zenon
by COPA-DATA

zenon Treiber Handbuch

IEC850

v.8.20

© 2020 Ing. Punzenberger COPA-DATA GmbH

Alle Rechte vorbehalten.

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

Inhaltsverzeichnis

1 Willkommen bei der COPA-DATA Hilfe ... 6

2 IEC850 ... 6

3 IEC850 - Datenblatt ... 7

4 Voraussetzungen ... 8

4.1 PC ... 9

5 Konfiguration ... 9

5.1 Anlegen eines Treibers ... 10

5.2 Einstellungen im Treiberdialog ... 13

5.2.1 Allgemein ... 14

5.2.2 Treiberdialog Basic settings ... 18

5.2.3 Connections ... 28

6 Variablen anlegen ... 55

6.1 Variablen im Editor anlegen ... 55

6.2 Adressierung ... 58

6.3 Treiberobjekte und Datentypen ... 62

6.3.1 Treiberobjekte ... 62

6.3.2 Zuordnung von Datentypen ... 65

6.4 Variablen anlegen durch Import ... 70

6.4.1 Online-Import von einem IEC 61850 Server ... 70

6.4.2 Offline-Import von einer SCL-Datei ... 75

6.4.3 XML Import ... 78

6.4.4 DBF Import/Export ... 79

6.5 Kommunikationsdetails (Treibervariablen) ... 85

7 Funktion Treiberkommandos ... 91

7.1 SwitchConnection ... 96

8 IEC850 Client Funktionen ... 97

8.1 Verbindungaufbau und Verbindungsausfallerkennung ... 98

8.2 Befehle (Control Model) ... 101

8.2.1 Select und Cancel ... 104

8.2.2 Additional Cause Diagnosis ... 106

8.2.3 Serviceparameter des Befehls ... 108
8.3 Service Tracking ... 111
8.4 Mapping von Double Point Values ... 115
8.5 Quality, Zeitstempel und Statusbits der Variable ... 118
8.5.1 TimeQuality und TimeAccuracy eines Befehls ... 121
8.6 Filetransfer ... 122
8.6.1 Ordnerinformation anfordern ... 123
8.6.2 Datei vom Server holen ... 123
8.6.3 Datei löschen ... 124

9 Reporting ... 124
9.1 Unbuffered Reporting ... 132
9.2 Buffered Reporting ... 135
9.3 RCB Aktivierung - Überprüfungsmöglichkeiten ... 138
9.4 Dynamische Datasets ... 141

10 Treibermodus zur Runtime ändern ... 143

11 TLS-verschlüsselte Kommunikation ... 148

12 Fehleranalyse ... 152
12.1 Analysetool ... 152
12.2 Treiberüberwachung ... 153
12.3 Checkliste ... 155

13 Anhang A - Beschreibung der Functional Constraints (FCs): ... 157

14 Anhang B - Abkürzungen für data object/data attribute ... 160

15 Anhang C - Conformance statement ... 173
15.1 Protocol implementation conformance statement (PICS) ... 173
15.2 Model implementation conformance statement (MICS) ... 182

16 Anhang D - PIXIT ... 183
16.1 PIXIT for Configuration ... 183
16.2 PIXIT for Association model ... 184
16.3 PIXIT for Server model ... 185
16.4 PIXIT for Data set model ... 189
16.5 PIXIT for Substitution model ... 190
16.6 PIXIT for Setting group control model ... 191

16.7 PIXIT for Reporting model 192
16.8 PIXIT for Logging model 196
16.9 PIXIT for GOOSE control block model 196
16.10 PIXIT for Control model 196
16.11 PIXIT for Time and time synchronization model 199
16.12 PIXIT for File transfer model 200
16.13 PIXIT for Service Tracking model 201

17 Appendix E - TICS 202
17.1 Part 6 203
17.2 Part 7-1 204
17.3 Part 7-2 204
17.4 Part 7-3 205
17.5 Part 7-4 205
17.6 Part 8-1 208

1 Willkommen bei der COPA-DATA Hilfe

ZENON VIDEO-TUTORIALS

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

ALLGEMEINE HILFE

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

PROJEKTUNTERSTÜTZUNG

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

LIZENZEN UND MODULE

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

2 IEC850

Die Kommunikation zwischen dem Treiber und der SPS basiert auf dem IEC 61850 Protokoll mit Client/Server *MMS* Diensten via TCP/IP (Profile A1/T1). Bei der Kommunikation verhält sich der Treiber wie ein Client (Master).

Sie können der Treiber manuell anlegen und konfigurieren oder - in der **Energy Edition** - mit Hilfe des **IEC850 Driver Configuration** Wizards, anhand einer SCL-Datei (SCD, CID, eventuell ICD etc).

Weiter können Sie die Variablen manuell anlegen oder sie importieren, auch ohne den Wizard:

- Offline - aus einer SCL-Datei
- Online - in der Kommunikation mit einem 850-Server.

Hinweis: Für vereinfachten Import der Variablen aus der Datasets in RCBs liefert der **IEC850 Driver Configuration** Wizard zusätzliche Möglichkeiten.

> **Achtung**
>
> Konfigurationen, die mit einem Treiber in einer höheren Version geändert wurden, können mit älteren Treiber nicht mehr geöffnet werden.

3 IEC850 - Datenblatt

Allgemein:	
Treiberdateiname	IEC850.exe
Treiberbezeichnung	IEC 61850 Treiber
Steuerungs-Typen	IEC 61850 kompatible IEDs (850-Server) Edition 1 oder Edition 2.
Steuerungs-Hersteller	ABB; GE Multilin; IEC; Kalki; NARI; NR Electric; SAT; Schweitzer Engineering Laboratories; SEL; Sprecher Automation

Treiber unterstützt:	
Protokoll	IEC 61850
Adressierung: Adress-basiert	Name based
Adressierung: Namens-basiert	--
Kommunikation spontan	X
Kommunikation pollend	X
Online Browsing	X
Offline Browsing	X
Echtzeitfähig	X

Treiber unterstützt:	
Blockwrite	--
Modemfähig	--
RDA numerisch	--
RDA String	--
Hysterese	--
erweiterte API	X
Unterstützung von Statusbit **WR-SUC**	X
alternative IP-Adresse	X

Voraussetzungen:	
Hardware PC	Standard Netzwerkkarte
Software PC	XML-Lite wird benötigt, ist Bestandteil von Microsoft Internet Explorer 7.0; xmllite.dll, kann auch explizit von der Microsoft Homepage heruntergeladen werden.
Hardware Steuerung	--
Software Steuerung	--
Benötigt v-dll	--

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

4 Voraussetzungen

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

4.1 PC

XML Lite, das ein Bestandteil des Microsoft Internet Explorer 7.0 ist, ist erforderlich; xmllite.dll kann auch separat von der Microsoft Website heruntergeladen werden.

Dieser Treiber unterstützt eine Verbindung über die Standard Netzwerkkarte des PCs. Achten Sie darauf, dass die Steuerung und der PC im selben Netzwerkbereich liegen und die Subnet-Masken auf beiden Geräten dementsprechend konfiguriert sind.

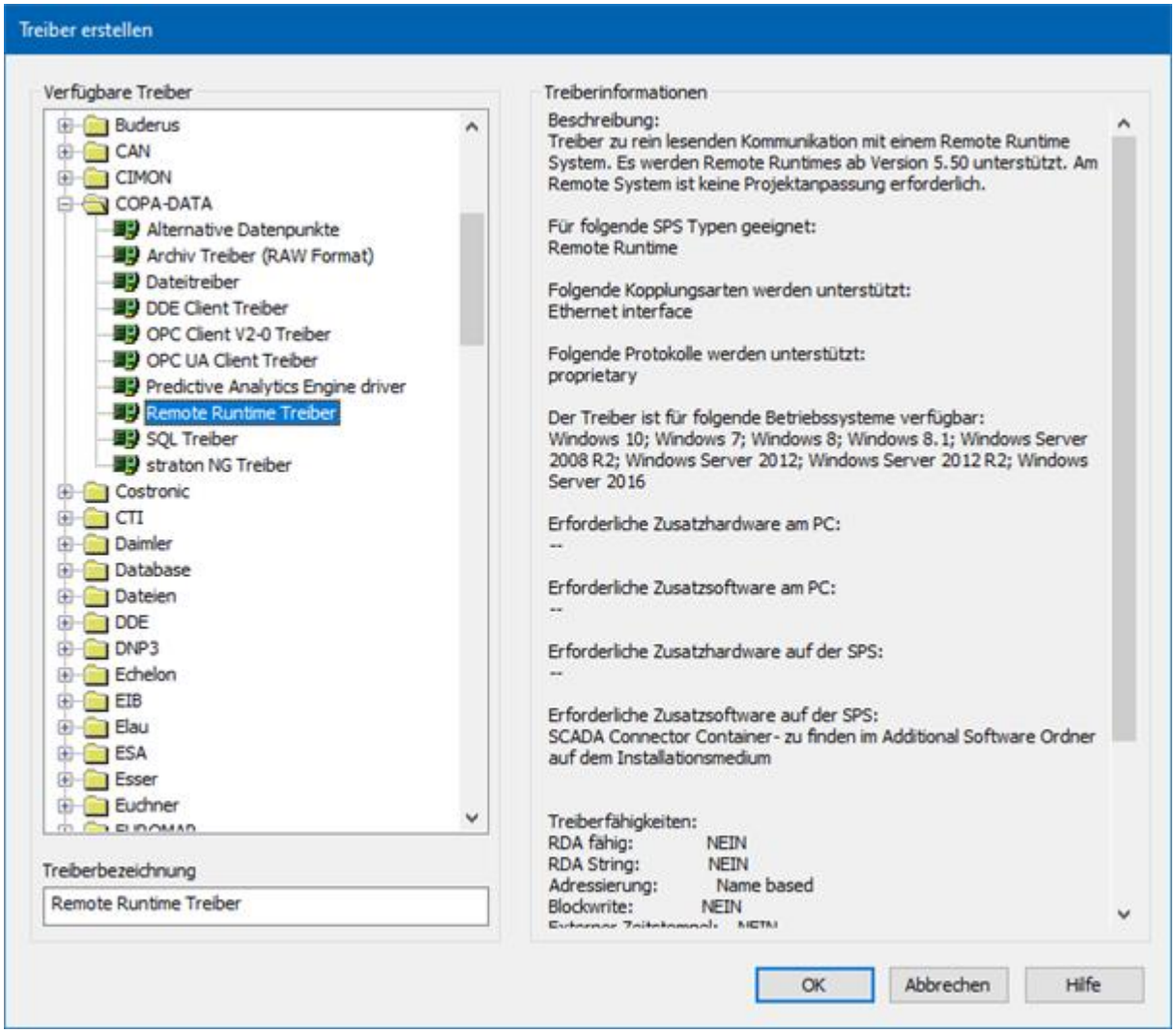
5 Konfiguration

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

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

5.1 Anlegen eines Treibers

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



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

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

DIALOG BEENDEN

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

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

TREIBER NEU ANLEGEN

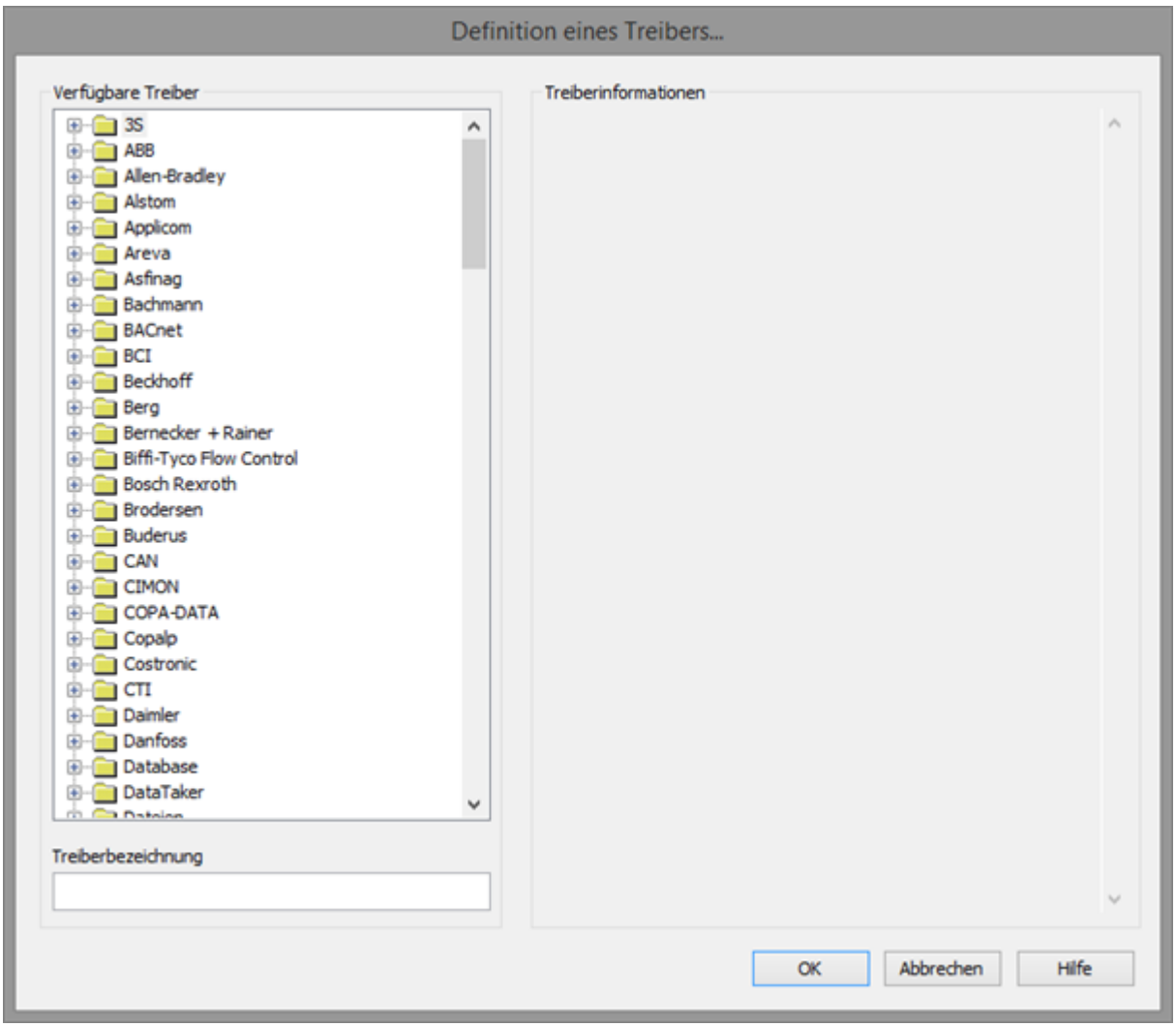
Um einen neuen Treiber anzulegen:

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

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

   Der Dialog **Treiber erstellen** wird geöffnet.

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



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

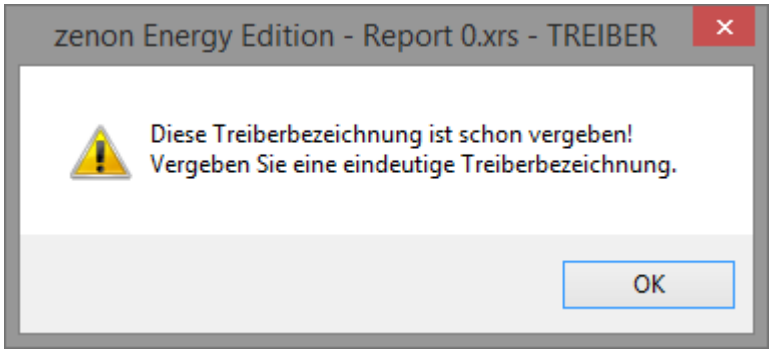
   - Die **Treiberbezeichnung** muss eindeutig sein.
     Wird ein Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
     Dies wird durch Klick auf die Schaltfläche **OK** evaluiert. Ist die Treiber im Projekt bereits vorhanden wird dies mit einem Warndialog angezeigt.
   - Die **Treiberbezeichnung** ist Bestandteil des Dateinamens.
     Daher darf Sie nur Zeichen enthalten, die vom Betriebssystem unterstützt werden. Nicht gültige Zeichen werden durch einen Unterstrich (_) ersetzt.
   - **Achtung:** Die Bezeichnung kann später nicht mehr geändert werden.

4. Bestätigen Sie den Dialog mit Klick auf die Schaltfläche **OK**.
   Der Konfigurationsdialog des ausgewählten Treibers wird geöffnet.

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

DIALOG TREIBERBEZEICHNUNG BEREITS VORHANDEN

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



ZENON PROJEKT

Bei neu angelegten Projekten werden die folgenden Treiber automatisch angelegt:

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

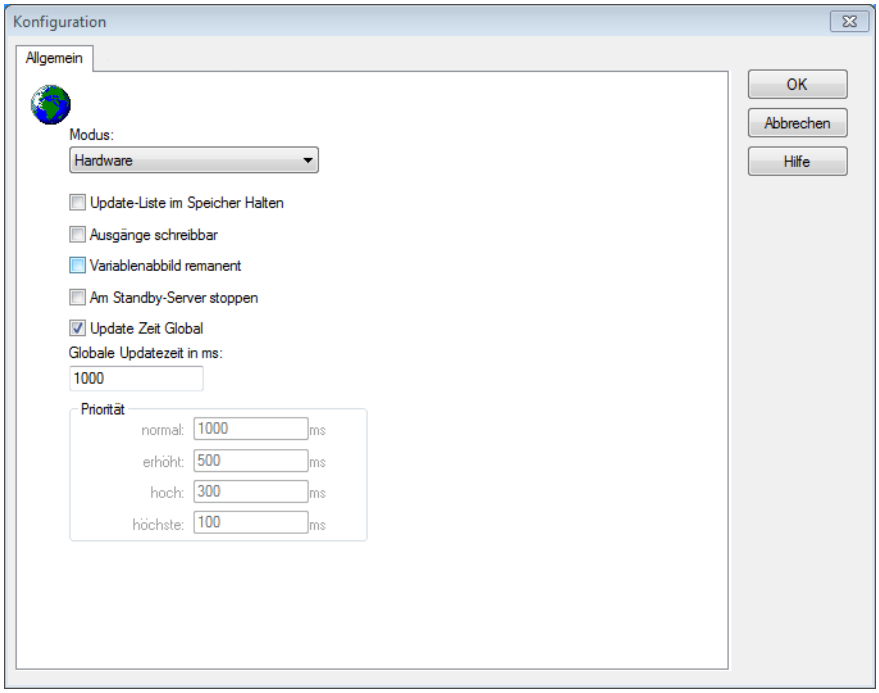
> **Info**
>
> In einem zenon Projekt müssen nur die benötigten Treiber vorhanden sein. Treiber können bei Bedarf zu einem späteren Zeitpunkt hinzugefügt werden.

