Trap angibt. Danach folgt ein Knoten **ENTRY_[Nr]** für jedes Variable-Binding mit den Inhalten **OID**, **OID_TRANSLATED**, **TYPE** und **VALUE**.

SCHEMA

Parameter	Beschreibung
[DEFAULT]	
COUNT=x	Anzahl der Einträge.
[ENTRY_x]	Index
OID=x	OID im numerischen Format.
OID_TRANSLATED=x	Beschreibender Text zur OID, wenn die OID-Übersetzung verfügbar ist .
[TYPE]=x	zenon Datentyp des Werts
[VALUE]=x	Empfangener Wert. Empfangene OID-Werte werden entsprechend der Verfügbarkeit und Konfiguration der OID-Übersetzung hier eingetragen.

BEISPIEL

[DEFAULT]

COUNT=7

[ENTRY_000]

OID=.1.3.6.1.2.1.1.3.0

OID_TRANSLATED=DISMAN-EVENT-MIB::sysUpTimeInstance

TYPE=u32

VALUE=59769454

[ENTRY_001]

OID=.1.3.6.1.6.3.1.1.4.1.0

OID_TRANSLATED=SNMPv2-MIB::snmpTrapOID.0

TYPE=string

VALUE=.1.3.6.1.6.3.1.1.5.3 (IF-MIB::linkDown)

[ENTRY_002]

OID=.1.3.6.1.2.1.2.2.1.1.6

OID_TRANSLATED=IF-MIB::ifIndex.6

TYPE=i32

VALUE=6

[ENTRY_003]

OID=.1.3.6.1.2.1.2.2.1.7.6

OID_TRANSLATED=IF-MIB::ifAdminStatus.6

TYPE=i32

VALUE=1

[ENTRY_004]

OID=.1.3.6.1.2.1.2.2.1.8.6

OID_TRANSLATED=IF-MIB::ifOperStatus.6

TYPE=i32

VALUE=2

[ENTRY_005]

OID=.1.3.6.1.3.1057.1

OID_TRANSLATED=SNMPv2-SMI::experimental.1057.1

TYPE=string

VALUE=192.168.0.120

[ENTRY_006]

OID=.1.3.6.1.6.3.1.1.4.3.0

OID_TRANSLATED=SNMPv2-MIB::snmpTrapEnterprise.0

TYPE=string

VALUE=.1.3.6.1.6.3.1.1.5 (SNMPv2-MIB::snmpTraps)

6.2.5 Beispielkonfiguration

Um übersetzte OID Variablen und Listen der Inhalte von Traps von einem Agenten zu empfangen und darzustellen:

1. konfigurieren Sie den SNMP Agent
2. legen Sie einen SNMP Treiber an
3. konfigurieren Sie den SNMP Agenten im Treiber
4. fügen Sie die benötigten OIDs zur Offline MIB-Liste hinzu
5. importieren Sie Variablen vom Treiber
6. stellen Sie die Variablen in einem Bild dar

SNMP AGENT KONFIGURIEREN

Der Agent muss der Standard-SNMP-Community **public** zumindest Leserechte einräumen.

Als Ziel für Traps muss die IP-Adresse des späteren Runtime-Rechners eingetragen werden. Wenn Engineering Station und Runtime-Rechner unterschiedliche Rechner sind, kann die IP-Adresse der Engineering Station hier zusätzlich eingetragen werden. Sofern der Agent das unterstützt, kann man so zur Designzeit auch Traps vom Agenten bekommen. Unterstützt der Agent keine multiplen Trapziele, kann hier die IP-Adresse der Engineering Station als einziges Trap-Ziel eingetragen werden. Dieser Eintrag muss aber vor dem Starten der Runtime durch die IP-Adresse des Runtimerechners ersetzt werden, damit dieser die Traps bekommt.

SNMP TREIBER ANLEGEN

Um einen neuen Treiber anzulegen:

1. öffnen Sie den Dialog zur Erstellung eines neuen Treibes über den Befehl **Treiber neu** :
   - im Kontextmenü im Projektknoten **Treiber**
     oder
   - Kontextmenü in der Treiberliste
     oder
   - Symbolleiste in der Treiberansicht
2. wählen Sie im Dialog den **SNMP** Treiber

SNMP AGENTEN IM TREIBER KONFIGURIEREN

In der Treiberkonfiguration legen Sie in der Registerkarte SNMP Agents (auf Seite 21) die Einstellungen für die Agenten fest. Beachten Sie dabei:

- Passen Sie den Port an: der Port für SNMP-Abfragen ist 161, für Traps 162.
- Die Standardcommunity **public** muss zulässig sein.
- Als IP-Adresse muss die IP-Adresse des SNMP Agenten eingetragen werden.
- Über die Optionen im Konfigurationsbereich **Trap Variablen** steuern Sie das Lesen von Initialwerten für Trapvariablen und ob diese als Wischer behandelt werden.
- Über die Optionen im Konfigurationsbereich OID Variablen übersetzen steuern die OID-Übersetzung. Hier wird eine der übersetzenden Varianten (alles außer „Nur OID anzeigen") ausgewählt.

BENÖTIGTE OIDS ZUR OFFLINE MIB-LISTE HINZUFÜGEN

Da in diesem Beispiel nur Traps von Interesse sind, aktivieren Sie in der Registerkarte Offline MIB-Liste (auf Seite 26) die Option **Empfange Online Traps:**

- Der automatisch für den Agenten erzeugte Eintrag **TrapVariableList** wird zur Offline MIB-Liste hinzugefügt.
- Nach dem Empfang einer Trap werden die benötigten OIDs zur Liste hinzugefügt.