5.2 Einstellungen im Treiberdialog

Folgende Einstellungen können Sie beim Treiber vornehmen:

5.2.1 Allgemein

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



Option	Beschreibung
Modus	Ermöglicht ein Umschalten zwischen Hardware und Simulationsmodus ▸ *Hardware*: Die Verbindung zur Steuerung wird hergestellt. ▸ Simulation - statisch: Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert. In diesem Modus bleiben die Werte konstant oder die Variablen behalten die über zenon Logic gesetzen Werte. Jede Variable hat seinen eigenen Speicherbereich. Zum Beispiel zwei Variablen vom Typ Merker mit Offset 79, können zur Runtime unterschiedliche Werte haben und beeinflussen sich gegenseitig nicht. Ausnahme: Der Simulatortreiber. ▸ *Simulation - zählend*: Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert.

Option	Beschreibung
	In diesem Modus zählt der Treiber die Werte innerhalb ihres Wertebereichs automatisch hoch. ▸ *Simulation - programmiert*: Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden von einem frei programmierbaren Simulationsprojekt berechnet. Das Simulationsprojekt wird mit der zenon Logic Workbench erstellt und läuft in einer in den Treiber integrierten zenon Logic Runtime ab. Details siehe Kapitel Treibersimulation.
Update-Liste im Speicher Halten	Einmal angeforderte Variablen werden weiterhin von der Steuerung angefordert, auch wenn diese aktuell nicht mehr benötigt werden. Dies hat den Vorteil, dass z B. mehrmalige Bildumschaltungen nach dem erstmaligen Aufschalten beschleunigt werden, da die Variablen nicht neu angefordert werden müssen. Der Nachteil ist eine erhöhte Belastung der Kommunikation zur Steuerung.
Ausgänge schreibbar	▸ *Aktiv*: Ausgänge können beschrieben werden. ▸ *Inaktiv*: Das Beschreiben der Ausgänge wird unterbunden. **Hinweis**: Steht nicht für jeden Treiber zur Verfügungen.
Variablenabbild remanent	Diese Option speichert und restauriert den aktuellen Wert, den Zeitstempel und die Status eines Datenpunkts. Grundvoraussetzung: Die Variable muss einen gültigen Wert und Zeitstempel besitzen. Das Variablenabbild wird im Modus Hardware gespeichert, wenn einer dieser Status aktiv ist: ▸ Benutzerstatus *M1* (*0*) bis *M8* (*7*) ▸ *REVISION(9)* ▸ *AUS(20)* ▸ *ERSATZWERT(27)* Das Variablenabbild wird immer gespeichert wenn: ▸ die Variable vom Objekttyp

Option	Beschreibung
	Kommunikationsdetails ist ▸ der Treiber im Simulationsmodus läuft. (nicht programmierte Simulation) Folgende Status werden beim Start der Runtime nicht restauriert: ▸ *SELECT(8)* ▸ *WR-ACK(40)* ▸ *WR-SUC(41)* Der Modus **Simulation - programmiert** beim Treiberstart ist kein Kriterium, um das remanente Variablenabbild zu restaurieren.
Am Standby Server stoppen	Einstellung für Redundanz bei Treibern, die nur eine Kommunikationsverbindung erlauben. Dazu wird der Treiber am Standby Server gestoppt und erst beim Hochstufen wieder gestartet. **Achtung:** Ist diese Option aktiv, ist die lückenlose Archivierung nicht mehr gewährleistet. ▸ *Aktiv*: Versetzt den Treiber am nicht-prozessführenden Server automatisch in einen Stopp-ähnlichen Zustand. Im Unterschied zum Stoppen über Treiberkommando erhält die Variable nicht den Status **abgeschaltet**, sondern einen leeren Wert. Damit wird verhindert, dass beim Hochstufen zum Server nicht relevante Werte in AML , CEL und Archiv erzeugt werden. Default: *inaktiv* **Hinweis:** Nicht verfügbar, wenn CE Terminal als Datenserver dient. Weitere Informationen dazu erhalten Sie im Handbuch zenon Operator im Kapitel CE Terminal als Datenserver.
Update Zeit Global	Einstellung für globale Update-Zeiten in Millisekunden: ▸ *Aktiv*: Die eingestellte **Globale Update Zeit** wird für alle Variablen im Projekt verwendet. Die bei den Variablen eingestellte Priorität wird nicht

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

DIALOG BEENDEN

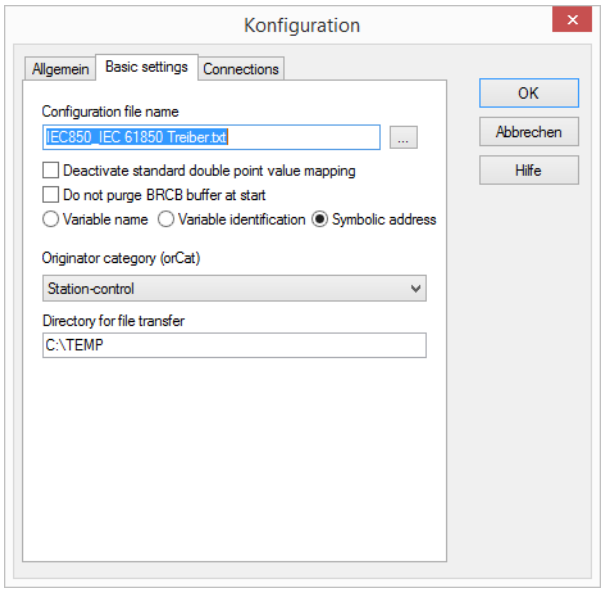
Option	Beschreibung
OK	Übernimmt alle Änderungen in allen Registerkarten und schließt den Dialog.
Abbrechen	Verwirft alle Änderungen in allen Registerkarten und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

UPDATE ZEIT SPONTANE TREIBER

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

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

5.2.2 Treiberdialog Basic settings



Parameter	Beschreibung
Configuration file name	Name der Datei, in der die treiberspezifischen Konfigurationen gespeichert sind. Klick auf die Schaltfläche **...** öffnet Dialog zur Auswahl eines Ablageortes. **Hinweis:** Den Aufbau dieser Konfigurationsdatei entnehmen Sie dem Kapitel Konfigurationsdatei des Treibers (auf Seite 20).
Deactivate standard double point value mapping	▸ *Inaktiv* : Die Werte von Double Point Values werden den Betriebselementen von zenon angepasst. Benutzen Sie diese Konfiguration, wenn Sie die Module der zenon Energy Edition verwenden möchten. Details finden sie im Kapitel Double Point Value Mapping (auf Seite 115). ▸ *Aktiv*: Die Werte der Double Point Values werden so wie sie sind an zenon weitergeleitet. In diesem Fall können Sie z. B. die Befehlsgabe-Funktionen von zenon Energy oder ALC nicht verwenden.

Parameter	Beschreibung
	Default: *inaktiv*
Do not purge BRCB buffer at start	▸ *Aktiv*: Die Puffer der BRCBs werden bei der ersten Verbindung nach dem Start der Runtime nicht am IED gelöscht. **Voraussetzung**: Für diesen BRCB wurde die zuletzt empfangene EntryID gespeichert. Wenn noch keine EntryID gespeichert wurde, dann wird die Eigenschaft ignoriert und der Treiber setzt doch BRCB.PurgeBuf=TRUE. Wirkt sich nur auf den ersten Versuch ein BRCB zu aktivieren nach dem Neustart der Runtime aus. Bei wiederholten Versuchen ein BRCB.RptEna=TRUE zu setzen - wenn die **RCBs enable retries** Zeit abgelaufen ist - wird ignoriert. **Hinweis**: Wurde die Runtime nicht gestoppt und nur die Verbindung zum IED abgebrochen ist die Einstellung irrelevant. Nach Wiederverbindung versucht der Treiber immer die gepufferte Werte zu holen; nicht löschen. Mehr im Kapitel **Buffered Reporting** (auf Seite 135).
Adressierung:	Gibt an, welche Variableneigenschaft im Treiber für die Adressierung - ObjectReference - verwendet wird. ▸ **Variable name** (**Name**) ▸ Variable identification (**Kennung**) ▸ Symbolic address (**Symbolische Adresse**) Default: **Symbolic address** **Hinweis**: Beim Import der Variablen befüllt der Treiber alle drei Variableneigenschaften mit gültigen Referenzen. Die Eigenschaften, die dann der Treiber für die Adressierung nicht verwendet, können Sie unbenennen.
Originator category (orCat)	Der Treiber sendet konfigurierter **orCat** Wert in Kommandos - in SBOw und Oper Strukturen - die er zum IED schreibt. Mögliche Einstellungen:

Parameter	Beschreibung
	▸ *1 - Bay-control* ▸ *2 - Station-control* ▸ *3 - Remote-control* ▸ *4 - Automatic-bay* ▸ *5 - Automatic-station* ▸ *6 - Automatic-remote* ▸ *7 - Maintenance* Default: *2 - Station-control*
Directory for file transfer	Zielordner des Dateitransfers. In diesem Ordner werden alle hinaufgeladenen Dateien gespeichert. Mehr im Kapitel Filetransfer (auf Seite 122).

5.2.2.1 Konfigurationsdatei des Treibers

Im Editor der Treiber speichert seine Konfiguration in der TXT-Datei wie in **Configuration file name** definiert. In der Runtime der Treiber bezieht seine Konfiguration aus der Kopie der Datei, die der Editor zur Verfügung gestellt hat.

AUFBAU EINER KONFIGURATIONSDATEI DES TREIBERS

Zeile	Beschreibung
1	Anzahl der konfigurierten Server
2 – (m-1)	Server Konfiguration (siehe Server Konfiguration)
m – n	evtl. weitere Server Konfiguration(en)

SERVER KONFIGURATION

Zeile	Beschreibung	Beispiel
1	Beginn einer Server Konfiguration (= **** SERVER ****)	*** SERVER ***
2	Anzahl der Items in dieser Server Konfiguration; abgekürzt: *CNTSRVITEMS*	44
3	Netzadresse	1

Zeile	Beschreibung	Beispiel
4	Servername	RELAY1
5	Primäre IP-Adresse	192.168.250.22
6	Primärer IP-Port	102
7	**Polling rate** Leseintervall in Millisekunden	1000
8	Calling AE qualifier	12
9	Called AE qualifier	12
10	Calling AP title[0]	1
11	Called AP title[0]	1
12	Calling AP title[1]	1
13	Called AP title[1]	1
14	Calling AP title[2]	1
15	Called AP title[2]	999
16	Calling AP title[3]	999
17	Called AP title[3]	1
18	Calling AP title[4]	-1
19	Called AP title[4]	1
20	Calling AP title[5]	-12851
21	Called AP title[5]	-1
22	Calling AP title[6]	-12851
23	Called AP title[6]	-12851
24	Calling AP title[7]	-12851
25	Called AP title[7]	-12851
26	Calling AP title[8]	-12851
27	Called AP title[8]	-12851
28	Calling AP title[9]	-12851

Zeile	Beschreibung	Beispiel
29	Called AP title[9]	-12851
30	**Max. auto used URCBs** Maximal Anzahl an Unbuffered Reports (URCBs), die der Treiber bei automatische Zuweisung pro Logical Device aktiviert	10
31	* - in neueren Konfigurationen bei alter Konfiguration: mit Komma getrennte Namen der zugeordneten Buffered Reports (BRCBs).	*
32	**Use preconfigured (SCL) options** 0 = Nachfolgend konfigurierte RCB Einstellungen (TrgOps, OptFlds, IntgPd, BufTm) werden verwendet 1 = die RCB Einstellungen, die im IEC61850 Server bereits vorkonfiguriert sind - in seiner SCL-Datei, werden benutzt	0
33	**Use Report-ID for RCB assignment** 0 = RCB Instanzen des Servers werden per Name identifiziert 1 = Report ID wird anstatt des Reportnamens im Dialog für RCB assignment verwendet.	0
34	**Use Authentication** für Master Typ der Authentifizierung: 0 = TLS deaktiviert; keine ISO-Authentication benutzt 1 = TLS aktiviert; der Treiber sendet beim Verbindungsaufbau im **Authentication String** das Passwort	0
35	**Authentication String** Entspricht dem Passwort bei gesicherter Kommunikation.	*MyVerySecretPassword*
36	Alternative IP-Adresse	
37	Alternativer IP-Port	0
38	TrgOp data-change: 0 = inaktiv; 1 = aktiv	1
39	TrgOp quality-change: 0 = inaktiv; 1 = aktiv	1

Zeile	Beschreibung	Beispiel
40	TrgOp data-update: 0 = inaktiv; 1 = aktiv	0
41	TrgOp integrity: 0 = inaktiv; 1 = aktiv	0
42	TrgOp general-interrogation: 0 = inaktiv; 1 = aktiv	1
43	**GetNameList on DO** 0 = Normales GetNameList 1 = Der Treiber liest das Objekt Model mittels Abfrage von Data Objekten (DO) für jede im Server vorhandene Logical Node und jeden im IEC61850 Standard definiertem Functional Constraint (FC). Diese Option wurde in der Version 8.00 aus dem Konfigurationsdialog entfernt und durch einen Automatismus ersetzt.	0
44	**Integrity Period**	7000
45	**Buffer Time**	500
46	**OptFlds** Optional fields des RCB	73
47	**RCBs enable reties** Zyklus in Sekunden in welchem versucht wird nicht erfolgreich aktivierte RCBs erneut zu aktivieren. Nur vorhanden wenn CNTSRVITEMS >= 45 ist	7
48	**Automatic Watchdog**	1
49	**Data consistency scan**	300
50	**Use SCADA network orIdend**	0
51	Anzahl der Client Konfigurationen	
52 – (m-1)	Client Konfiguration (siehe Client Konfiguration)	

CLIENT KONFIGURATION FÜR GESICHERTE KOMMUNIKATION

Zeile	Beschreibung	Beispiel
53	**Use Authentication** für Client	*2*

Zeile	Beschreibung	Beispiel
	Typ der Authentifizierung: 1 = Authentifizierung mit Passwort 2 = Authentifizierung mit Zertifikat (SHA1): 3 = Authentifizierung mit Zertifikat (SHA256) Default: *1*	
54	**MMS certificate file name** Name des Zertifikats, das für die Kommunikation verwendet wird.	*MMS_OK.pem*
55	Verwendung von **Transport Layer Security** für Kommunikation: 0 = TLS deaktiviert - keine verschlüsselte Kommunikation 1 = TLS aktiviert - Kommunikation erfolgt verschlüsselt, auf Basis der folgenden Parameter Default: *0*	*1*
56	**TLS 1.2 Cipher List** Verwendetes kryptographisches Verfahren für die abgesicherte Kommunikation bei Verwendung von TLS 1.2. Default: ""	
57	**TLS certificate file name** Name der Zertifikatsdatei die für die Kommunikation mit TLS verwendet wird.	*TLS_OK.pem*
58	**TLS peer certificate subject** Verwendete Zertifikats-Identifizierung. Das für die Kommunikation verwendete Zertifikat muss den hier eingetragenen **SUBJECT** enthalten. Bei Nicht-Übereinstimmung wird die Verbindung abgebrochen. Diese Konfiguration ist optional Default: ""	
59	**TLS session resumption timeout** in Sekunden	*43200*

Zeile	Beschreibung	Beispiel
	Maximale Zeitdauer eines benutzten Schlüssels, bevor er erneuert wird. Es erfolgt nur eine Erneuerung des Schlüssels. Das Zertifikat wird dabei nicht aktualisiert. Nach Ablauf der projektierten Zeit wird das Zertifikat erneuert. Zeitangabe in Sekunden. Default: *43200*	
60	**TLS session max resumption bytes** Datenmenge, die für einen Schlüssel verwendet wird, bevor er erneuert wird. Nach Übertragung der konfigurierten Datenmenge wird die Verschlüsselung erneuert. Das Zertifikat wird dabei nicht aktualisiert. Angabe in Bytes. Default: *1048576*	*1048576*
61	**TLS session renegotiation timout** in Sekunden Maximale Dauer einer verschlüsselten Verbindung, bevor diese erneuert wird. Nach Ablauf der projektierten Zeit wird die Verschlüsselung neu initialisiert. Dabei werden Zertifikate und Schlüssel neu ausgetauscht. Default: *86400*	*86400*
62	**TLS session max renegotiation bytes** Datenmenge, für die eine verschlüsselte Verbindung benutzt wird, bevor diese erneuert wird. Nach Übertragung der projektierten Datenmenge wird die Verschlüsselung neu initialisiert. Dabei werden Zertifikate und Schlüssel neu ausgetauscht. Angabe in Bytes. Default: *10485760*	*10485760*
63	**TLS revocation list checking interval** in Sekunden Intervall innerhalb der die Zertifikatssperrliste abgefragt wird. Default: *21600*	*21600*
64	**TLS minimum version**	*0*

Zeile	Beschreibung	Beispiel
	Minimale TLS Version die für eine gültige Kommunikation akzeptiert wird. 0 = keine TLS Versionsbeschränkung 10 = TLS 1.0 11 = TLS 1.1. 12 = TLS 1.2. 13 = TLS 1.3. Default: *0*	
65	**TLS maximum version** Höchste TLS Version die für eine gültige Kommunikation akzeptiert wird: 0 = keine TLS Versionsbeschränkung 10 = TLS 1.0 11 = TLS 1.1 12 = TLS 1.2 13 = TLS 1.3 Default: *12*	*12*
66	**TLS 1.3 cipher Suite** Verwendetes kryptographisches Verfahren für die abgesicherte Kommunikation bei Verwendung von TLS 1.3. Default: ""	
85	**TLS certificate store path** Pfad zum Ablageort der Zertifikate, die für TLS-Kommunikation verwendet werden. Angabe als Absolutpfad.	*D:\PKI\850C\TLS*
86	**MMS certificate store path** Pfad zum Ablageort der Zertifikate, die für TLS-Kommunikation verwendet werden. Angabe als Absolutpfad.	*D:\PKI\850C\MMS*

Zeile	Beschreibung	Beispiel
m – n	evtl. weitere Client Konfigurationen	

CLIENT KONFIGURATION

Zeile	Beschreibung	Beispiel
1	Beginn einer Client Konfiguration (=*** *CLIENTCFG* ***)	*** CLIENTCFG ***
2	Anzahl der Items in dieser Client Konfiguration	1
3	**Hostname (RT computer name)** Name des Computers auf dem der Treiber läuft, der die Reports empfängt	WKS007
4	**ClientLN.iedName**	SCADA_Server 1
5	**orIdent**	
6	Anzahl der RCB Konfigurationen	2
7 – (i-1)	RCB Konfiguration (siehe RCB Konfiguration)	
i – (j-1)	evtl. weitere RCB Konfigurationen	
j	Anzahl der dynamischen Dataset Konfigurationen	
k – (l-1)	Dynamische Dataset Konfiguration (siehe dynamische Dataset Konfiguration)	
l – m	evtl. weitere dynamische Dataset Konfiguration	

RCB KONFIGURATION

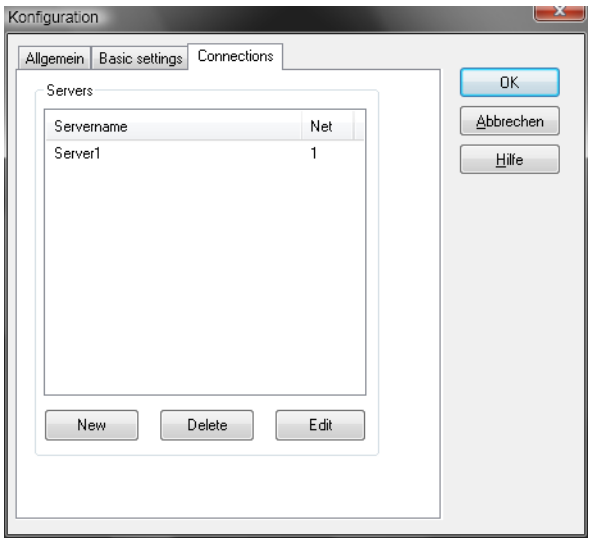
Zeile	Beschreibung	Beispiel
1	Beginn einer RCB Konfiguration (= *** *RCBCFG* ***)	*** RCBCFG ***
2	Anzahl der Items in dieser RCB Konfiguration	2
3	RCB Name bzw. ID.	UP_CTRL/LLN0/urcb_QxCSWI1_Pos02[RP]
4	Name des dynamischen Datasets.	NEW_DYN_DATASET

DYNAMISCHE DATASET KONFIGURATION

Zeile	Beschreibung	Beispiel
1	Beginn eines dynamischen Datasets (= **** DATASET ****)	*** DATASET ***
2	Anzahl der Items in dieser dynamischen Dataset Konfiguration	1
3	Name des dynamischen Datasets	NEW_DYN_DATASET
4	Anzahl der Object References des dynamischen Datasets	2
5	Object Reference	UPCTRL/Q1CSWI1CFPos
6 – n	evtl. weitere Object Reference.	UPCTRL/Q1CSWI1STPos

5.2.3 Connections

Die Verbindungen des IEC 850 Treibers werden in der Registerkarte **Connections** des Treiberdialogs eingestellt.



Parameter	Beschreibung
Servers	Liste der bereits konfigurierten Verbindungen zu den IEC 61850 Server.
New	Erstellt eine neue Verbindung zu einem IEC 61850 Server. Öffnet den Dialog **Server**.

Parameter	Beschreibung
Delete	Entfernt die ausgewählte Verbindung aus der Liste.
Edit	Öffnet den Dialog **Server**, um die ausgewählte Verbindung zu editieren

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.

5.2.3.1 Server

Ein Klick auf **New** oder **Edit** im Dialog **Connection** (auf Seite 28) öffnet den Dialog für die Konfiguration einer Verbindung zu einem IEC 61850 Server.

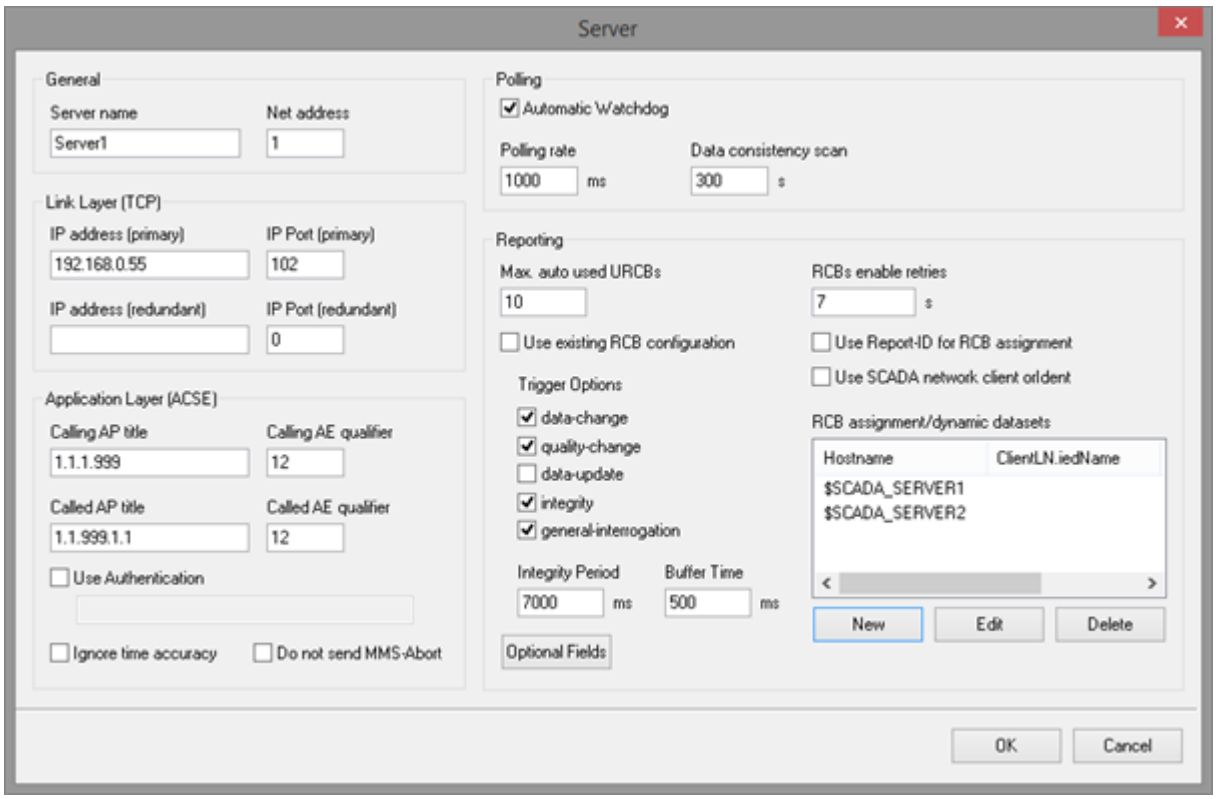
Hinweis: Dieser Dialog steht nur in englischer Sprache zur Verfügung.

BASISKONFIGURATION

Um eine erste Verbindung zu einem typischen 850-Server aufbauen zu können, genügt es in der Regel, folgende Eigenschaften zu konfigurieren:

- **Server name**
- **Net address**
- **IP address (primary)**

Damit können Sie in der Runtime mit dem Server kommunizieren oder im zenon Editor Variablen via Online-Import (auf Seite 70) anlegen.



Nachdem Sie dann die Variablen importiert und die Verbindung zum 850-Server in der Runtime erprobt haben sollen Sie jedoch noch mal die Treiberkonfiguration - für Reporting - sorgfältig überlegen und **RCB assignment** vornehmen.

GENERAL

Parameter	Beschreibung
Server name	Frei wählbarer Name. Wird bei Variablenimport für die zenon Eigenschaften **Name**, **Kennung** und **Symbolische Adresse** der Variablen verwendet, Syntax: *<Server name>!<ObjectReference>* - um die Eindeutigkeit der Variablenname zu gewährleisten in Systemen wo mehrere 850-Server gleiche IED und LDevice Namen beinhalten. **Hinweis**: halten Sie der **Server name** kurz (bereits eine Buchstabe ist ausreichend). Der Name dient nur Eindeutigkeit der Variablenname. Die Zuordnung der Variablen zur Verbindung (zum 850-Server) nimmt der Treiber aus der Eigenschaft **Net address**, nicht aus diesen Namen. Default: *Server1*

Parameter	Beschreibung
Net address	Entspricht der Eigenschaft **Netzadresse** (Eigenschaftengruppe **Adressierung**) bei der Variablenkonfiguration. Frei wählbare Nummer. Über diese Eigenschaft ordnet der Treiber die zenon Variablen dem 850-Server zu. Default: *1* Maximalwert: *65535*

LINK LAYER (TCP)

Verbindungseinstellungen zum 850-Server, mit dem die Verbindung hergestellt werden soll.

Parameter	Beschreibung
IP adress (primary)	IP-Adresse des 850-Servers, mit dem eine Verbindung hergestellt werden soll. Default: *192.168.0.55*
IP Port (primary)	IP-Port des 850-Servers, mit dem die Verbindung hergestellt werden soll. Default: *102*
IP adress (redundant)	Alternative IP-Adresse. Schlägt der Verbindungsversuch zur ersten IP Adresse fehl, so wird nach der Netz-Fehlerwartezeit (je nach Netzwerk *20-30 s*) umgeschaltet auf die alternative IP Adresse. Diese wird beibehalten bis entweder der Treiber neu gestartet wird (über Treiberfunktionen oder Neustart der zenon Runtime) oder die Verbindung fehlschlägt. Dann wird wieder versucht, mit der ersten IP-Adresse zu verbinden. **Achtung:** Um einen Verbindungsausfall zu erkennen, sollte die Eigenschaft **Automatic watchdog** aktiviert sein oder wenigstens eine Variable, z. B. **LLN0/Mod/stVal[ST]*, immer gepollt werden.
IP Port (redundant)	Alternativer IP-Port.

POLLING

Parameter	Beschreibung
Automatic Watchdog	Checkbox zur Aktivierung des Watchdog. Wenn aktiv sendet der Treiber automatisch je projektierter

Parameter	Beschreibung
	Polling rate ein Read-Request für **LLN0STMod$stVal* um einen eventuellen Verbindungsausfall zu erkennen. **Hinweis**: Wenigstens ein Datenattribut soll immer gepollt werden, auch wenn alle Daten des Projekts im Reports vorhanden sind. Sonst kann der Ausfall des IEC61850 Servers nicht erkannt werden. Deaktivieren Sie diese Eigenschaft nur dann, wenn im Projekt gewährleistet ist, dass mindestens eine gepollte Variable immer angemeldet ist. Default: *aktiv*
Polling rate	Definiert die Aktualisierungsfrequenz in Millisekunden. Der Treiber benützt diese Frequenz, um Daten, die nicht in aktiven **Reports** vorhanden sind, zu pollen. Dadurch wird auch ein eventueller Ausfall des IEC61850 Servers erkannt. Maximalwert: *4294967295* Wird ein höherer Wert eingegeben, wird er beim Speichern automatisch auf 1 geändert. Default: *1000 ms* **Achtung:** Der Datenaustausch mit der Runtime hängt weder von dieser Einstellung noch von der **globalen Updatezeit** ab. Diese ist fest auf *100 ms* eingestellt.
Data consistency scan	Definiert den Zyklus in dem der Treiber das Datenmodell auf dessen Konsistenz überprüft. Wird nur verwendet, wenn in projektierten Reports die Datasets einzelne Datenattribute anstelle von Datenobjekten enthalten. Default: *300 Sekunden* Eingabebereich: *0 - 999999*

REPORTING

Parameter	Beschreibung
Reporting	Einstellung für die Reports (auf Seite 124).
Max. auto used URCBs	Maximal Anzahl an Unbuffered Reports, die der Treiber bei automatischer Zuweisung pro Logical Device aktiviert, also zusätzlich zu RCBs, die im Dialog **RCB assignment** konfiguriert wurden.

Parameter	Beschreibung
	Default: *10* Eintrag ist nur für Unbuffered Reports (URCB) gültig, betrifft keine BRCBs . **Hinweis**: die Einstellung erleichtert die Nutzung von Reporting bereits in der Testphase des Projekts - noch bevor die Zieleinstellungen für Reporting vorgenommen wurden. Konfigurieren Sie die Nutzung von URCB im Dialog **RCB assignment** und setzen Sie die Eigenschaft auf 0 bevor Sie das Projekt freigeben. Mehr Information zur automatischen Zuweisung finden Sie im Kapitel **Unbuffered Report** (auf Seite 132).
Vertikale Gruppe der Optionsgruppe	Einstellungen für URCBs, die der Treiber automatisch zuweist und Default-Werte für gleichnamige Eigenschaften im Dialog **RCB assignment**.
Use existing RCB configuration	Sie können die folgenden **Trigger Options** unabhängig voneinander aktivieren/deaktivieren: ▸ *data-change* Default: *aktiv* ▸ *quality-change* Default: *aktiv* ▸ *data-update* Default: *inaktiv* ▸ *integrity* Default: *aktiv* ▸ *general-interrogation* Default: *aktiv* **Hinweis**: Nicht alle Server unterstützen TrgOps data-change und data-update zusammen. Auch TrgOp intergity kann zu unnötiger Überlastung der Kommunikation führen wenn für RCB eine zu kurze IntgPd (Integrity Period) im Server definiert wurde. Im Zweifelsfall setzten Sie TrgOps: data-change + quality-change + general-interrogation.
Integrity Period	Zeitintervall (IntgPd) in Millisekunden, in welchem der Server einen Integrity Report sendet. Default: *7000* ms

Parameter	Beschreibung
	Hinweis: nicht aktiv, wenn *TrgOp integrity* deaktiviert ist oder **Use preconfigured (SCL) options** aktiviert ist. Da ein Integrity Report im Normalfall keine Wertänderungen beinhaltet, wird dringend empfohlen, nur einen einzigen Report am Server mit *TrgOp: integrity* zu aktivieren. Bei einem aktivierten Integrity Report kann der Server einen Verbindungsabbruch schneller erkennen. zenon braucht diesen Report jedoch nicht. Default: *7000 ms*
Buffer Time	Zeitintervall (BufTime) in Millisekunden, in welchem der Server die Daten für einen Report sammelt. Default: *500 ms* **Hinweis:** Inaktiv, wenn die Option **Use preconfigured (SCL) options** aktiviert ist.
OptFlds	Öffnet Dialog zur Konfiguration der Optional Fields (auf Seite 40) des Reports. **Hinweis:** Ist die Option **Use preconfigured (SCL) options** aktiviert, werden die Optionen des Dialogs angezeigt, können aber nicht geändert werden.
Vertikale Gruppe der Einstellungen	Einstellungen relevant für Dialog **RCB assignment**.
RCBs enable retries	Konfiguration der Periode der erneuten Versuche, die nicht erfolgreich angemeldeten RCBs anzumelden. Wird die Anmeldung eines RCB von 850-Server abgelehnt, wird der Treiber es - je **RCBs enable retries** - erneut versuchen. Eingabe der Zeit in Sekunden. Eingabebereich: *0 - 999999* Default: *7* der Treiber versucht je 7s erneut RCB.RptEna=TRUE zu schreiben. **Hinweis:** ▸ Solange die Anmeldung nicht gelingt werden die Variablen gepollt. Deswegen soll die Periode grösser **Polling rate** sein, sonnst ist es nicht garantiert dass die

Parameter	Beschreibung
	betroffene Variablen die Initialwerte haben. ▸ Bei Eingabe 0 erfolgt kein Versuch ein RCB erneut anzumelden. ▸ Diese Einstellungen betrifft nur RCBs welche im Abschnitt **RCB assignment** gelistet sind.
Use Report-ID for RCB assignment	Auswahl ob der Treiber die RCB Instanzen anhand deren Namen (normales Verfahren) oder mit speziellen Verfahren über *Report ID* aussuchen und anmelden soll. ▸ *Inaktiv*: (empfohlen) Die RCB Instanzen des Servers werden per Name identifiziert. Dieses Verfahren wird von allen - mit IEC61850 Standard konformen - Servern unterstützt. ▸ *Aktiv*: Die *Report ID* wird anstatt des Reportnamens im Dialog für statische RCB Zuweisung im Dialog **RCB assignment** verwendet. Das Verfahren ist nur für spezielle Anwendungen geeignet. Der Treiber wird alle RCB Instanzen anmelden, die die gleiche ID haben. Erfordert entsprechende Konfiguration in der SCL-Datei des 850-Servers: ▸ für jeden 850-Client im System müssen eigene Report-IDs definiert sein; also unterschiedliche IDs je **Hostname**; ▸ es können mehrere RCBs die selbe ID haben. Der Treiber wird für jeden unterschiedlichen RCB genau eine Instanz mit der ID anmelden (enable). Damit werden nur einzelne Instanzen belegt. **Hinweis**: Der IEC 61850 Standard erlaubt dass mehrere RCBs den gleichen Wert in RptID haben. In der Runtime wird der Treiber **alle** RCBs mit der jeweiligen Report-ID aussuchen und anmelden. Damit kann man den Treiber so konfigurieren, dass er mit nur einem Eintrag in der Liste mehrere RCBs anmeldet; und auch solche die zur Zeit der Treiberkonfiguration noch nicht existiert haben. Default: *inaktiv* **Achtung:** Nicht aktivieren, wenn Sie nicht sicher sind, dass Sie diese spezielle Feature im Treiber nutzen möchten.