Schließen Sie den Treiber-Konfigurationsdialog.

VARIABLEN VOM TREIBER IMPORTIEREN

Öffnen Sie das Kontextmenü des SNMP Treibers in der Treiberliste und wählen Sie den Eintrag **Variablen vom Treiber importieren**.

Im sich öffnenden Dialog (auf Seite 45):

- wählen Sie die die Einträge in der Offline MIB-Liste aus
- stellen Sie den zu verwendenden Agenten ein
- importieren Sie die Variablen mit Klick auf OK in die Variablenliste

VARIABLEN IN EINEM BILD DARSTELLEN

Die gewählten Variablen stellen Sie abhängig von ihrem Datentyp in einem dynamischen Zahlenwert oder in einem dynamischen Text dar. Für die Liste aller Inhalte einer Trap sollte ein großes Element projektiert werden, da diese sehr lang werden kann.

Optional kann noch eine Pingstatus-Variable für den Agenten angelegt und mit einem Schalter verknüpft werden, um die Verfügbarkeit des Agenten anzuzeigen.

VERHALTEN ZUR RUNTIME

Nachdem die Runtime gestartet wurde, startet diese den SNMP Treiber. Ab der Treiberinstanzerzeugung durch die Runtime werden folgende Schritte durchgeführt:

1. Der Treiber durchläuft seine Initialisierung.
   Dabei werden auch die OID-Übersetzung initialisiert und der Index für die MIBs angelegt.
2. Die Runtime meldet die benötigten Variablen beim Treiber an.
   Auf Basis der Netzadresse entscheidet der Treiber, welche Variable welchem Agenten zugeordnet wird und liest die Konfiguration des Agenten aus dem Treiberkonfigurationsfile.
3. Für pollende Variablen werden die ersten Werte gelesen und der Polling-Zyklus wird gestartet. Empfangene Werte vom SNMP-Datentyp **OID** werden entsprechend der Agentenkonfiguration übersetzt.
4. Traps erfassen.
   Bei eingehenden Traps werden zuerst die einzelnen Variable Bindings abgearbeitet - OIDs übersetzen, Werte an Runtime schicken - und in einer Liste erfasst. Auf Basis dieser Liste wird der String für die Variable mit der Liste aller Inhalte einer Trap generiert und zur Runtime geschickt.
5. Beim Beenden der Runtime wird dem Treiber das Kommando zum Beenden übermittelt.
6. Der Treiber durchläuft seine Deinitialisierung und löscht dabei den zuvor angelegten Index.

7. Variablen anlegen

So werden Variablen im zenon Editor angelegt:

7.1 Variablen im Editor anlegen

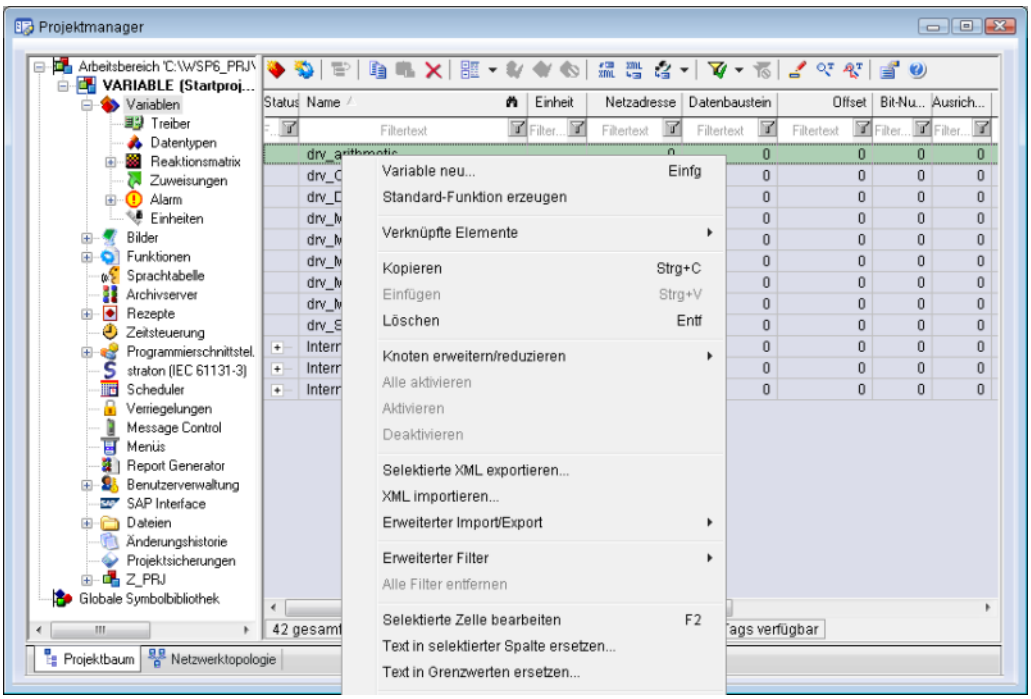
Variablen können angelegt werden:

- 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:

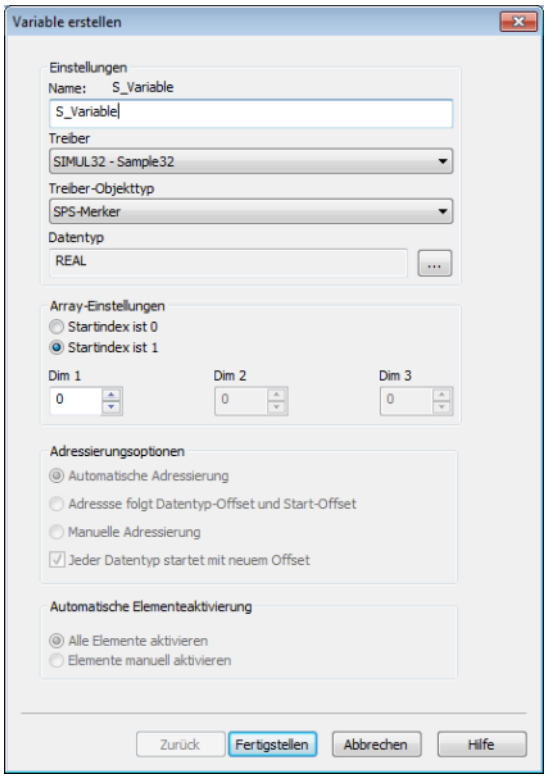
1. Wählen Sie im Knoten **Variablen** im Kontextmenü den Befehl **Variable neu**.



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 **Signalbereic**h nicht vom Datentyp abgeleitet wurde. In diesem Fall muss der **Messbereich** manuell angepasst werden.

7.2 Adressierung

Eigenschaft	Beschreibung
Name	Frei vergebbarer Name. Achtung: Je zenon Projekt muss der Name eindeutig sein.
Kennung	Frei vergebbare Kennung. Z. B. für **Betriebsmittelkennung**, Kommentar usw. Wenn über die Variablenkennung adressiert wird muss hier die OID eingetragen werden.
Netzadresse	Netzadresse der Variablen. Diese Adresse bezieht sich auf die Netzadresse der Verbindungsprojektierung im Treiber. Damit wird ausgewählt auf welcher Steuerung sich die Variable befindet.
Datenbaustein	Wird für diesen Treiber nicht verwendet.
Offset	Offset der Variablen. Entspricht der Speicheradresse der Variablen in der Steuerung. Einstellbar von 0 bis 4294967295.
Ausrichtung	Wird für diesen Treiber nicht verwendet.
Bitnummer	Nummer des Bits innerhalb des eingestellten Offsets. Mögliche Eingabe: 0 bis 65535.
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.
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	Einstellen der Prioritätsklasse. Damit wird die Variable der Prioritätsklasse zugeordnet, so wie sie im Treiberdialog in der Registerkarte **Allgemein** konfiguriert wurde. Die Prioritätsklassen werden nur dann verwendet, wenn die **Update Zeit Global** deaktiviert ist. Bei aktivierter Option **Update Zeit Global** und Verwendung der Prioritätsklassen erfolgt ein Fehlereintrag in die Log-Datei des Systems. Der Treiber verwendet die höchstmögliche Priorität.

Beim Importieren von Variablen vom Treiber aus der **Offline-MIB-Liste** kann für jeden Agenten eine Stringvariable mit allen Trapinhalten angelegt werden. Diese muss am SNMP Treiber angelegt werden. Details siehe Abschnitt Stringvariable mit allen Trap-Inhalten (auf Seite 31).

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 Treiberobjekttypen stehen in diesem Treiber zur Verfügung:

OBJEKTE VON PROZESSVARIABLEN IN ZENON

Treiberobjekttyp	Kanaltyp	Lesen	Schreiben	Unterstützte Datentypen	Kommentar
Pingstatus	64	X	X	BOOL	
SNMP-Traps	67	X	X	DINT, UDINT, STRING	Es werden V1 und V2 Traps unterstützt.
SNMP-Variablen	65	X	X	DINT, UDINT, STRING	
SNMP-Zähler	66	X	X	DINT, UDINT	
Kommunikationsdetails	35	X	X	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variablen zur statistischen Auswertung der Kommunikation. Weitere Informationen dazu finden Sie im Kapitel Kommunikationsdetails (Treibervariablen) (auf Seite 56).

Objekt	Lesen	Schreiben	Kommentar
Konfiguration			Ruft das Treiberkonfigurationsmenü auf
Pingstatus	X		Bit.
+i/u32Bit	X		
String	X		

Legende:

X => wird unterstützt

-- => wird nicht unterstützt

> **Achtung**
>
> Der Treiber setzt den Wert für TrapVariableList nicht automatisch zurück. Die Werte anderer Variablen werden automatisch zurückgesetzt.

FUNKTIONSWEISE SNMP-ZÄHLER

Ein Zähler berechnet den Durchschnitt der Wertänderung über Zeit (in Sekunden) zwischen zwei Lesezyklen und sendet diesen an die Runtime. Der Wert aus der OID wird allerdings nicht an die Runtime gesendet. Ist der neue Wert aus der OID kleiner als den vorigen Wert (overflow), wird dieser Wert ignoriert, und solange kein neuer Wert für den Zähler an die Runtime gesendet bis ein neuer Wert berechnet werden kann.

Über diesen Zähler kann z. B. der Netzwerktraffic gemessen werden: Der Agent muss dafür z. B. die Anzahl der gelesenen Bytes ständig hochzählen. Mit einem Lesezyklus von 1000 ms erhält man die Anzahl der Bytes pro Sekunde.

BEISPIEL

- Treiber-Updatezyklus: `30 s`
- Wert 1: `2500`
- Wert 2: `7500`
- Ergebnis für Zählervariable: `166`

  Denn: 5000 Wertdifferenz über 30 Sekunden = `166`

7.3.2 Zuordnung der Datentypen

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

Steuerung	zenon	Datenart
	BOOL	8

	USINT	9
	SINT	10
	UINT	2
	INT	1
	UDINT	4
	DINT	3
	ULINT	27
	LINT	26
	REAL	5
	LREAL	6
	STRING	12
	WSTRING	21
	DATE	18
	TIME	17
	DATE_AND_TIME	20
	TOD (Time of Day)	19

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

7.4 Variablen anlegen durch Import

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

Info

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

7.4.1 XML Import

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 Export-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 Variablen vom Treiber importieren

Um Variablen vom Treiber zu importieren:

1. wählen Sie im Kontextmenüs des Treibers den Befehl **Variablen vom Treiber importierten**
2. der Konfigurationsdialog wird geöffnet

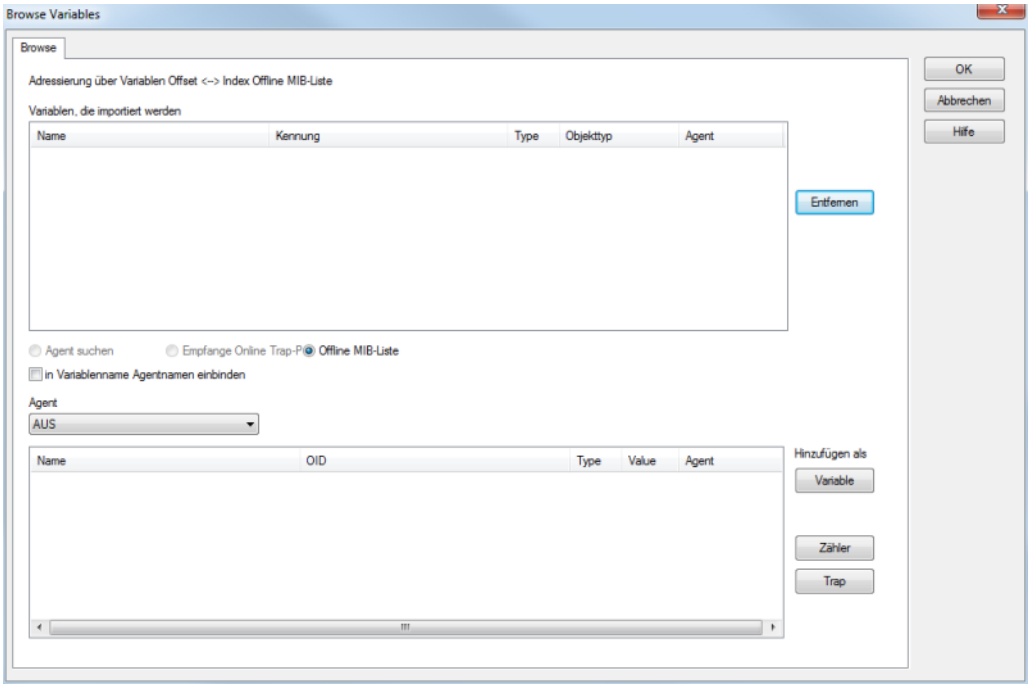
   Die Optionen des Dialogs hängt von den Einstellungen in der Konfiguration (auf Seite 19) ab

   a) Adressierung über Offset

   b) Adressierung über Variablenname oder Kennung

> **Achtung**
>
> *Der SNMP-Treiber sucht auf Basis der Netzadresse erst den Agent und danach die zur OID gehörende Variable. Das bedeutet:*
>
> - Pro Agent darf nur eine Variable für eine OID definiert werden. Existieren für einen Agenten mehrere Variablen mit der gleichen OID, ist undefiniert, welche der Variablen bei Ankunft eines Traps aktualisiert wird.
> - Mehrere Variablen mit der selben OID aber unterschiedlichen Agenten sind erlaubt.
>
> *Eine ähnliche Einschränkung gilt für die Agents. Für Trapvariablen wird über die Absender IP-Addresse der Agent zugeordnet. Mehreren Agents mit der gleiche IP-Addresse, oder mehrere Treiber mit Agents mit der gleiche IP-Addresse sind nicht unterstützt. Es ist undefiniert welcher Agent bzw. Treiber den Trapwert empfängt.*