Parameter	Beschreibung
	Wenn Sie diese Option nachträglich ändern, müssen Sie alle RCB Zuweisungen im Dialog **RCB assignment** löschen, die Treiberkonfiguration schließen, erneut öffnen und die RCB Zuweisungen wieder erstellen.
Use SCADA network orIdend	Ist diese Checkbox aktiviert, wird in Befehlen der *orIdent* eingesetzt, basierend auf den Rechnernamen, auf dem der Benutzer den Befehl ausgelöst hat. Der Rechnername wird aus der Liste der **Hostnamen** aufgelöst. Diese Konfiguration erfolgt im Dialog **Client configuration** (auf Seite 41). Ist die Checkbox nicht aktiviert, wird der orIdent des Prozessführenden SCADA Servers verwendet. Default: *inaktiv* **Hinweis:** Der Default origin.orIdent *'zenon: <computer_name>'* wird in Befehlen verwendet, wenn: ▸ in der Client-Konfiguration für den entsprechenden Rechner (= **Hostname**) kein orIdent projektiert ist. ▸ in der Client-Konfiguration für den entsprechenden Rechner kein Eintrag mit **Hostname** projektiert wurde.
RCB assignment/dynamic datasets	Statische Report Control Block Zuweisungen und Einstellungen für Dynamic Data Sets . ▸ *Hostname* ▸ *ClientLN.iedName* ▸ *orIdent* wie im Dialog **Client configuration** (auf Seite 41) konfiguriert. Die Breite der Spaltenanzeige kann mit Mausklick vergrößert oder verkleinert werden. Mit der Scrollleiste kann der Ausschnitt verschoben werden. **Hinweis**: Wenn Sie in Ihrem zenon Projekt Redundanz verwenden (Eigenschaft **Netzwerk aktiv** aktiviert sowie **Server 1** und **Server 2** konfiguriert), müssen Sie hier wenigstens zwei Einträge erstellen: einen für den Prozessführenden Server und einen für den Standby Server im zenon Netzwerk.

Parameter	Beschreibung
	Per Default sind bei Neuanlage eines Servers bereits zwei Einträge (*$SCADA_SERVER1* und *$SCADA_SERVER1*) vorgegeben. Diese Einträge enthalten jeweils eine leere Konfiguration und müssen zwingend konfiguriert werden. Wird in der Runtime keine Client-Konfiguration für den Rechnernamen gefunden und läuft das Projekt Standalone oder am projektierten **Server 1**, so wird nach einer Client-Konfiguration *$SCADA_SERVER1* gesucht. Läuft das Projekt im obigen Fall am projektierten **Server 2**, so wird nach einer akternativen Client-Konfiguration *$SCADA_SERVER2* gesucht. Per Default ist die Konfiguration für *$SCADA_Server1* und *$SCADA_SERVER2* leer. Mit Klick auf die Schaltfläche **Edit** können Sie Ihre korrekte Konfiguration projektieren.
New	Erzeugt einen neuen Eintrag in der Liste. Öffnet den Dialog **Client configuraton (auf Seite 41).**
Edit	Öffnet den Dialog **Client configuration** (auf Seite 41), um den ausgewählten Eintrag zu bearbeiten.
Delete	Entfernt den ausgewählte Eintrag aus der Liste. **Achtung:** der Eintrag wird ohne weitere Sicherheitsabfrage sofort gelöscht.

APPLICATION LAYER (ACSE)

Einstellungen für ACSE.

Parameter	Beschreibung
Calling AP title	Einstellungen nach ISO 8650-1(ACSE), der Wert für *OSI ACSE AP Title* des Clients in der Universal Identifier Notierung. Default: *1.1.1.999* Soll im Standardfall nicht geändert werden.
Calling AE qualifier	Einstellungen nach ISO 8650-1(ACSE). Default: *12* Soll im Standardfall nicht geändert werden.
Called AP title	Einstellungen nach ISO 8650-1(ACSE), *OSI-AP-Title* des Servers im

Parameter	Beschreibung
	Format für Universal Identifier. Default: *1.1.999.1.1* Soll im Standardfall nicht geändert werden.
Called AE qualifier	Einstellungen nach ISO 8650-1(ACSE). Default: *12* Soll im Standardfall nicht geändert werden.
Use Authentication	Aktivieren Sie diese Checkbox, wenn Sie die Authentifikation nach ISO 8650-1 benutzen möchten. *Aktiv*: Der Treiber sendet beim Verbindungsaufbau den Authentication String mit.
[Eingabefeld für Authentication String]	Eingabefeld für *Authentication String*. Beachten Sie die Längenbeschränkung auf 55 Zeichen laut ISO 8327-1 OSI Session Protocol. **Hinweis:** Nur aktiv, wenn **Use Authentication** aktiv ist. Per default ist diese Option deaktiviert.
Ignore time accuracy	Checkbox um die Genauigkeit des empfangenen Zeitstempels zu ignorieren. ▸ *aktiv:* alle Nachkommastellen des empfangenen Zeitstempel werden ausgewertet, unabhängig davon, wieviele Nachkommastellen des Zeitstempels laut *TimeAccuracy* gültig sind. ▸ *inaktiv:* die Nachkommastellen des empfangenen Zeitstempels wird entsprechend *TimeAccuracy* ausgewertet.
Do not send MMS-Abort	Bestimmt die Art, wie Verbindung zum IED geschlossen wird. ▸ *aktiv*: Der Treiber schließt den TCP Socket anstatt ein MMS (ACSE) konformen Abort zu senden ▸ *inaktiv*: Treiber sendet den standardkonformen Abort-Dienst Default: *inaktiv*

DIALOG BEENDEN

Option	Beschreibung
OK	Übernimmt Einstellungen und schließt den Dialog.
Abbrechen	Verwirft alle Änderungen und schließt den Dialog.

Info

OSI-Werte (ACSE)

Der Treiber benutzt folgende festen OSI Selector Werte:

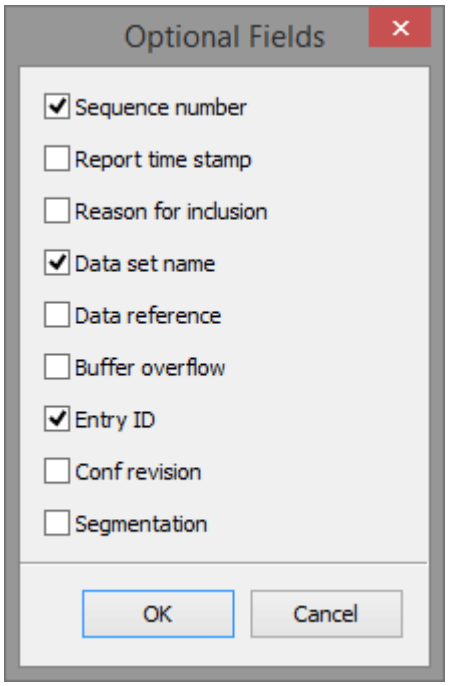
- Presentation Selector (OSI-PSEL) = "00000001"
- Session Selector (OSI-SSEL) = "0001"
- Transport Selector (OSI-TSEL) = "0001"

In einer Server SCL-Datei entspricht es folgenden Einträgen:

- <P type="OSI-AP-Title">41,999,1,1</P>
  - kodiert nach Basic Encoding Rules (IEC 8825) für Object Identifier - entspricht Default des Universal Identifiers in der **Called AP title**.
- <P type="OSI-AE-Qualifier">12</P>
  - entspricht Default in der **Called AE qualifier**.
- <P type="OSI-PSEL">00000001</P>
  - fest eingestellt im Treiber
- <P type="OSI-SSEL">0001</P>
  - fest eingestellt im Treiber
- <P type="OSI-TSEL">0001</P>
  - fest eingestellt im Treiber

5.2.3.1.1 Optional Fields

Die Optional Fields werden am Server bei Aktivieren eines Reports geschrieben. Diese entsprechen den Bits im Datenattribut OptFlds des RCB.



Per Default sind "**Sequence number**", "**Data set name**" und "**Entry ID**" aktiviert:

- Die Option **Data set name** ist per default aktiviert. Es wird empfohlen diese Option nicht zu deaktivieren.
  - Bei deaktivierter Option gilt:
    wird ein InformationReport ohne *DatSet* Name empfangen, so wird über *RptID* gesucht, welches Dataset im RCB verknüpft ist. Grundvoraussetzung ist, dass die *RptIDs* für den gesamten 850-Server eindeutig sind.
- **Entry ID** ist für Buffered Reports (BRCB) zwingend erforderlich um nach der Verbindung-Wiederaufbau die Gepufferte Werte anzufordern.

  Hinweis: In URCB das **Entry ID** ist nicht von Standard erfordert und wenn es in URCB nicht existiert, dann soll ein 850-Server es ignorieren, auch wenn angefordert wird. Manche 850-Server nummerieren auch die Unbufered Reports mit *EntryID*, was ein detailiertes Logging über den Empfang ermöglicht.

  Achtung: In zweifelsfall, oder wenn die Variablen in der Runtime - trotz zugewiesenen und aktivierten URCBs - ohne Werte bleiben - deaktivieren Sie das **Entry ID** Checkbox für alle URCBs; jedoch nur für URCBs, bei BRCBs muss **Entry ID** aktiv sein.
- **Sequence number** dient nur der Überprüfung, ob die Reports lückenlos übertragen werden. Wird eventuell in LOG ausgewertet.

WEITERE OPTIONAL FIELDS:

- **Report time stamp**
  sss
- **Reason for inclusion**
  Aktivieren Sie "**Reason for inclusion**" um Bei den Variablen mit den Statusbits **COTx** die Information auszulesen, wie deren Wert empfangen wurde: über ein spontaner Report oder GI oder Integrity Report.
- **Data reference**
  Eine Aktivierung der Option "**Data reference**" hat zur Folge, dass die Reports sehr groß werden. Diese Reports beinhalten dann die ObjektReference - als lesbare String - für jedes Datenattribut (Variable) des Datensets. Dies erleichtert die Analyse, z.B. im **Wireshark**. Diese Reports (mit **Data reference**) sind jedoch für freigegebene Systeme ungeeignet.
- **Buffer overflow**
  Wir vom Treiber ein Report mit gesetztem *BufOvfl* Flag empfangen, und ist dieser empfangene Report nicht der erste empfangene Report (nach Aktivierung des RCBs mit *PurgeBuf*), wird dies vom Treiber mit einer entsprechende Warnmeldung in der LOG-Datei protokolliert:

  IED reports a buffer overflow with Report-ID (Entry-ID)
- **Conf revision**
  x
- **Segmentation**
  y

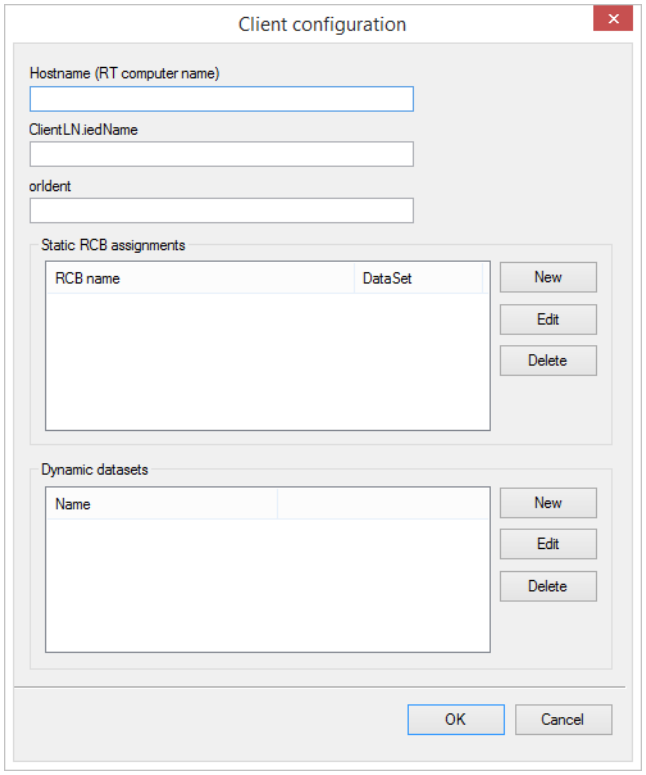
DIALOG BEENDEN

Option	Beschreibung
OK	Übernimmt Einstellungen und schließt den Dialog.
Abbrechen	Verwirft alle Änderungen und schließt den Dialog.

5.2.3.2 Client configuration

Ein Klick auf **New** oder **Edit** in der **RCB assignments/dynamic datasets** Liste des Server dialogs (auf Seite 29) öffnet den Dialog für die Einstellungen der Client-Konfiguration:

Hinweis: Dieser Dialog steht nur in englischer Sprache zur Verfügung.



Parameter	Beschreibung
Hostname (RT computer name)	Name des Computers auf dem der Treiber läuft, welcher die Reports empfängt: ▸ für Einzelplatz Projekte: *$SCADA_SERVER1* oder Rechnername des Computers, auf dem die zenon Runtime läuft ▸ im zenon Netzwerk: *$SCADA_SERVER1* oder Rechnername des Projekt-Servers **Server 1** *$SCADA_SERVER2* oder Rechnername des Projekt-Standby Servers **Server 2** Rechnername des Netzwerkclients für die Bestimmung des orIdent. Dies wird nur dann berücksichtigt, wenn die Option **Use SCADA network client orIdent** im Dialog **Server** aktiviert ist. **Hinweis**: Benutzen Sie nicht "*localhost*", sondern den Rechnernamen aus dem Betriebssystem. Der **Hostname** darf nicht leer sein, wenn mindestens eine der folgenden Konfigurationseinstellungen projektiert wurden.

Parameter	Beschreibung
	Achtung: Findet der Treiber auf der Liste keinen **Hostname** der mit der aktuelle **Runtime Rechnername** übereinstimmt - kann er die Reports nicht aktivieren. Es wird als Fehler/Warnung geloggt.

Parameter	Beschreibung
ClientLN.iedName	IED Name des IEC61850 Clients, wie in der SCD-Datei, und dort unter RCB.RptEnabled.ClientLN angegeben. Erlaubte Zeichen: A bis Z und a bis z und 0 bis 9 und Unterstrich _ maximal 64 Zeichen. Default: *leer* **Tipp:** Um den Treiber mit einer SCD-Datei einfach zu konfigurieren, benutzen Sie den **IEC850 Driver Configuration Wizard**. Weitere Details finden Sie in der Wizard-Dokumentation. **Hinweis:** Der Zustand des *Resv* Attributs wird bei Validierung der Verfügbarkeit des *RCB*s für statisch zugeordnete *URCBs* ignoriert, wenn die Option **ClientLN.iedName** projektiert ist.
orIdent	Eingabefeld für die Konfiguration des orIdent. Für jeden Rechner im zenon Netzwerk kann ein frei definierbarer orIdent Eintrag konfiguriert werden. Die Eingabe ist auf 32 Zeichen beschränkt. Default: *leer (= 'zenon: <computer_name>')* **Hinweis:** Welcher Originator in Befehlen verwendet wird, ist auch abhängig von der Konfiguration im Dialog Basic **Settings** (Eigenschaft **orCat**) und im Dialog **Server** (Checkbox **Use Client orIdent**). Ist die Checkbox **Use Client orIdent** nicht aktiv, wird in Befehlen immer der *orIdent* des Prozessführenden Servers verwendet. Ist die Checkbox **Use Client orIdent** aktiv: Der Default origin.orIdent *'zenon: <computer_name>'* wird in Befehlen verwendet, wenn: ▸ in der Client-Konfiguration für den entsprechenden Rechner (= **Hostname**) kein orIdent projektiert ist. ▸ in der Client-Konfiguration für den entsprechenden Rechner kein Eintrag mit **Hostname** projektiert wurde.

LISTE: STATIC RCB ASSIGNMENTS

Einstellungen für die statische Zuweisung von RCBs (gleich für Unbuffered und Buffered RCBs).

Hinweis: Buffered Reports müssen in der Treiberkonfiguration konfiguriert sein. Anderenfalls werden sie nicht verwendet.

Parameter	Beschreibung
RCB name	Name der konfigurierten RCBs. RCBs werden im Unterdialog Statically assigned RCB konfiguriert. Diese Konfiguration startet mit Klick auf die entpsrechende Schaltfläche **New**.
DataSet	Zeigt das dynamic DataSet, wenn der RCB eins zugewiesen wurde. Zuweisung: 1. Projektieren Sie in der Liste **Dynamic datasets** mit der dazugehörigen Schaltfläche **New** ein entsprechendes dynamic dataset. 2. Dieses dynamic dataset ist im Dialog **Statically assigned RCB** in der Dropdownliste **Dynamic dataset** auswählbar. **Hinweis**: die Nutzung von dynamic Data Set ist optional. Nicht alle 850-Server unterstützen dynamisches Anlegen von zusätzlichen Datensätzen.
New	Erzeugt einen neuen Eintrag in der Liste. Öffnet den Dialog **Statically assigned RCB** (auf Seite 47), um einen neuen Eintrag mit einem Reportnamen (oder einer Report-ID) anzulegen - Sie können entweder manuell eingeben oder den 850-Server online browsern und wenn nötig mit **Dynamic Dataset** aus der unteren Liste verknüpfen.
Edit	Öffnet den Dialog '**Statically assigned RCB** (auf Seite 47)', um den ausgewählten Eintrag zu bearbeiten.
Delete	Entfernt den ausgewählte Eintrag aus der Liste.

LISTE: DYNAMIC DATASETS

Einstellungen für dynamische Datensätze (dynamic data sets).