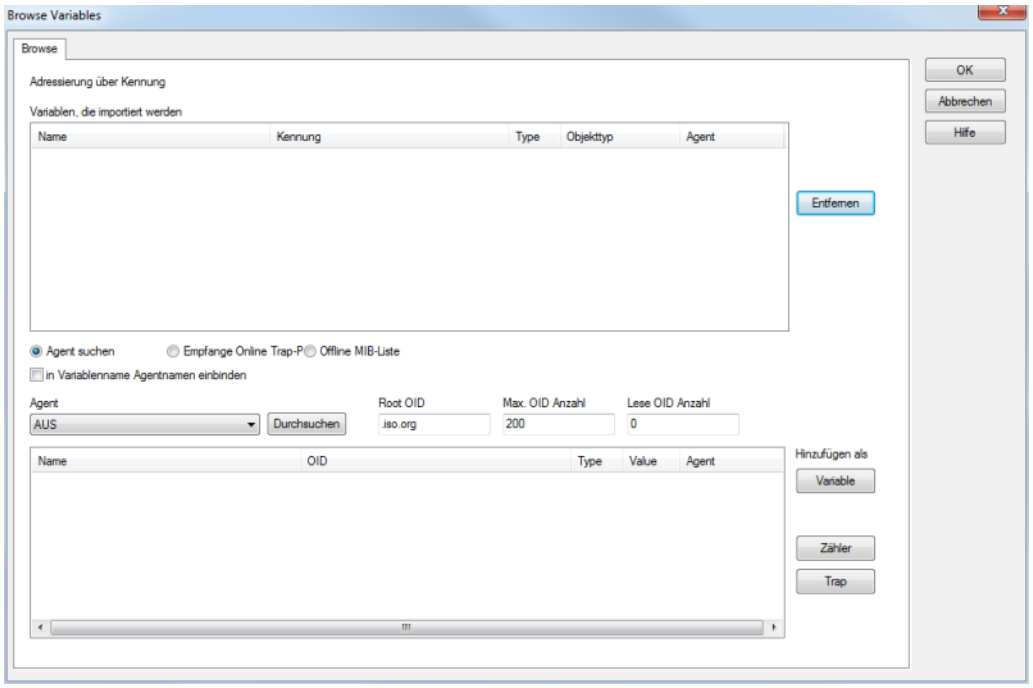
ADRESSIERUNG ÜBER OFFSET



Parameter	Beschreibung
Adressierung über Variablen Offset <--> Index Offline MIB-Liste	Hinweis auf die unter Konfiguration (auf Seite 19) definierte Adressierung.
zenon Variable zum Erzeugen	Liste der durch Import anzulegenden Variablen in zenon.
Entfernen	Entfernt Einträge aus der Liste.
Browse Agent	Kann nicht genutzt werden bei Addressierung über Offset.
Empfange Online Traps	Kann nicht genutzt werden bei Addressierung über Offset.
Offline MIB-Liste	Aktiv: Variablen werden aus der Offline MIBliste (auf Seite 26) importiert. Einträge der Liste werden im Listenfeld unterhalb angezeigt.
In Variablenname Agentnamen einbinden.	`Aktiv:` Dem Variablennamen wird in der Importliste der Name des Agent vorangestellt, beide Einträge werden durch einen Unterstrich getrennt: **Agent_Variablenname**.
Agent	Dropdownliste der Agents.
`Liste`	Liste der Variablen, die in der Offline MIB-Liste enthalten sind.
Hinzufügen als:	Auswahl über Schaltflächen, wie zu importierende Variablen zur Liste "zenon Variablen zum Erzeugen" hinzugefügt werden sollen:
▸ **Variable**	Hinzufügen als Variable.
▸ **Zähler**	Hinzufügen als Zähler.

Trap	Hinzufügen als Trap.
OK	Übernimmt alle Einstellungen und beendet den Dialog. Die Variablen in der Liste "zenon Variablen zum Erzeugen" werden importiert.
Abbrechen	Alle Einstellungen werden verworfen und der Dialog geschlossen.
Hilfe	Öffnet die Online-Hilfe.

ADRESSIERUNG ÜBER VARIABLENNAME ODER KENNUNG



Parameter	Beschreibung
Adressierung über Variablen Offset <--> Index Offline MIBliste	Hinweis auf die unter Konfiguration (auf Seite 19) definierte Adressierung.
zenon Variable zum Erzeugen	Liste der durch Import anzulegenden Variablen in zenon.
Entfernen	Entfernt Einträge aus der Liste.
Browse Agent	`Aktiv`: Agent wird nach OIDs durchsucht. Der ausgewählte Agent wird mittels **GETNEXT** so lange nach OIDS durchsucht, bis entweder die definierte maximale Anzahl an OIDs empfangen wurde oder der Agent auf GET mit **endofMIBview** antwortet als Kennzeichnung, dass es keine weitere OIDs mehr gibt. Hinweis: Für manche SNMP Agents kann konfiguriert werden, wie viele SNMP Pakete pro Sekunde beantwortet werden. Eine zu geringe Zahl kann sich auf das Auslesen der OIDs mittels **Browse Agent** auswirken.
Empfange Online Traps	`Aktiv`: Über den Windows Dienst **SNMP Trap** werden vom Treiber die SNMP Trap Meldungen von verschiedenen Agents empfangen und angezeigt. Dazu muss der Agent nicht zwingend konfiguriert sein, auch Traps von nicht konfigurierten Agents werden aufgelistet. Falls die Trapmeldung von einem bereits konfigurierten Agent kommt, wird der Name des Agents in der Spalte "**Agent**" angezeigt. Damit der Projektierungsrechner Traps empfängt, muss: ▸ die Firewall UDP Traffic auf Port `161` und `162` zulassen ▸ die Firewall den Prozess **snmptrap.exe** zulassen ▸ im Agent der Projektierungsrechner als Ziel für SNMP Traps definiert sein Hinweis: Erhalten Sie eine Fehlermeldung mit Bezug auf **0x2a** oder **0x64**, läuft möglicherweise der Windows Dienst **SNMP Trap** nicht.
Agent	Auswahl des Agents aus Dropdownliste, der durch den Treiber mit dem Befehl "GETNEXT" nach unterstützten Informationen durchsucht werden und dessen MIB-Objekte angezeigt werden sollen. Diese Liste beinhaltet alle Agents, die in der Registerkarte **SNMP Agents** (auf Seite 21) angelegt wurden.
Browse Agent	Durchsucht ausgewählten Agent.
Root OID	Hier ist die Definition "**.iso.org**" oder "**.1.3**" zu verwenden.
Max. OID Anzahl	Die Anzahl der OIDs die vom Browser ausgelesen werden sollen.
Lese OID Anzahl	Statusanzeige der bis zu diesem Zeitpunkt ausgelesenen OIDs.
`Liste`	Liste der Variablen, die in der Offline MIB-Liste enthalten sind.
Hinzufügen als:	Auswahl über Schaltflächen, wie zu importierende Variablen zur Liste "zenon Variablen zum Erzeugen" hinzugefügt werden sollen:
▸ **Variable**	Hinzufügen als Variable.

▸ **Zähler**	Hinzufügen als Zähler.
▸ **Trap**	Hinzufügen als Trap.
OK	Übernimmt alle Einstellungen und beendet den Dialog. Die Variablen in der Liste "zenon Variablen zum Erzeugen" werden importiert.
Abbrechen	Alle Einstellungen werden verworfen und der Dialog geschlossen.
Hilfe	Öffnet die Online-Hilfe.

UNTERSCHIEDE MIB EDITOR UND ONLINE VARIBALEN-IMPORT

Werden Variablen Offline über den MIB Editor (auf Seite 26) angelegt oder über Online Variablenimport (auf Seite 45) angelegt, ergeben sich einige Unterschiede:

VARIABLEN UND AGENT

- Werden Traps für die **Offline MIB-Liste** empfangen, werden Traps von allen IP-Adressen angezeigt. Beim Hinzufügen der empfangenen Traps zur **Offline MIB-Liste** werden Informationen zum Agent nicht gespeichert.
- Werden Variablen vom Treiber importiert, muss darauf geachtet werden, die korrekten Variablen für den korrekten Agent als Trap zu erstellen.

KONFIGURIERTE UND UNKONFIGURIERTE AGENTS IM ONLINE-IMPORT

Wenn Traps über den Import-Dialog empfangen werden, werden SMNP-Traps von allen IP-Adressen angezeigt.

- Wenn ein Trap von einem Agent kommt, der bereits konfiguriert ist, wird der Agent beim Hinzufügen des Traps angepasst: die Net-Adresse wird korrekt gesetzt.
- Kommt der Trap von einem im Treiber nicht konfigurierten Agent (Source IP-Adresse wird in der Liste empfangener Traps angezeigt) und die Variable wird hinzugefügt, wird die Net Adresse 0 verwendet. Diese Net Adresse könnte allerdings bereits von einem anderen Agent verwendet werden und so zu Problemen führen.

ADRESSIERUNG

Beim Browsen eines Agents oder beim Empfangen von Traps im Treiber Online-Import, werden Variablen nicht zur **Offline MIB-Liste** hinzugefügt und empfangen keinen Offset.

Wird die Art der Adressierung in der Treiberkonfiguration (auf Seite 19) von **Variablenname** oder **Kennung** auf **Offset** geändert, werden diese Variablen nicht mehr kommunizieren. Wählen Sie daher die Art der Kommunikation bevor Sie Variablen erstellen.

7.4.3 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
- Variablen 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. Treiberspezifische Funktionen

Dieser Treiber unterstützt folgende Funktionen:

GET, SET, GETNEXT

Mit den Funktionen GET, SET und GETNEXT werden Daten aus SNMP fähigen Geräten ausgelesen.

PING-STATUS

Über den Ping-Status wird festgestellt, ob das Endgerät per ICMP-Protokoll erreichbar ist.

Dazu wird das Statusbit INVALID (main.chm::/24148.htm) über das Combi-Element oder Reaktionsmatrizen abgefragt.

Wert `1`: Gerät ist erreichbar.

Wert mit Status INVALID: Kommunikation ist gestört.

SNMP TRAPS

Der Treiber unterstützt das Empfangen von SNMP Traps mit SNMPV1 und SNMPv2c header.

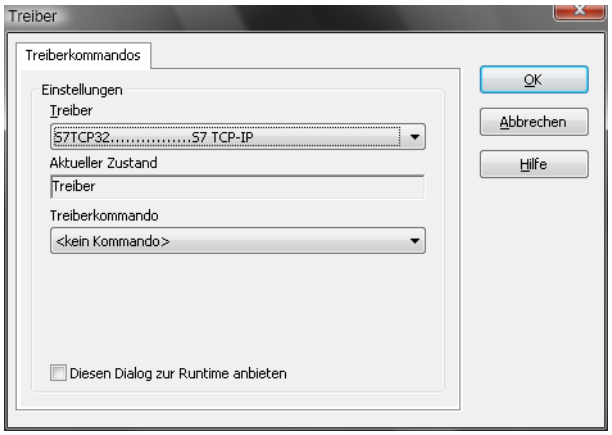
INFORMS (Traps mit Empfangsbestätigung) werden nicht unterstützt.

9. Treiberkommandos

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

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

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



Parameter	Beschreibung
Treiber	Dropdownliste mit allen im Projekt geladenen Treibern.
Aktueller Zustand	Fixer Eintrag, in aktuellen Versionen ohne Funktion.
Treiberkommando	Dropdownliste zur Auswahl des Kommandos.
▸ `Treiber starten (Online-Modus)`	Treiber wird neu initialisiert und gestartet.
▸ `Treiber stoppen (Offline-Modus)`	Treiber wird angehalten, es werden keine neuen Daten angenommen. **Hinweis:** Ist der Treiber im Offline-Modus, erhalten alle Variablen, die für diesem Treiber angelegt wurden, den Status `Abgeschaltet` (`OFF`; Bit 20).
▸ `Treiber in Simulationsmodus`	Treiber wird in den Simulationsmodus gesetzt. Die Werte aller Variablen des Treibers werden vom Treiber simuliert. Es werden keine Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
▸ `Treiber in Hardwaremodus`	Treiber wird in den Hardwaremodus gesetzt. Für die Variablen des Treibers werden die Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
▸ `Treiberspezifisches Kommando`	Eingabe treiberspezifischer Kommandos. Öffnet Eingabefeld für die Eingabe eines Kommandos.
▸ `Treiber Sollwertsetzen aktivieren`	Sollwert setzen auf Treiber ist erlaubt.
▸ `Treiber Sollwertsetzen deaktivieren`	Sollwert setzen auf Treiber wird verhindert.
▸ `Verbindung mit Modem aufbauen`	Verbindung aufbauen (für Modem-Treiber). Öffnet Eingabefelder für Hardware-Adresse und Eingabe der zu wählenden Nummer.
▸ `Verbindung mit Modem trennen`	Verbindung beenden (für Modem-Treiber).
Diesen Dialog zur Runtime anbieten	Dialog wird zur Runtime für Änderungen angeboten.