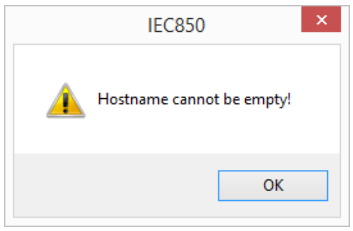
Hinweis: Dynamische Datensätze sind keine Voraussetzung für Buffered oder Unbuffered Reports. Es sind optionale Einstellungen, wenn der IEC61850 Server, der verwendet wird, dies unterstützt und in ausgewählten RCBs keine bzw. ungeeignete und austauschbare Konfiguration des Dataset hat.

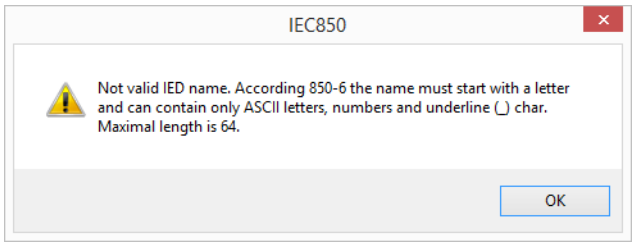
Für mehr Information siehe Dynamische Datasets (auf Seite 141).

Parameter	Beschreibung
Name	Zeigt die bereits eingegebenen Datasets - die stehen dann in der oberen Liste zur verfügung.
New	Öffnet den Dialog zur Erstellung eines Datasets.
Edit	Öffnet den Dialog **Dynamic DataSet configuration** (auf Seite 53), um den ausgewählten Eintrag zu bearbeiten.
Delete	Entfernt den ausgewählte Eintrag aus der Liste.

WARNHINWEISE

Bei fehlerhafter Projektierung werden Sie mit einem entsprechenden Warndialog darauf hingewiesen. Bestätigen Sie diesen Dialog mit Klick auf die Schaltfläche OK um zum Konfigurationsdialog zurückzukehren. Korrigieren Sie anschließend den Fehler.

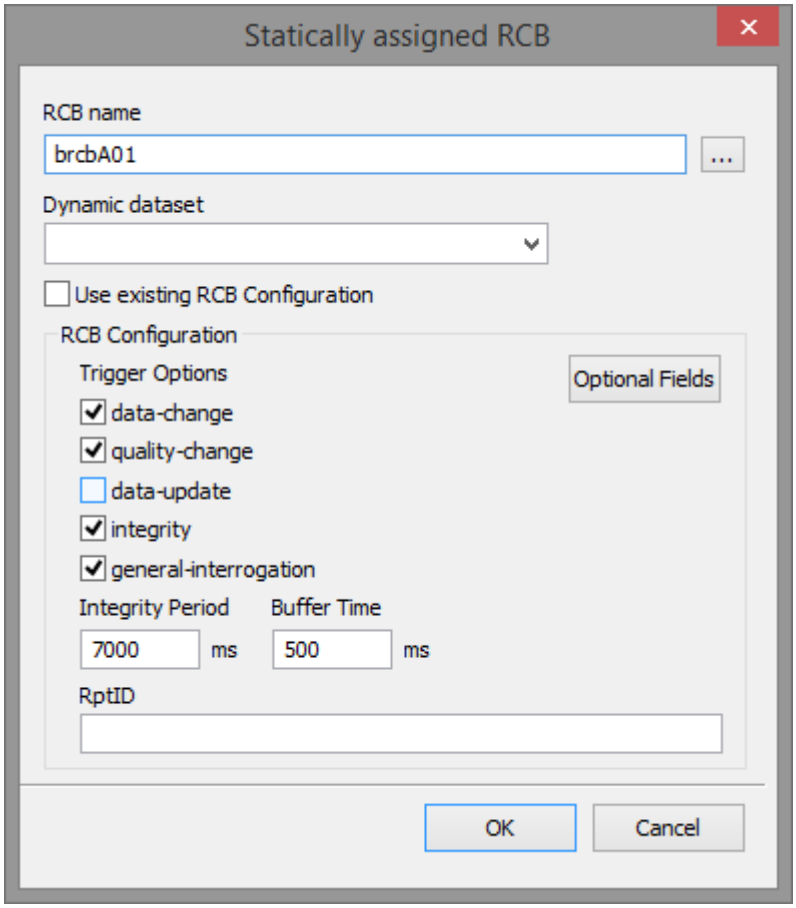




Hinweis: Der Warndialog ist nur in englischer Sprache verfügbar.

5.2.3.2.1Statically assigned RCB

Ein Klick auf **New** oder **Edit** in der **Static RCB assignment** Liste des Dialogs **Client configuration** (auf Seite 41) öffnet den Dialog, um eine Zuweisung zu RCB zu erstellen und/oder einem RCB ein dynamic Dataset zuzuweisen. RCBs sind über den Namen (standard Fall) oder ID (speziell Fall) definiert:



Parameter	Beschreibung
RCB name	Der Name (eindeutig) oder die RptID (eventuell nicht eindeutig) des Reports - buffered oder unbuffered - die der Treiber verwenden soll. Bei Verbindungsaufbau wird der Treiber konfigurierte RCB Instanzen anmelden (enable). Muss genau dem Namen/ReportID in der Steuerung (IEC61850 Server) entsprechen. Ein Klick auf **...** startet beim ersten Aufruf das Online-Browsing vom konfigurierten 850-Server. Bei erneutem Aufruf wird das Ergebnis des ersten Aufrufs aufgelistet. Wenn der Server läuft, werden alle Instanzen jedes Report Control Blocks, die in

Parameter	Beschreibung
	Daten Model des Servers vorhanden sind aufgelistet. Mehr Details finden Sie weiter in diesem Kapitel. **Hinweis:** Per Default wird hier **RCB name** erwartet. Der RCB Name ist mit voller *ObjectReference* gespeichert und dadurch eindeutig. Die Benuztzung des **RCB name** kann mithilfe der Eigenschaft **Use Report-ID for RCB assignment** in den **Server** (auf Seite 29) Eigenschaften auf Report-ID geändert werden. **Report-ID**: die Report-ID wird so wie von der 850-Server empfangen gespeichert, somit kann gleiche ID mehrere RCB zutreffen. Der Standard erlaubt dass mehrere RCBs gleicher Wert in RptID haben. **Achtung:** In der Runtime wird der Treiber alle RCBs mit der jeweiligen Report-ID aussuchen und anmelden. Bei Konfigurationsfehler könnte es zu Folge haben dass der Treiber alle Instanzen bestimmtes RCB anmeldet obwohl sie alle die gleiche Daten liefern. Damit bleibt dann auch keine freie Instanz für die andere Clients. **Hinweis:** Wenn auch die Runtime zur selben Zeit auf dem selben Computer läuft, manche 850-Server können die zweite Verbindung von gleicher IP verweigern. Dann kann der Treiber im Editor die RCBs nicht ausbrowsen. Schließen Sie die Runtime vorher ab.
Dynamic DataSet	Optional können Sie ein Data Set aus der Dropdownliste der bereits im Dialog **Dynamic dataSet configuration** (auf Seite 53) eingegebenen Data Sets auswählen. **Hinweis:** Wenn Sie hier kein Data Set definieren, wird für dieser RCB ein Data set verwendet, das in der Steuerung (IEC61850 Server) bereits vordefiniert wurde (in SCL-Datei).
Use existing RCB configuration	

Parameter	Beschreibung
	aktiv: Der Treiber aktiviert einen Report, ohne die Datenattribute des RCB zu überschreiben. Dadurch wird der Report aufgrund der Inhalte der SCL-Datei des Servers definiert. Folgende Datenattribute sind davon betroffen: ▸ IntgPd ▸ BufTime ▸ TrgOps ▸ OptFields *inaktiv*: der Treiber schreibt die Datenattribute des RCB während der Aktivierung. Der Treiber überschreibt die Datenattribute des RCB im IED nur dann, wenn diese nicht mit den Einstellungen des Treibers übereinstimmen. Default: *inaktiv*

RCB CONFIGURATION

Parameter	Beschreibung
Trigger Options	Sie können die folgenden **Trigger Options** unabhängig voneinander aktivieren/deaktivieren: ▸ *data-change* Default: *aktiv* ▸ *quality-change* Default: *aktiv* ▸ *data-update* Default: *inaktiv* ▸ *integrity* Default: *aktiv* ▸ *general-interrogation* Default: *aktiv* **Hinweis**: Nicht alle Server unterstützen TrgOps data-change und data-update zusammen. Auch TrgOp intergity kann zu unnötiger Überlastung der

Parameter	Beschreibung
	Kommunikation führen wenn für RCB eine zu kurze IntgPd (Integrity Period) im Server definiert wurde. Im Zweifelsfall setzten Sie TrgOps: data-change + quality-change + general-interrogation.
Integrity Period	Zeitintervall (IntgPd) in Millisekunden, in welchem der Server einen Integrity Report sendet. Default: *7000* ms **Hinweis:** nicht aktiv, wenn *TrgOp integrity* deaktiviert ist oder **Use preconfigured (SCL) options** aktiviert ist. Da ein Integrity Report im Normalfall keine Wertänderungen beinhaltet, wird dringend empfohlen, nur einen einzigen Report am Server mit *TrgOp: integrity* zu aktivieren. Bei einem aktivierten Integrity Report kann der Server einen Verbindungsabbruch schneller erkennen. zenon braucht diesen Report jedoch nicht. Default: *7000 ms*
Buffer Time	Zeitintervall (BufTime) in Millisekunden, in welchem der Server die Daten für einen Report sammelt. Default: *500 ms* **Hinweis:** Inaktiv, wenn die Option **Use preconfigured (SCL) options** aktiviert ist.
Optional Fields	Öffnet Dialog zur Konfiguration der Optional Fields (auf Seite 40) des Reports. **Hinweis:** Ist die Option **Use preconfigured (SCL) options** aktiviert, werden die Optionen des Dialogs angezeigt, können aber nicht geändert werden.
RptID	Eingabefeld für die Report-ID, die der Treiber bei Aktivierung von RCB (enable) vergibt. Ist dieses Eingabefeld leer, wird die RCB mit der ID weiterverwendet, wie sie bereits dem IED bekannt ist.

DIALOG BEENDEN

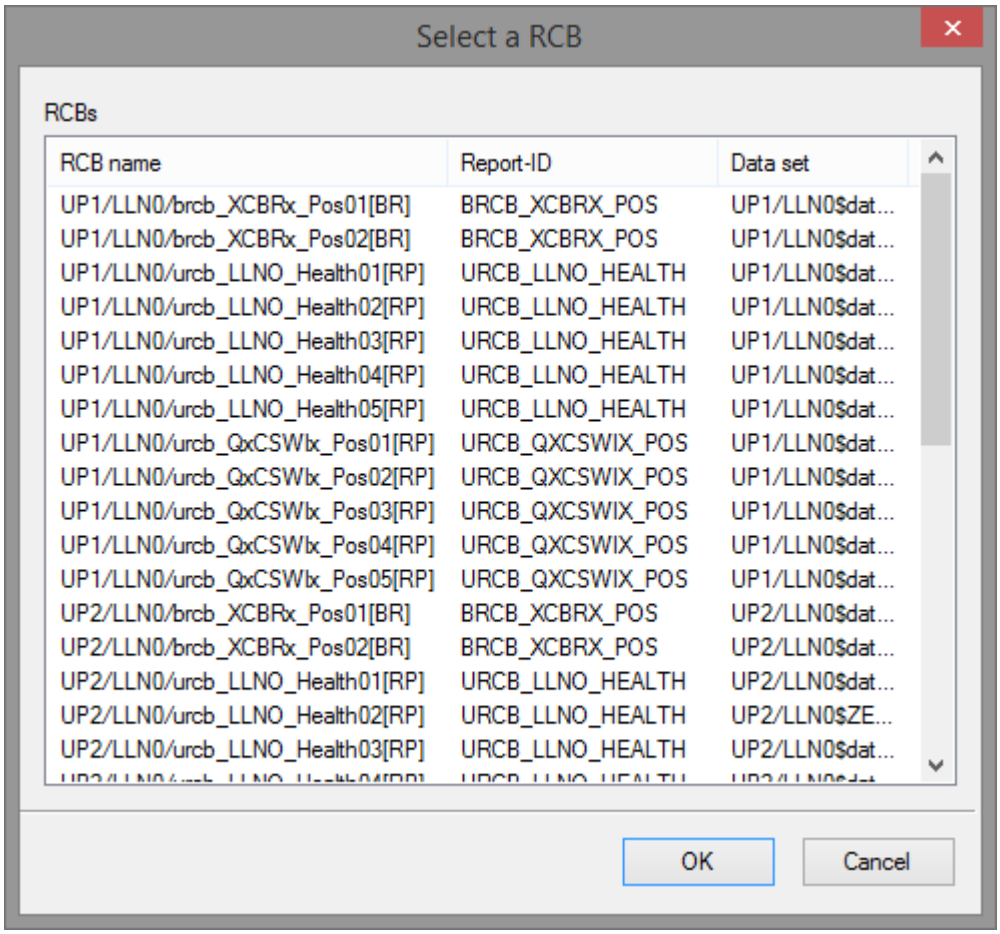
Option	Beschreibung
OK	Übernimmt Einstellungen und schließt den Dialog.
Abbrechen	Verwirft alle Änderungen und schließt den Dialog.

ONLINE-BROWSING DER RCB INSTANZEN

Ein Klick auf **...** startet das Online-Browsing vom 850-Server und zeigt, wenn der Server läuft, alle Instanzen jedes Report Control Blocks, die in Daten Model des Servers vorhanden sind.

Achtung: Um die Verbindung zu der **IP Adresse** des 850-Server aufbauen zu können muss die Treiberkonfiguration vorher wenigsten ein Mal abgespeichert werden.

Nachdem der Treiber die RCBs online von dem IEC 61850 Server eingelesen hat, wird folgende Liste angezeigt:



Parameter	Beschreibung
RCB name	Name des RCBs, wie im Datenmodell auf dem Server konfiguriert.

Parameter	Beschreibung
Report ID	Identifikation des RCBs, wie im Datenmodell am Server in Attribut *RptID* konfiguriert.
Data set	Name des Datasets im RCB, wie im Datenmodell auf dem Server konfiguriert.
OK	Übernimmt Einstellungen und schließt den Dialog. Vor dem Schließen des Dialogs wird eine Validierung durchgeführt. Im Fehlerfall wird ein Warndialog angezeigt.
Cancel	Verwirft alle Änderungen und schließt den Dialog.

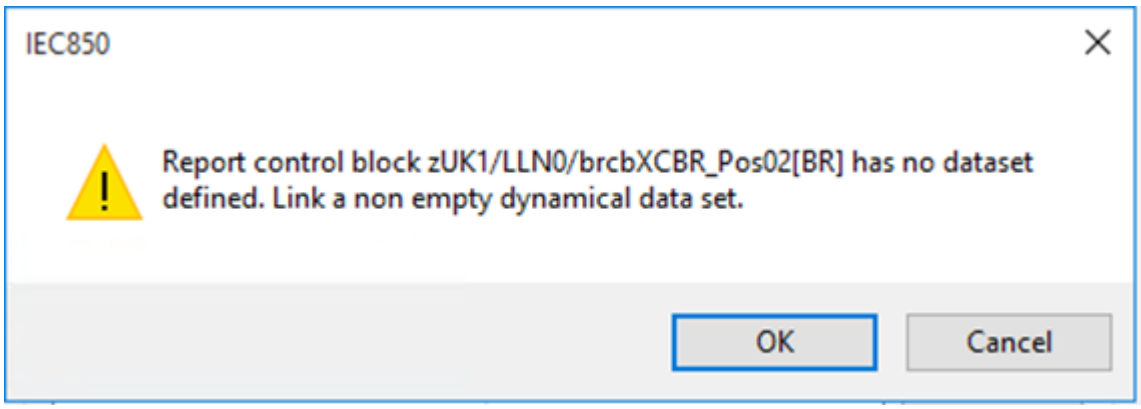
Wählen Sie die gewünschte Instanz eines Report Control Block aus der Liste und klicken OK. Sie können die Prozedur beliebig wiederholen um weitere RCBs zu zuweisen.

Hinweis: Achten Sie darauf für jeden **Hostname** die Instanzen mit anderen Namen (u/o anderen Nummer), bzw. anderen Report-ID, auszuwählen. Laut IEC61850 Standard eine Instanz des RCB darf nur von einen Client angemeldet (enabled) werden.

VALIDIERUNG DES DATASETS

Bei Bestätigung der Konfiguration durch Klick auf die Schaltfläche **OK** wird für alle neu hinzugefügten RCBs validiert, ob der RCB ein Datasetname vom 850-Server hat oder ein existierendes und nicht leeres dynamisches Dataset verknüpft wurde. Die Validierung meldet die RCBs, die zur Runtime keine Daten liefern können.

Im Fehlerfall wird eine Warnmeldung angezeigt.



Parameter	Beschreibung
OK	Bestätigt die Warnmeldung und schließt den Dialog.
Cancel	Bricht das Schließen ab und kehrt zum Dialog **Statically assigned RCB** zurück.

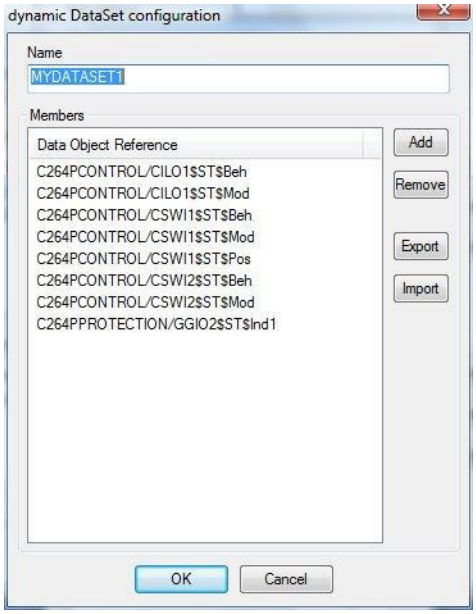
PROJEKTIERUNGSHINWEISE

Beachten Sie, dass nicht alle 850-Server Dynamische Datasets unterstützen. In diesem Fall können konfigurierte RCBs ohne Dataset-Namen im Datenmodell des Servers nicht genutzt werden.

Unterstützt der 850-Server nicht Dynamische Datasets, muss das IED mit einer korrigierten SCL-Datei (CID) aktualisiert werden.

5.2.3.2.2 Dynamic dataset configuration

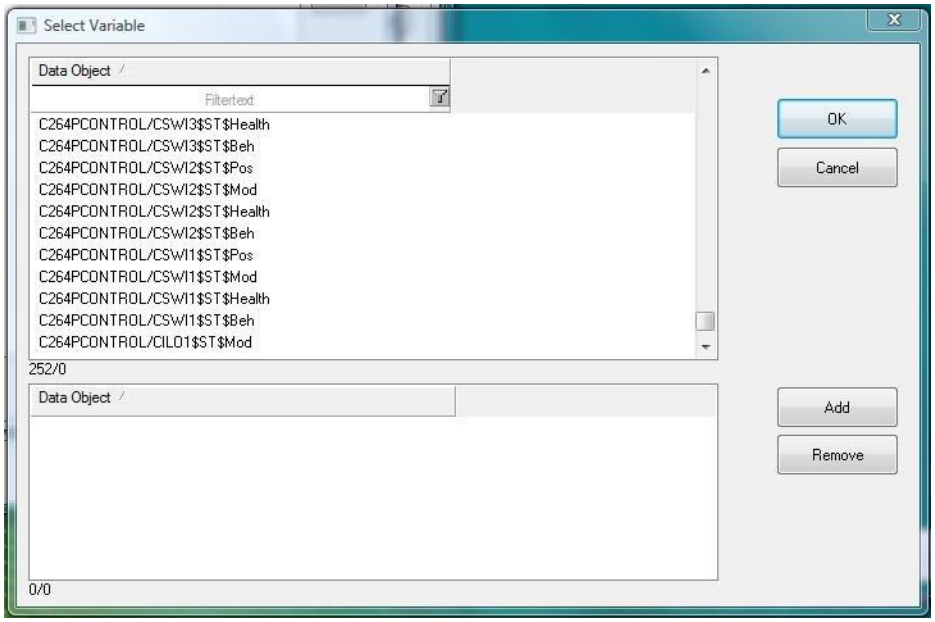
Ein Klick auf **New** oder **Edit** in der **Dynamic datasets** Liste des **Client configuration** (auf Seite 41) Dialogs öffnet den Dialog, um die Erstellung eines dynamischen Datensatzes (Data Set) zu definieren.



Parameter	Beschreibung
Name	Name des Data Set. **Hinweis**: geben Sie ganz kurze Namen, z.B. 'DS1'. Der Treiber wird der Name mit dem aktuellen **Runtime Rechnername** automatisch erweitern - um die eigene dynamische Datasets von Datasets anderer Clients unterscheiden zu können. Der RCB wird also angemeldet mit dem DatSet parameter: ZENON_**hostname**_<**Name**> Der RCB.DatSet Parameter hat laut IEC61850 Standard Ed.2 die maximale Länge 32 (genauer: 64/16$32). Die übrige Zeichen können abgeschnitten werden.
Members	Liste der bereits zugewiesenen Datenobjekte.

Parameter	Beschreibung
Add	Startet das Browsen von ObjectReferences aus der Variablenliste des zenon Projekts und zeigt alle Datenobjekte für die im Projekt eine Variable bereits erstellt wurde. **Hinweis**: Der Treiber listet die Datenobjekte nicht vom IEC61850 Server auf, sondern aus der Variablenliste im Editor. Wenn aber die Runtime zur selben Zeit auf dem selben Computer läuft, können die Variablen nicht durchsucht werden.
Remove	Entfernt den ausgewählte Eintrag aus der Liste.
Export	Exportiert Member in eine TXT-Datei.
Import	Importiert Member in eine TXT-Datei. im offenen Dialog: ▸ Wählen Sie die entsprechende Datei aus. ▸ Wählen Sie die gewünschten Variablen aus. ▸ Bestätigen Sie die Einstellungen durch einen Klick auf OK.

Nachdem Sie passende Projektvariablen ausgelesen haben, wird folgende Liste angezeigt:



Der Treiber zeigt alle - für das Reporting passenden - Datenobjekte aus der Variablenliste des zenon Projekts außer denen, die schon für das Data Set ausgewählt wurden. In der oberen Liste können Sie die aufgelisteten Variablen filtern und auswählen. Mehrfachauswahl ist dabei möglich (mit gedrückter **Alt-Taste** bzw. **Shift-Taste**).

Klick auf die Schaltfläche **Add** fügt Ihre Auswahl zu der unteren Liste hinzu. Bereits ausgewählte Objekte bleiben erhalten. Mithilfe der Schaltfläche **OK** fügen Sie die ausgewählte Variablen (als Referenzen zum Datenobjekten) zu den **Members** des dynamischen Data Sets hinzu.

6 Variablen anlegen

So werden Variablen im zenon Editor angelegt:

6.1 Variablen im Editor anlegen

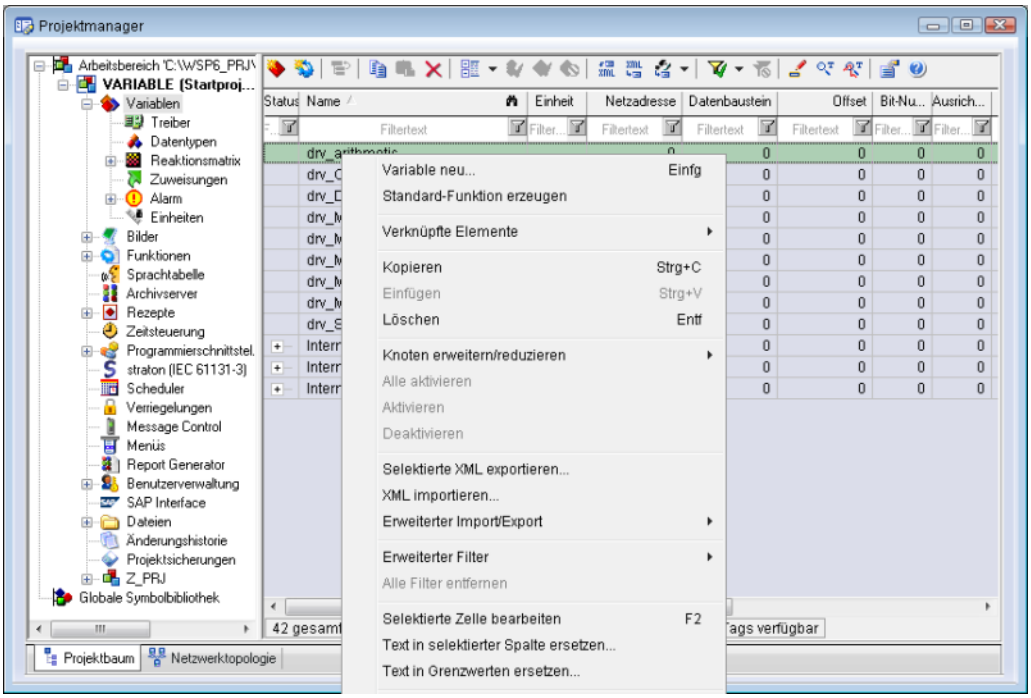
Variablen können angelegt werden:

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

DIALOG VARIABLE

Um eine neue Variable zu erstellen, gleich welchen Typs:

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

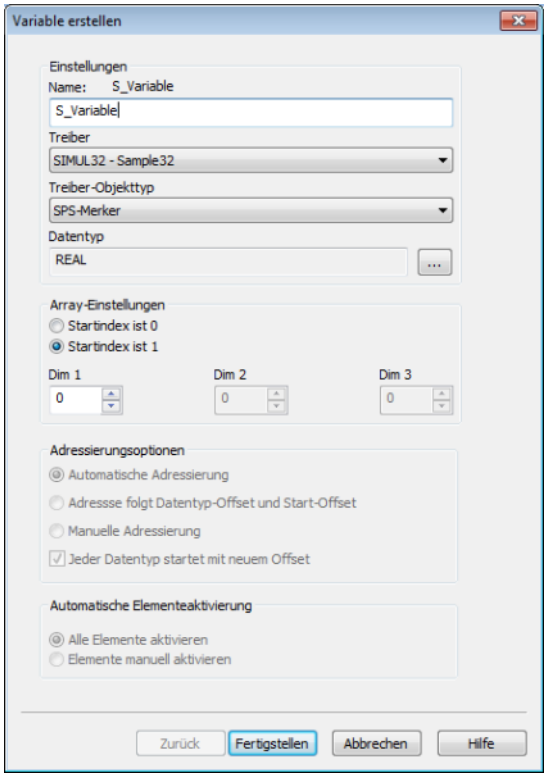


   Der Dialog zur Konfiguration der Variable wird geöffnet.

2. Konfigurieren Sie die Variable.

3. Welche Einstellungen möglich sind, hängt ab vom Typ der Variablen.

DIALOG VARIABLE ERSTELLEN



Eigenschaft	Beschreibung
Name	Eindeutiger Name der Variablen. Ist eine Variable mit gleichem Namen im Projekt bereits vorhanden, kann keine weitere Variable mit diesem Namen angelegt werden. Maximale Länge: 128 Zeichen **Achtung:** Die Zeichen **#** und **@** sind für Variablennamen nicht erlaubt. Bei Verwendung nicht zugelassener Zeichen kann die Variablenerstellung nicht abgeschlossen werden, die Schaltfläche **Fertigstellen** bleibt inaktiv. **Hinweis:** Manche Treiber erlauben die Adressierung auch über die Eigenschaft **Symbolische Adresse**.
Treiber	Wählen Sie aus der Dropdownliste den gewünschten Treiber. **Hinweis:** Sollte im Projekt noch kein Treiber angelegt sein, wird automatisch der Treiber für interne Variable (**Intern.exe**) geladen.
Treiberobjekttyp	Wählen Sie aus der Dropdownliste den passenden Treiberobjekttyp aus.

Eigenschaft	Beschreibung
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.

Folgende Treiber unterstützen die **Symbolische Adresse**:

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

ABLEITUNG VOM DATENTYP

Messbereich, **Signalbereich** und **Sollwert Setzen** werden immer:

- vom Datentyp abgeleitet
- beim Ändern des Datentyps automatisch angepasst

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

6.2 Adressierung

Die Adressierung basiert auf den Variablennamen.

Name	Beschreibung
Name	Frei wählbarer Name **Hinweis:** Die Adressierung der Variable erfolgt symbolisch über eine Referenz. Der Treiber übernimmt die Referenz aus der Eigenschaft **Name**, **Kennung** oder **Symbolische Adresse** der Variable, abhängig von den **Basic Settings** der Treiberkonfiguration (**Adressierung**). Wird der **Name** zur Adressierung der Variablen benutzt, muss er eine gültige ObjectReference beinhalten mit "*!" Präfix. Dabei ist * ein beliebiger Text. Bereits eine Buchstabe ist ausreichend. Zusätzlich zur Referenz identifiziert der Treiber die Verbindung mithilfe der Eigenschaft **Netzadresse** - also nicht anhand des Präfixes. Dies gilt für alle **Treiberobjekttypen**. **Achtung**: Der **Name** muss in jedem Projekt eindeutig sein. Wenn mehrere 850-Server die Variablen mit gleichen Referenzen liefern, differenzieren Sie die Variablennamen durch die Präfixe mit unterschiedlichen Servernamen oder verwenden Sie zur Adressierung die **Symbolische Adresse**. **Hinweis**: Beim **Online-Import** und **Offline-Import** werden **Name**, **Kennung, Symbolische Adresse** und **Netzadresse** automatisch konfiguriert.
Kennung	Geben Sie einen beliebigen Text ein; z. B. Ressourcenbezeichnung, Kommentar usw. Kann optional für die Adressierung verwendet werden.

Name	Beschreibung
	Wird beim Online-Import automatisch gesetzt.
Netzadresse	Netzadresse der Variable: Nummer der Verbindung in der Treiberkonfiguration. Diese Adresse bezieht sich auf die **Netzadresse** in der Liste **Connections - Server** der Treiberkonfiguration. Sie identifiziert den 850-Server, auf dem die Variablen abgelegt sind.
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.
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. Die **Symbolische Adresse** der Variable ist vergleichbar mit der ObjectReference laut IEC61850 Standard in einer Namenskonvention, angepasst an die Anforderungen der Eindeutigkeit der Adressierung in einem SCADA System. **Hinweis:** Die Adressierung der Variable erfolgt symbolisch über **Symbolische Adresse, Name** oder **Kennung** abhängig von den **Basic Settings** in der Treiberkonfiguration (**Adressierung**). Wird die **Symbolische Adresse** zur Adressierung der Variablen benutzt, muss sie eine gültige ObjectReference beinhalten mit **!* als Präfix. Zusätzlich zur Referenz identifiziert der Treiber die Verbindung mithilfe der Eigenschaft **Netzadresse**. Dies gilt für alle **Treiberobjekttypen**. **Hinweis**: Beim **Online-Import** und **Offline-Import** werden **Name**, **Kennung, Symbolische Adresse** und **Netzadresse** automatisch konfiguriert.
Treiber	Abhängig von der Art der Variablen, wird der Objekttyp während der

Name	Beschreibung
Anbindung/Treiberobjekttyp	Erstellung der Variablen ausgewählt. Der Typ kann hier später geändert werden. Für mehr Informationen siehe Abschnitt **Treiberobjekte** (auf Seite 62).
Treiber Anbindung/Datentyp	Datentyp der Variablen, die während der Erstellung der Variablen ausgewählt wurde. Der Typ kann hier später geändert werden. **Achtung:** Wenn Sie den Datentyp später ändern, müssen Sie alle Eigenschaften der Variablen überprüfen und gegebenenfalls anpassen; z. B. **Wertebereich**.
Treiber Anbindung/Priorität	Wird für diesen Treiber nicht verwendet. Der Treiber unterstützt keine zyklisch pollende Kommunikation in Prioritätsklassen.

Hinweis: Sie können mehrere Variablen mit gleicher ObjectReference anlegen, wenn die Variablennamen dafür unterschiedlich sind, z. B. mit unterschiedlichen Präfix.

SYNTAX DER REFERENZEN FÜR ADRESSIERUNG DER VARIABLEN

Die Adressierung der Variablen erfolgt symbolisch über ihre Referenzen. Der Treiber übernimmt die Referenz aus einer der Eigenschaften der Variable (**Symbolische Adresse, Name** oder **Kennung**), abhängig der **Basic Settings** der Treiberkonfiguration (**Adressierung**). Zusätzlich zur Referenz identifiziert der Treiber die Verbindung mithilfe der Eigenschaft **Netzadresse**.

Für die Referenzen gilt folgender Syntax: *SERVER!LD/LN/DataObject/DataAttr[FC]*

Dabei gilt:

- Das Präfix (*SERVER*) wird durch ein Ausrufezeichen (**!**) von den folgenden Bezeichnern getrennt. Es dient Eindeutigkeit der Variablen im Projekt, nicht der Adressierung.
- Die weiteren Bezeichner werden durch einen Schrägstrich (**/**) voneinander getrennt.
- Nur wenn ein Bezeichner selbst eine Struktur ist, werden deren Elemente durch einen Punkt (**.**) getrennt, zum Beispiel: *SERVER!LD/LN/DataObject/DataAttr.item[FC]*.
- Das Functional Constraint wird in Klamern [], am Ende, definiert.

Die Bezeichnungen sind:

Bezeichnung	Beschreibung
SERVER	Frei definierbarer Name, wobei bereits eine Buchstabe ausreichend ist. Das kann z.B. ein gekürzter Name des 850-Servers sein. Wenn die Referenz sich in der Eigenschaft **Name** der Variable befindet, dann soll das Präfix sicher stellen, dass alle Variablennamen im zenon Projekt eindeutig sind.

Bezeichnung	Beschreibung
	Empfehlung: Benutzen Sie die gleichen Namen, wie im Treiberdialog Dialog Server (auf Seite 29) definiert. Der Treiber erkennt allerdings die zu verwendende Verbindung nicht über den **Server name** (also nicht über Präfix in der Referenz), sondern über die Einstellung der **Netzadresse.**
LD	Name des IEDs and Logical Device, z.B. "GE_F650".
LN	Name des logischen Knotens (Logical Node), z.B. "LLN0", "MMXU".
DataObject	Name des Datenobjekts, z.B. "*Mod*", "*PhV.phsA*". Siehe Anhang B - Datenobjekte/Datenattribute.
DataAttr	Name des Datenattributes, z.B. "*stVal*", "*cVal.mag.f*". Siehe Anhang B - Datenobjekte/Datenattribute.
FC	Functional Constraint des Datenattributes, z.B. "*ST*", "*MX*" Siehe Anhang A - Beschreibung der Functional Constraints (FCs).

Achtung

Die Namenskonvention für Referenzen der Variablen entspricht nicht vollständig der im IEC61850-7-1 Standard definierten **ObjectReference**.

Im Standard wird ein Schrägstrich (**/**) nur zwischen **LD** und **LN** als Trennzeichen benutzt und ansonsten mit Punktern (**.**) getrennt. Wenn **DataObject** eine Struktur ist, machen es Punkte allerdings unmöglich zu unterscheiden, was **DataObject** und was **DataAttr** ist. Daher wird für die Variablenadressierung eine eigene Namenskonvention benutzt.

BEISPIEL

Der logische Knoten 'XCBR1' (Schutzschalter 1) hat unter anderem ein Datenobjekt 'Pos' von CDC (Common Data Class) 'Controllable Double Point' (DPC). Dieses Datenobjekt 'Pos' wiederum hat die Datenattribute 'stVal', 'q' und 't', deren Name und Semantik durch ihr CDC spezifisch in IEC61850-7-3 definiert sind.