TREIBERKOMMANDOS IM NETZWERK

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

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

10. Fehleranalyse

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

10.1 Analysetool

Alle zenon Module wie z. B. Editor, Runtime, Treiber, usw. schreiben Meldungen in eine gemeinsame LOG-Datei. Um sie korrekt und übersichtlich anzuzeigen, benutzen Sie das Programm Diagnosis Viewer (main.chm::/12464.htm), das mit zenon mitinstalliert wird. Sie finden es unter *Start/Alle Programme/zenon/Tools 7.60 -> 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).

10.2 Fehlerbehandlung

FEHLERMELDUNGEN

Fehlertext	Beschreibung
`Error on SnmpMgrRequest GET oid=33.1.3.1.0, specified. Busadress=0 Error=0x28 Error=0x`	Fehlercode als hexadezimal Zahl à 0x28 = 40 Dec

GET Error: errorStatus=2, errorIndex=1 oid=.1.3.6.1.4.1.171.20.1.2.1.1.1.7.0, specified. Busadress=0 Error=0x0	Variablenname oder Kennung in zenon ist ungültig. (Die OID gibt es nicht in der MIB am Agent) Mögliche weitere Ursache: Initialwerte für Traps sollen vom Agent ausgelesen werden, aber der Agent stellt auf das GET hin keine OIDs zur Verfügung, sondern sendet diese OIDs nur spontan als Traps. **errorStatus=2**: = no such item
error Trap Start 0x64!	Fehlermeldung im Editor. Windows Trap Service wurde nicht gestartet. Hinweis: Tritt auf, nachdem **Empfange Online Traps** in der Treiberkonfiguration (auf Seite 26) oder im Importassistenten aktiviert wurde.
Init TRAP Error 0x64	Meldung aus dem Diagnosis Server. Es werden keine Traps empfangen. Ursache: der Windows Service "**SNMP Trap**" wurde nicht gestartet. Hinweis: Der Service "**SNMP Trap**" (**snmptrap.exe**) benötigt die UDP-Ports 161 und 162. Dieser muss in der Firewall frei geschaltet sein. Wurde der Service gestartet, aber eine andere Anwendung belegt den Port 162, dann: ▸ kann der Treiber keine Traps empfangen ▸ wird keine Fehlermeldung angezeigt

EINTRÄGE IN LOG-DATEI

Unexpected Error during OID Translation INIT. Tanslation of OIDs will not be available.	Bei der Initialisierung der OID-Übersetzung trat ein Fehler auf. Die OID-Übersetzung ist für diese Treiberinstanz deaktiviert.
Unexpected Error during translation of OID [numerische OID]	Bei der Übersetzung der angegebenen OID trat ein unerwarteter Fehler auf.
Could not translate OID [numerische OID]	Die angegebene OID konnte nicht übersetzt werden. Möglicherweise ist sie nicht im Standard-MIB-Baum auffindbar.
OID Translation successfully initialized.	Die OID-Übersetzung wurde erfolgreich initialisiert.
Translation of OID [numerische OID] successfull	Die angegebene OID wurde erfolgreich übersetzt.

FEHLERNUMMERN

Fehlernummern zu **SnmpMgrRequest**:

ID	Beschreibung	Beschreibung
0	SNMPAPI_FAILURE	/* Generic error code */
1	SNMPAPI_SUCCESS	/* Generic success code */
2	SNMPAPI_ALLOC_ERROR	/* Error allocating memory */
3	SNMPAPI_CONTEXT_INVALID	/* Invalid context parameter */
4	SNMPAPI_CONTEXT_UNKNOWN	/* Unknown context parameter */
5	SNMPAPI_ENTITY_INVALID	/* Invalid entity parameter */
6	SNMPAPI_ENTITY_UNKNOWN	/* Unknown entity parameter */
7	SNMPAPI_INDEX_INVALID	/* Invalid VBL index parameter */
8	SNMPAPI_NOOP	/* No operation performed */
9	SNMPAPI_OID_INVALID	/* Invalid OID parameter */
10	SNMPAPI_OPERATION_INVALID	/* Invalid/unsupported operation */
11	SNMPAPI_OUTPUT_TRUNCATED	/* Insufficient output buf len */
12	SNMPAPI_PDU_INVALID	/* Invalid PDU parameter */
13	SNMPAPI_SESSION_INVALID	/* Invalid session parameter */
14	SNMPAPI_SYNTAX_INVALID	/* Invalid syntax in smiVALUE */
15	SNMPAPI_VBL_INVALID	/* Invalid VBL parameter */
16	SNMPAPI_MODE_INVALID	/* Invalid mode parameter */
17	SNMPAPI_SIZE_INVALID	/* Invalid size/length parameter */
18	SNMPAPI_NOT_INITIALIZED	/* SnmpStartup failed/not called */
19	SNMPAPI_MESSAGE_INVALID	/* Invalid SNMP message format */
20	SNMPAPI_HWND_INVALID	/* Invalid Window handle */
40	SNMP_MGMTAPI_TIMEOUT	
41	SNMP_MGMTAPI_SELECT_FDERRORS	
42	SNMP_MGMTAPI_TRAP_ERRORS	
43	SNMP_MGMTAPI_TRAP_DUPINIT	
44	SNMP_MGMTAPI_NOTRAPS	
45	SNMP_MGMTAPI_AGAIN	
46	SNMP_MGMTAPI_INVALID_CTL	
47	SNMP_MGMTAPI_INVALID_SESSION	
48	SNMP_MGMTAPI_INVALID_BUFFER	
99	SNMPAPI_OTHER_ERROR	/* For internal/undefined errors */
100	SNMPAPI_TL_NOT_INITIALIZED	/* TL not initialized */