LN/DO/DA[FC]	Name	IEC 61850 Standard
XCBR1/Pos	**Schaltposition**	gemäß LN Definition XCBR in 61850-7-4
XCBR1/Pos/stVal[ST]	**Positionswert**	gemäß CDC Definition DPC in 61850-7-3
XCBR1/Pos/q[ST]	**Quality**	gemäß CDC Definition DPC in 61850-7-3
XCBR1/Pos/t[ST]	**Zeitstempel**	gemäß CDC Definition DPC in 61850-7-3

LN/DO/DA[FC]	Name	IEC 61850 Standard
XCBR1/Pos/Oper.ctlVal[CO]	**Wert des Befehls**	gemäß Oper Typ Definition in 61850-8-1, Table E.9

Alle Objekte im Datenmodell eines 850-Servers sind auf diese Art aufgebaut. Deshalb weiß der Treiber zum Beispiel, dass die Attribute */q[ST] und */t[ST] Qualität und Zeitstempel für die Variable */stVal[ST] enthalten, weil stVal im Standard als das Haupt-Datenattribut in FC=*ST* definiert wurde. Dies erlaubt dem Treiber, die existierende Werte für Qualität und Zeitstempel automatisch den Istwerten der Hauptvariablen zuordnen.

Operate Befehl: Wenn in der Runtime der Benutzer auf eine Variable */Oper.ctlVal[CO] einen Wert gesetzt hat, kann der Treiber dank der spezifischen Variablennomenklatur automatisch bemerken, dass ein Operate Befehl zum IED gesendet werden soll. Dann sendet der Treiber automatisch die benötigte Sequenz an speziellen Telegrammen zum 850-Server.

ZEITSTEMPEL UND QUALITÄT

Zeitstempel- und Qualitätsattribut sind immer Teile von Variablen in zenon. Variablen */q und */t müssen nicht erstellt werden. Der Treiber benutzt automatisch den Zeitstempel und die Qualität des Datenobjekts für seine Variablen mit FC = *ST* und *MX* (FC steht für Functional Constraint (auf Seite 157)). Ohne Rücksicht darauf, ob eine Variable für Attribut '*t*' erstellt wurde, wird der empfangene Zeitstempel für alle *ST* und *MX* Variablen verwendet , zum Beispiel Variable **stVal** bei **CEL**, **AML** oder **Archivserver**. Auch werden ausgewählte Qualitätsbits, die in Attribut '*q*' empfangen wurden, den Statusbits der **stVal** Variable zugeordnet (siehe auch Kapitel **Quality, Zeitstempel und Statusbits der Variable**).

6.3 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.

6.3.1 Treiberobjekte

Folgende Objekttypen stehen in diesem Treiber zur Verfügung:

Treiberobjekttyp	Kanaltyp	Lesen	Schreiben	Unterstützte Datentypen	Kommentar
PLC marker	8	X	X	*REAL*, *LREAL*, *WSTRING*, *DINT*, *UDINT*, *BOOL*, *INT*, *UINT*, *STRING*,	Variablen die direkt Datenatributen in Datenmodell des 850-Servers entsprechen

Treiberobjekttyp	Kanaltyp	Lesen	Schreiben	Unterstützte Datentypen	Kommentar
				USINT, SINT	
File Transfer	9	X	X	*STRING*	Befehls- und Rückgabevariablen für den Dateitransfer. **Hinweis**: Die Stringlänge sollte maximaler Länge des Dateinamen (inklusive des Pfads) in der Steuerung entsprechen, z. B. für **!Command* variable max. 260 Zeichen (aber für **!Directory* mehr). Mehr Informationen finden Sie im Kapitel Filetransfer (auf Seite 122).
Connection state	36	X	--	*UDINT*	Interne Variablen von diesem Objekttyp zeigen den Verbindungsstatus zum 850-Server an. Die Variable muss als Referenz *ConnectionState* und die korrekte **Netzadresse** haben. **Beispiel: Name** oder **Symbolische Adresse**: **!ConnectionState*. Mehr Informationen finden Sie im Kapitel Verbindungsaufbau und Verbindungsausfallerkennung (auf Seite 98).

Treiberobjekttyp	Kanaltyp	Lesen	Schreiben	Unterstützte Datentypen	Kommentar
Command Info	11	X	--	*UINT*	Interne Variablen zeigen den aktuellen Status des Befehls; wird von Treiber zur Runtime übertragen (nicht von SPS). Mehr Informationen finden Sie im Kapitel Additional Cause Diagnosis (auf Seite 106).
Service tracking	64	X	--	*REAL*, *LREAL*, *DINT*, *UDINT*, *BOOL*, *INT*, *UINT*, *STRING*, *USINT*, *SINT*	Zusätzliche Service tracking (*[SR]) Variablen. Mehr Informationen finden Sie im Kapitel über Service Tracking (auf Seite 111).
Settings	65	X	X	*USINT*	Gemäß Standard 61850-8-1, 8.1.3.7: ▸ *TimeQuality* ▸ *TimeAccuracy* Für T-attribute in *SBOw*, *Oper* und *Cancel* Befehlen. Mehr Informationen finden Sie im Kapitel TimeQuality und TimeAccuracy (auf Seite 121).
Kommunikationsdetails	35	X	X	*BOOL*, *SINT*, *USINT*, *INT*, *UINT*, *DINT*, *UDINT*, *REAL*, *STRING*	Variablen für die statische Analyse der Kommunikation. Werte werden nur zwischen Treiber und Runtime übertragen, nicht zur

Treiberobjekttyp	Kanaltyp	Lesen	Schreiben	Unterstützte Datentypen	Kommentar
					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 85).

Legende:

X: wird unterstützt

--: wird nicht unterstützt

KANALTYP

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

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

6.3.2 Zuordnung von 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.

ZUORDNUNG VON DATENTYPEN VON DER SPS ZU ZENON DATENTYPEN

Steuerung	zenon	Datenart	Anmerkung
BOOLEAN	BOOL	8	
INT8	SINT	10	
INT8U	USINT	9	
INT16	INT	1	
INT16U	UINT	2	
INT24	DINT	3	
INT24U	UDINT	4	
INT32	DINT	3	
INT32U	UDINT	4	
INT64/INT128	LINT, LREAL	3	**Achtung:** Im (U)LINT Variablenwert werden nur die niedrigsten 52 Bits übernommen. Restliche Bits werden abgeschnitten und nicht berücksichtigt. Alternativ kann die Variable mit dem Datentyp LREAL projektiert werden. Dadurch ist es möglich, den Gesamtwert zu übertragen. Beachten Sie dabei, dass für Werte die größer als 52 Bit sind, die Präzision der niedrigsten Stellen verloren geht.
FLOAT32	REAL	5	
FLOAT64	LREAL	6	
ENUMERATED	INT	1	
CODED ENUM	UDINT, STRING	4, 12	*MMS Bitstring*
OCTET STRING	STRING	12	

Steuerung	zenon	Datenart	Anmerkung
VISIBLE STRING	STRING	12	
UNICODE STRING	WSTRING	21	Kodierung erfolgt in UTF-8
PACKED LIST	UDINT, STRING	4, 12	*MMS Bitstring*, umgedrehte Bitreihenfolge
TIMESTAMP	LREAL	6	nur Oktets 0..7 (siehe auch TimeQuality Bits mapping (auf Seite 97))
EntryTime	LREAL	6	
TriggerConditions, Check	UDINT, STRING	4, 12	*MMS Bitstring*, umgedrehte Bitreihenfolge
Tcmd	UDINT, BOOL	4, 8	*MMS Bitstring*, *FALSE* entspricht *0x40* und *TRUE* entspricht *0x80*
Dbpos	UDINT, STRING	4, 12	*MMS Bitstring*, Für */stVal* Variablen wird per default **DPI Mapping** (auf Seite 115) verwendet

DATENART

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

DATENTYP MMS BITSTRING

Die Datenattributtypen, die gemäß dem Standard (siehe IEC61850-8-1) dem *MMS Bitstring* zugeordnet werden, z. B. *CODED ENUM*, *PACKED LIST*, *Check* und *TrigerConditions*, entsprechen in zenon per default Datentyp *UDINT*. Beim Import legt der Treiber die *UDINT* Variablen an, mit dem **Wertbereich** des Datentypes - 32 Bits (*0..4294967295*). Wieviele Bits dann die Werte haben hängt von der Größe des *Bitstrings* des Daten Attributes im 850-Server und kann von einen 850-Client nicht automatisch erkannt und konfiguriert werden.

Wenn der 850-Server sich an SCL Vorgaben hält, wird die zenon Variable in der Runtime weit weniger als *4294967296* unterschiedliche Werte haben können. Es besteht jedoch kein Bedarf der **Wertbereich** der Variablen neu definieren oder einschränken. Für bessere lesbarkeit des Projekts können Sie die **Wertbereiche** doch manuell anpassen, nachdem die Variablen importiert wurden. Sie

können auch die Datentypen der Variablen mit eigenen, an *UDINT* basierenden, Datentypen austauschen.

Der Treiber ermöglicht *Bitstring* Variablen auch mit Datentyp *STRING*, anstatt *UDINT*, manuell anlegen oder konfigurieren. Der Wert der Variable, wenn sie als String angelegt wurde, enthält nur die Zeichen "0" und "1", in der gleichen Reihenfolge wie die Bits in dem IEC 61850 Standard definiert sind, und auf die Länge des *Bitstrings* eingeschränkt.

Im Fall von allen *Bitstrings*, auser *CODED ENUM* , wurde im IEC 61850 Standard das Bit 0 als höchstwertig in seinem Oktett definiert.

Achtung: Dies bedeutet dass der *UDINT* Wert der Variable eine umgedrehte Bitreihenfolge beinhalten wird.

Beispiel: der Wert des *TrgOps* Attribut - 850-Datentyp TriggerConditions (abgeleitet von *PACKED LIST*) - in MMS *Bitstring(6)*:

- als *String="011111"*
- als *UDINT=124* (*01111100b*), und nicht *11111b=31* (weil die Reihenfolge umgekehrt wurde).

Der Wert "*011111*" in String bedeutet, dass alle Bits, außer nicht erlaubten Bit 0, gesetzt sind (61850-8-1):

- Bit 0 - Reserved
- Bit 1 - data-change
- Bit 2 - quality-change
- Bit 3 - data-update
- Bit 4 - integrity
- Bit 5 - general-interrogation

Und weites Beispiel - der Wert "*010001*" (in *UDINT*=68, weil binär 0100 0100) hat *data-change* und *general-interrogation* Bits.

MMS BITSTRING IN BEFEHLSRICHTUNG

Befehlsvariablen (**/Oper.ctlVal[CO]*), die einem MMS Bitstring entsprechen - wie Tcmd oder andere CODED ENUM basierende bTypes, werden bei Import als Variablen mit dem Datentyp *UDINT* angelegt. Der Datentyp kann nachträglich manuell auf den Datentyp *BOOL* geändert werden, z. B. für Variablen **/TapChg/Oper.ctlVal[CO]*. Hat die Befehlsvariable den Datentyp *BOOL*, wird der Treiber bei **Sollwert setzen** den Wert *FALSE* auf *0x40 (lower)* und den Wert *TRUE* auf *0x80 (higher)* mappen. Somit kann diese *BOOL* Variable direkt mit Schaltaktionen in der **zenon Befehlsgabe** verknüpft werden.

DATENTYP ENUMERATED

Die Datenattributtypen, die gemäß dem Standard (siehe IEC61850-8-1) dem *ENUMERATED* zugeordnet werden, entsprechen in zenon Datentyp *INT*.

Beispiel: das Daten Attribut *Beh/stVal[ST]* von Common Data Class *ENS* (siehe 850-7-3 und 850-7-4) ist *ENUMERATED*, und der Standard definiert nur 5 mögliche Werte. Eine SCL-Datei des 850-Servers:

```
<EnumType id="Beh">

     <EnumVal ord="1">on</EnumVal>

     <EnumVal ord="2">blocked</EnumVal>

     <EnumVal ord="3">test</EnumVal>

     <EnumVal ord="4">test/blocked</EnumVal>

     <EnumVal ord="5">off</EnumVal>

</EnumType>
```

Es besteht jedoch kein Bedarf der **Wertbereich** der *INT* Variablen neu definieren oder einschränken. Für bessere lesbarkeit oder Auswertung im Projekt können Sie die **Wertbereiche** doch manuell anpassen, nachdem die Variablen importiert wurden. Sie können auch die Datentypen der Variablen mit eigenen, an *INT* basierenden, Datentypen austauschen.

Hinweis: eine **Reaktionsmatrix** können Sie bereits mit dem Datentyp verknüpfen - dann gilt sie für alle Variablen des Datentyp.

DATENTYP BTYPE INT64(U) UND INT128

INT64 und INT128-Datentypen werden vollständig auf MMS-Kommunikationsebene unterstützt.

In zenon werden diese per default auf den Datentyp *LINT* gespiegelt. Der Treiber ermöglicht sie auch auf Variablen vom Datentyp *LREAL* abzubilden.

Für diese Datentyp gelten folgende Einschränkungen:

- Im *LINT* Variablenwert werden nur die niedrigsten 52 Bits des Wertes übernommen. Restliche Bits werden abgeschnitten und nicht berücksichtigt.
- In *LREAL*-Variablen wird der Wert der 64-Bits übernommen, jedoch mit Verlust der Präzision, wenn der Wert größer wie 52-Bit ist.

DATEN ATTRIBUTE DES TYPES "ARRAY OF"

Kommt in einer IEC 61850 Common Data Classs (CDC), in einen der Data Attribute eine Array-Ebene vor, so wird diese Ebene für die Darstellung in zenon ans Ende der Variablenstruktur gezogen.

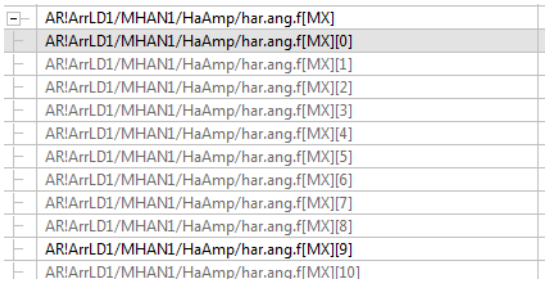
Ist das Data Attribut mit *count > 1* selbst eine Struktur (z.B. ein Vector) dann - aus einem **ARRAY OF** - entstehen im zenon Variablenmodell mehrere Arrays von skalaren Werten. Dieses Remapping

erfolgt zur Projektierungszeit beim **Offline/Online-Import** und zur Laufzeit bei der Zuordnung zwischen zenon Variablenliste und dem Objektmodell des 850-Servers.

Dadurch besteht die Einschränkung, dass nur eine Array-Ebene unterstützt wird. Auch ist damit die Unterstützung von Arrays auf die Data Attribute beschränkt, die Arrays aus Datenobjekten (aus ganzen CDC) werden ignoriert.

Beispiel

Das Datenobjekt *HaAmp* (der Common Data Class *HMV*) in Logical Node *MHAN*, beinhaltet das Data Attribut *har* - im IEC 61850-7-3 Standard definiert: *ARRAY[0..numHar] of Vector*. Dies wird in zenon zu folgendem Array:



Hinzu kommen natürlich auch die Arrays für **ang.i**, **mag.f** und **mag.i**.

Hinweis: bei **Offline/Online-Import** der Variablen, in der Spalte **Type**, werden die Arrays auch mit eckigen Klammer (*[n]*) gekennzeichnet, z.B.: *BOOL[99]*, *FLOAT32[20]*, etc.

6.4 Variablen anlegen durch Import

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

> **Info**
>
> Details zu Import und Export von Variablen finden Sie im Handbuch Import-Export im Abschnitt Variablen.

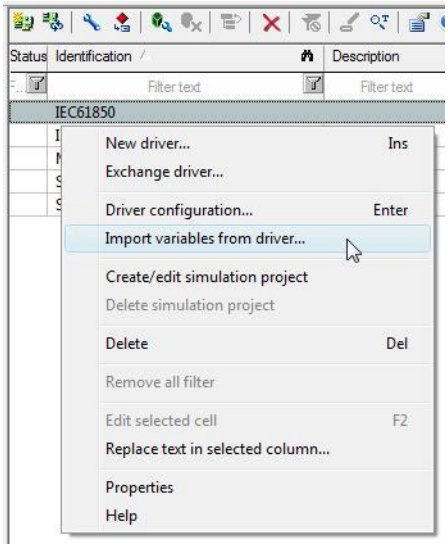
6.4.1 Online-Import von einem IEC 61850 Server

Variablen in einem zenon Projekt können in <CD_PRODUKTNAME> Editor mit dem Import des Treibers erstellt werden. Der Treiber kann das Datenmodell des IEC 61850 Server auslesen (browsen) und als Variablenliste darstellen.

Achtung

Wenn zur gleichen Zeit die Runtime auf dem gleichen Computer läuft, können die Variablen im Editor nicht von der Steuerung importiert werden, da der Treiber in der Runtime die gegebenen Einstellungen zur Kommunikation benutzt.

Sie rufen den Import über das Kontextmenü des Treibers in der Treiberliste auf.



Ein Dialog wird geöffnet:

SELECT A SERVER

Parameter	Beschreibung
Server	Liste der im Treiber angelegten Verbindungen zu IEC61850 Servern. Weitere Informationen zur Konfiguration von Serververbindungen lesen Sie im Kapitel Connections (auf Seite 28).
Source	Quelle des Datenmodelles: ▸ PLC Browsing Variablen werden von einem online verfügbaren IEC61850 Server importiert. ▸ File Browsing Variablen werden von einer SCL-Datei importiert. Weitere Informationen zur SCL-Datei lesen Sie im IEC61850-6 Standard.
OK	Übernimmt Einstellungen und schließt den Dialog.

Parameter	Beschreibung
Cancel	Verwirft alle Änderungen und schließt den Dialog.

ABLAUF FÜR PLC BROWSING

- Im sich öffnenden Dialog wählen Sie die Option PLC Browsing und wählen Sie einen Server, von dem die Variablen importiert werden können.
- Die Verbindung zum Server wird aufgebaut und die Liste der vorhandenen Datenpunkte (LDs, LNs, Datenobjekte, Datenattribute) wird gelesen.
- Wenn das Einlesen beendet ist, öffnet sich ein wichtiger Dialog, in dem Sie die gebrowsten Variablen filtern (Groß- /Kleinschreibung, z. B. *XCBR1*stVal*), auswählen (auch Mehrfachauswahl) und von der oberen zur unteren Liste hinzufügen können.
- Mit der Schaltfläche OK erstellen Sie die ausgewählten Variablen im Projekt.