101	**SNMPAPI_TL_NOT_SUPPORTED**	/* TL does not support protocol */
102	**SNMPAPI_TL_NOT_AVAILABLE**	/* Network subsystem has failed */
103	**SNMPAPI_TL_RESOURCE_ERROR**	/* TL resource error */
104	**SNMPAPI_TL_UNDELIVERABLE**	/* Destination unreachable */
105	**SNMPAPI_TL_SRC_INVALID**	/* Source endpoint invalid */
106	**SNMPAPI_TL_INVALID_PARAM**	/* Input parameter invalid */
107	**SNMPAPI_TL_IN_USE**	/* Source endpoint in use */
108	**SNMPAPI_TL_TIMEOUT**	/* No response before timeout */
109	**SNMPAPI_TL_PDU_TOO_BIG**	/* PDU too big for send/receive */
199	**SNMPAPI_TL_OTHER**	/* Undefined TL error */

10.3 Checkliste

- Ist der SNMP-Dienst korrekt installiert ? (Muss auf allen Geräten installiert sein, die ausgelesen werden sollen – außer bei Pingstatus-Abfrage)?
- Ist der korrekte Key installiert und ausgewählt? (default=public)?
- Ist das TCP/IP Protokoll installiert?

TRAPS UND OIB-ÜBERSETZUNG

Zur Diagnose kann der Netzwerkverkehr mit einem Tool wie Wireshark aufgezeichnet und analysiert werden. Wireshark übersetzt OIDs für die gleichen Standard-MIBs wie der SNMP-Treiber. Daher kann die OID-Übersetzung durch Analyse der Wireshark-Captures kontrolliert werden.

Problem	Diagnose	Mögliche Ursache
Der Agent sendet keine Traps.	Ereignisse, die Traps verursachen, werden ausgelöst. Wireshark zeigt aber keine eingehenden Traps an.	Der Trapversandt ist nicht korrekt konfiguriert. Die am Agenten eingestellte Zieladresse muss die IP-Adresse des Rechners mit dem SNMP Treiber sein.
Traps werden nicht in der Runtime angezeigt.	Wireshark zeigt eingehende Traps, angemeldete Variablen werden jedoch nicht aktualisiert.	▸ Der SNMP Trap Dienst von Windows läuft nicht. Dieser Dienst muss laufen, damit Traps empfangen werden können. ▸ Der SNMP Trap Dienst läuft zwar, hat jedoch keinen Zugriff auf das Netzwerk. Die Windows Firewall (oder andere verwendete Firewall) muss dem Dienst das Empfangen von UDP Paketen auf dem Port 162 erlauben. ▸ Der Agent ist nicht korrekt konfiguriert. Die IP-Adresse des Agenten in der SNMP-Treiberkonfiguration muss der tatsächlichen IP-Adresse des SNMP Agenten entsprechen. Auf Basis der IP-Adresse erfolgt im Treiber die Zuordnung einer eingehenden Trap zu einem konfigurierten Agenten. ▸ Die Variablen haben falsche OIDs. Innerhalb eines Agenten werden die in der Trap enthaltenen Variable Bindings auf Basis ihrer jeweiligen OIDs den derzeit für den Agenten angemeldeten Trap-Variablen zugewiesen. Eine Trap-Variable wird erst aktualisiert, wenn eine Trap mit einem Variable Binding mit der gleichen OID ankommt. ▸ Die Traps sind als Wischer eingestellt. Wenn bei einem Agenten die Option **Trap Variablen nach jeder Trap zurücksetzen** aktiv ist, werden für alle von einer Trap aktualisierten Variablen unmittelbar nach dem Hochsenden der Werte die Variablen auf die Werte `0` bzw. `leerer String` zurückgesetzt. In einem Text-Element oder numerischen Element in einem Bild kann das so aussehen, als wären die Variablen nie aktualisiert worden. Lösungen: - Grenzwerte auf Wertänderungen der betroffenen Variablen - die betroffenen Variablen in Archiven aufzeichnen

		- Deaktivierung des Zurücksetzens
Die OID Übersetzung funktioniert nicht.	Obwohl die OID-Übersetzung aktiviert ist und Variablenwerte mit übersetzbaren OIDs ankommen (siehe Wireshark), werden in der Runtime keine Übersetzungen angezeigt.	Die MIB-Files sind nicht da. Die OID-Übersetzung funktioniert nur, wenn ein Ordner `SNMP-MIBS` mit den MIB-Dateien als Inhalt vorhanden ist. Pfad: `%ProgramData%\COPA-DATA\zenon7.60\CommunicationProfiles\SNMP-MIBS` Ist dies nicht der Fall, können keine OIDs übersetzt werden. Die einzige Übersetzung, die durchgeführt wird, ist das Ersetzen der ersten Ziffer (immer 1) durch „iso", das Root-Element aller OIDs.