> **Achtung**
>
> Der Treiber erstellt Variablen mit default Wertebereichen, z. B. ein UDINT mit <0.. 4294967295> und ein LREAL mit <-1000..1000>, die eventuell nicht den Wertebereichen der Attribute entsprechen, z. B. stVal-Attribut UDINT <0..3> oder t-Attribut LREAL<0.. 2147483647> mit drei Dezimalstellen für Millisekunden. Stellen Sie darum nach dem Import sicher, dass alle Variablen die richtigen Wertebereich-Eigenschaften haben und die String die richtigen Länge haben.

IMPORT VON ADRESSEN MIT MEHR ALS 128 ZEICHEN

Beim Auslesen (browsen) des Datenmodells generiert der Treiber die Adressen der Variablen. Dies wird in der Spalte '*Symbolic address*' angezeigt. Die generierte Adresse ist Server!ObjectReference[FC].

Ist diese Adresse länger als die für Variablenname erlaubten 128 Zeichen, wird diese auf 124 Zeichen gekürzt und mit _### ergänzt. ### steht dabei für eine laufende Nummer. Diese Nummer wird von *001* bis *999* hochgezählt.

Werden neue Variablen angelegt, deren generierter Name länger als 128 Zeichen war, wird dieser gekürzte und durch Ziffern ergänzte Name als Variablen-**Name** und **Kennung** verwendet. Der ungekürzte Adresse wird als **Symbolische Adresse** verwendet.

Achtung

Es werden in der Runtime nur dann mehr als 128 Zeichen unterstützt, wenn in der Treiberkonfiguration, im Dialog Basic settings (auf Seite 18) bei der Adressierung **Symbolic address** ausgewählt ist.

EMPFEHLUNG

Benennen Sie die Variablen nach dem Import um. Anderenfalls können bei einem erneuten Import Namenskonflikte auftreten.

"ARRAY OF" IMPORT

Durch die Einschränkung, dass nur eine Array-Ebene unterstützt (auf Seite 65) wird, ist auch die Unterstützung von Arrays auf die Data-Attribute beschränkt.

Beim Onlineimport werden diese **ARRAY OF** Variablen als zenon Arrays angelegt.

Beispiel

Das Data Attribut *HaAmp.har* der Common Data Class **MHAN** - im IEC61850 Standard definiert: *ARRAY[0..numHar] of Vector* - wird in zenon zu bis zu vier Arrays, mit jeweils gleicher Anzahl an Elementen:

- ***/HaAmp.har.ang.f[MX][0..count-1]**
- ***/HaAmp.har.ang.i[MX][0..count-1]**
- ***/HaAmp.har.mag.f[MX][0..count-1]**
- ***/HaAmp.har.mag.i[MX][0..count-1]**

SERVICE TRACKING IMPORT

Service tracking (auf Seite 111) wird nur dann angeboten, wenn ein oder mehrere Elemente mit FC = SR am Logical Device vorhanden sind. Z.B. ein oder mehrere Logical Nodes LTRK. Für Import wird nur Service tracking für Control Services automatisch angeboten. Control Services bedeutet: nur Service tracking Informationen der Kommando-Ausführung.

Ist Service tracking am Server vorhanden, wird für jedes Datenobjekt, das die Data Attribute mit FC = CO beinhaltet, ein Vorschlag zur Erstellung von zusätzlichen Service-Tracking-Variablen erstellt.

Beispiel

Datenobjekt mit Kommando-Ausführung:
S1!Device/Node/Pos/Oper.ctlVal[CO]

Diese zusätzlichen Variablen existieren nicht am Server, sondern werden zusätzlich im jeweiligen Datenobjekt mit möglicher Kommando-Ausführung angelegt.

Aus dieser Vorschlagsliste können jene Variablen ausgewählt werden, welche tatsächlich als zusätzliche Service-Tracking-Variable angelegt werden.
Dieser Vorschlag setzt sich im Namen wie folgt zusammen:
Datenobjektreferenz_Servicetracking-Datenattributname[SR]. Als Servicetracking-Datenattributname werden die Datenatribute vorgeschlagen, welche in dem Logical Node LRTK für Control Services vorhanden sind.

Beispiel

Benennung: S1!Device/Node/Pos_errorCode[SR].

Der Datentyp einer zusätzlichen Service-Tracking-Variable entspricht dem Datentyp des SR Data Attributes, der in der Logical Node LRTK vorhanden ist.
Der Treiberobjekttyp ist Service tracking.

Zusätzlich wird auch eine String-Variable *_allCTS[SR] angeboten. Diese String-Variable beinhaltet die komplette Service tracking (auf Seite 111) Information und kann mit Hilfe von zenon Logic oder VBA/VSTA ausgewertet werden.

GETNAMELIST ON DO

Die Option *GetNameList on DO* von Versionen vor 8.00 wird im IEC850 Treiber ignoriert. Ab zenon Version 8.00 schickt der Treiber für jeden Knoten immer *getVariableAccessAttributes* an den Server. Erhält der Treiber vom Server eine negative Antwort (response), wird für den entsprechenden Knoten (Node) eine nochmalige Anfrage auf einer tieferen Ebene an den Server gesendet. Sollte dies erneute Anfrage ebenfalls fehlschlagen, wird auf Ebene des DataObject verfeinert.

Im Fehlerfall wird ein entsprechender LOG-Eintrag erstellt: *Failed to retrieve variable information for <Failed Request Name>*

Für den Online-Import von Variablen gilt:

- Das Online-Auslesen der Variablen berücksichtigt nur die oberste Ebene von einem *DataObject*.
  Ist diese Vertiefung nicht ausreichend, importieren Sie die Variablen von einer SCL-Datei.
- In der Runtime wird bis zur Ebene von *DataAttribute* verfeinert.
- Kann das *DataAttribute* nicht gelesen werden, bleibt die Variable in der Runtime ohne Werte und erhält den Status *Invalid*.

Beim Online-Import von Variablen im zenon Editor gilt:

- der Knoten muss vollständig erfolgreich ausgelesen werden können, damit die im Knoten enthaltenen Variablen im Auswahldialog angeboten werden.

Hinweis: Wird das zenon Projekt für eine Version vor 8.00 kompiliert, wird ein entsprechender Eintrag in der Konfigurationsdatei weiter berücksichtigt.

6.4.2 Offline-Import von einer SCL-Datei

Variablen für ein Projekt können in <CD_PRODUKTNAME> Editor mit dem Import des Treibers erstellt werden. Der Treiber kann das Datenmodell aus einer SCL-Datei auslesen und als Variablenliste darstellen.

Hinweis: Für vereinfachten Import der Variablen aus der Datasets in RCBs liefert der **IEC850 Driver Configuration** Wizard zusätzliche Möglichkeiten. Der Wizard ist nur in **Energy Edition** erhältlich.

Für Offline-Import werden folgenden Dateiendungen akzeptiert:

- **.scl*
- **.scd*
- **.cid*
- **.icd*
- **.xml*, wenn Inhalt der Datei nach Regeln des IED61850-6 Standards definiert wurde.

Der Treiber importiert offline nur DOI-Elemente. Die sogenannten Control Blocks , z.B. *ReportControl* (Report Control Block) und *SettingControl* (Setting Control Block) werden beim Offline-Import nicht unterstützt. Sie können die Variablen für diese Attribute über Online Import oder manuell erstellen.

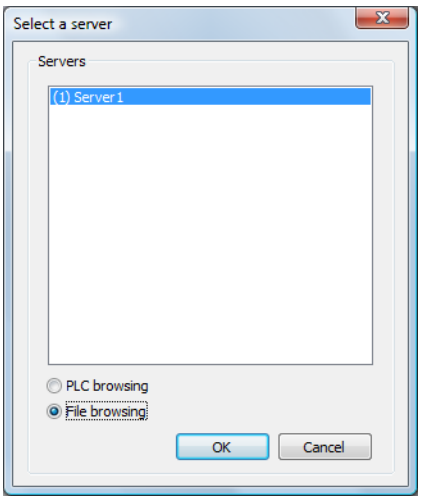
Achtung

Der Treiber kann vom Datenmodell des IEDs nur dann importieren, wenn das Datenmodell mit dem IEC61850 Standard konform ist. Alle nicht konforme Elemente werden dann ignoriert und im Import-Dialog nicht zum Auswahl stehen. Zum Beispiel der Treiber ignoriert die Logical Nodes, die direkt zum IED Knoten - anstatt zum Logical Device Knoten - verlinkt sind.

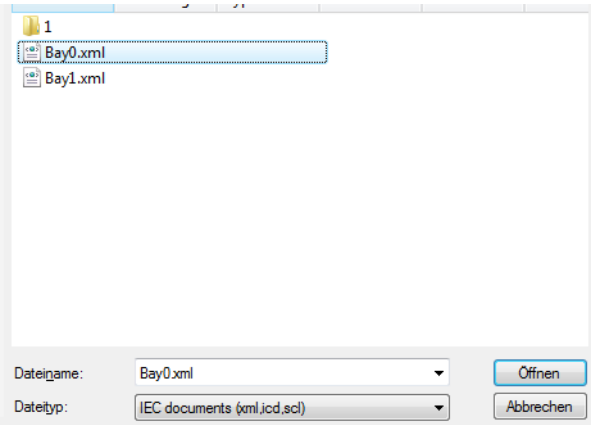
Um Variablen aus einer Datei zu importieren:

1. Sie rufen den Import über das Kontextmenü des Treibers in der Treiberliste auf.

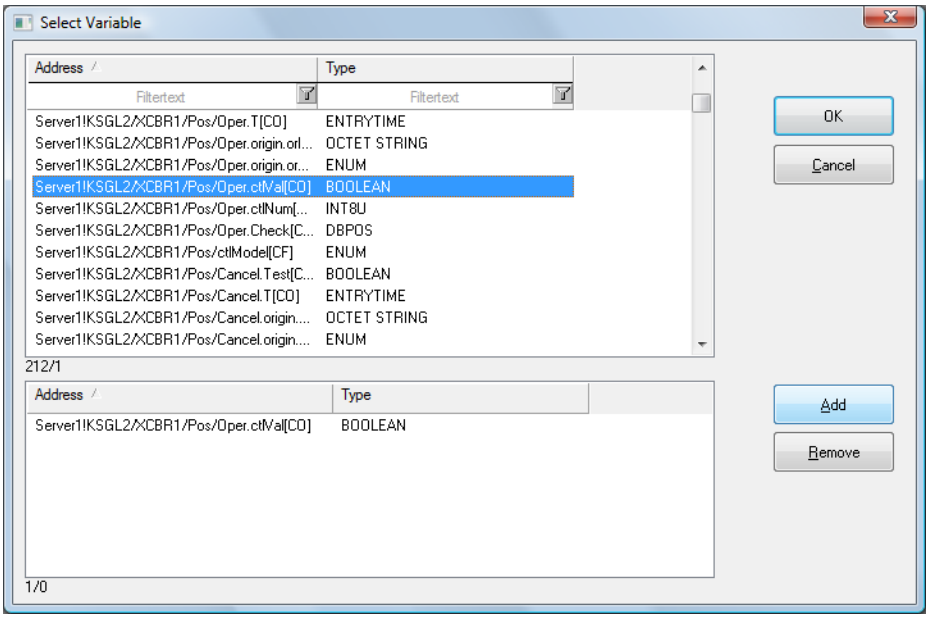
2. wählen Sie bei der Serverauswahl die Option **File browsing**



3. wählen Sie dann die Import-Datei aus.



4. die folgenden Schritte für die Auswahl der Variablen sind analog zum Online-Import.



IMPORT VON ADRESSEN MIT MEHR ALS 128 ZEICHEN

Beim Auslesen (browsen) des Datenmodells generiert der Treiber die Adressen der Variablen. Dies wird in der Spalte '*Symbolic address*' angezeigt. Die generierte Adresse ist Server!ObjectReference[FC].

Ist diese Adresse länger als die für Variablenname erlaubten 128 Zeichen, wird diese auf 124 Zeichen gekürzt und mit _### ergänzt. ### steht dabei für eine laufende Nummer. Diese Nummer wird von *001* bis *999* hochgezählt.

Werden neue Variablen angelegt, deren generierter Name länger als 128 Zeichen war, wird dieser gekürzte und durch Ziffern ergänzte Name als Variablen-**Name** und **Kennung** verwendet. Der ungekürzte Adresse wird als **Symbolische Adresse** verwendet.

> **Achtung**
>
> Es werden in der Runtime nur dann mehr als 128 Zeichen unterstützt, wenn in der Treiberkonfiguration, im Dialog Basic settings (auf Seite 18) bei der Adressierung **Symbolic address** ausgewählt ist.

EMPFEHLUNG

Benennen Sie die Variablen nach dem Import um. Anderenfalls können bei einem erneuten Import Namenskonflikte auftreten.

SERVICE TRACKING IMPORT

Service tracking wird nur dann angeboten, wenn ein oder mehrere Elemente mit FC = SR am Logical Device vorhanden sind. Z.B. ein oder mehrere Logical Nodes LTRK. Für Import wird nur Service tracking für Control Services automatisch angeboten. Control Services bedeutet: nur Service tracking Informationen der Kommando-Ausführung.

Ist Service tracking in der SCL-Datei vorhanden, wird für jedes Datenobjekt, das die Data Attribute mit FC = CO beinhaltet, ein Vorschlag zur Erstellung von zusätzlichen Service-Tracking-Variablen erstellt.

> **Beispiel**
>
> **Datenobjekt mit Kommando-Ausführung:**
> S1!Device/Node/Pos/Oper.ctlVal[CO]

Diese zusätzlichen Variablen existieren nicht am Server, sondern werden zusätzlich im jeweiligen Datenobjekt mit möglicher Kommando-Ausführung angelegt.

Aus dieser Vorschlagsliste können jene Variablen ausgewählt werden, welche tatsächlich als zusätzliche Service-Tracking-Variable angelegt werden.
Dieser Vorschlag setzt sich im Namen wie folgt zusammen:
Datenobjektreferenz_Servicetracking-Datenattributname[SR]. Als Servicetracking-Datenattributname

werden die Datenatribute vorgeschlagen, welche in dem Logical Node LRTK für Control Services vorhanden sind.

Beispiel

Benennung: S1!Device/Node/Pos_errorCode[SR]

Der Datentyp einer zusätzlichen Service-Tracking-Variable entspricht dem Datentyp des SR Data Attributes, der in der Logical Node LRTK vorhanden ist.
Der Treiberobjekttyp ist Service tracking.

Zusätzlich wird auch eine String-Variable **_allCTS[SR]* angeboten. Diese String-Variable beinhaltet die komplette Service tracking Information und kann mit Hilfe von zenon Logic oder VBA/VSTA ausgewertet werden.

6.4.3 XML Import

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

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

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

VORAUSSETZUNGEN

Beim Import gelten folgende Bedingungen:

- **Abwärtskompatibilität**

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

- **Konsistenz**

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

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

- **Struktur-Datentypen**

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

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

6.4.4 DBF Import/Export

Daten können nach dBase exportiert und aus dBase importiert werden.

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

IMPORT DBF-DATEI

Um den Import zu starten:

1. Führen Sie einen Rechtsklick auf die Variablenliste aus.
2. Wählen Sie in der Dropdownliste von **Erweiterter Export/Import** ... den Befehl **dBase importieren**.
3. Folgen Sie den Anweisungen des Importassistenten.

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

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

EXPORT DBF-DATEI

Um den Export zu starten:

1. Führen Sie einen Rechtsklick auf die Variablenliste aus.
2. Wählen Sie im Dropdownliste von **Erweiterter Export/Import** ... den Befehl **dBase exportieren...** .
3. Folgen Sie den Anweisungen des Exportassistenten.

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

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

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

DATEIAUFBAU DER DBASE EXPORTDATEI

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

Achtung

dBase unterstützt keine Strukturen oder Arrays (komplexe Variablen).

DBF-Dateien müssen:

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

STRUKTUR

Bezeichnung	Typ	Feldgröße	Bemerkung
KANALNAME	Char	128	Variablenname. Länge kann über den Eintrag **MAX_LAENGE** in der **project.ini** eingeschränkt werden.
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)

Bezeichnung	Typ	Feldgröße	Bemerkung
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,

Bezeichnung	Typ	Feldgröße	Bemerkung
			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

Bezeichnung	Typ	Feldgröße	Bemerkung
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.

6.5 Kommunikationsdetails (Treibervariablen)

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

- Information
- Konfiguration
- Statistik und
- Fehlermeldungen

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

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

Info

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

Zum Beispiel:

- Variablen für Modem-Informationen werden nur von modemfähigen Treibern unterstützt.
- Treibervariablen für den Polling-Zyklus stehen nur für rein pollende Treiber zur Verfügung.
- Verbindungsbezogene Informationen wie **ErrorMSG** werden nur von Treibern unterstützt, die zu einem Zeitpunkt nur eine Verbindung bearbeiten.

INFORMATION

Name aus Import	Typ	Offset	Erklärung
MainVersion	*UINT*	0	Haupt-Versionsnummer des Treibers.
SubVersion	*UINT*	1	Sub-Versionsnummer des Treibers.
BuildVersion	*UINT*	29	Build-Versionsnummer des Treibers.
RTMajor	*UINT*	49	zenon Hauptversionsnummer
RTMinor	*UINT*	50	zenon Sub-Versionsnummer
RTSp	*UINT*	51	zenon Service Pack-Nummer
RTBuild	*UINT*	52	zenon Buildnummer
LineStateIdle	*BOOL*	24.0	TRUE, wenn die Modemleitung belegt ist.
LineStateOffering	*BOOL*	24.1	TRUE, wenn ein Anruf rein kommt.
LineStateAccepted	*BOOL*	24.2	Der Anruf wird angenommen.
LineStateDialtone	*BOOL*	24.3	Rufton wurde erkannt.
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.

Name aus Import	Typ	Offset	Erklärung
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

Name aus Import	Typ	Offset	Erklärung
			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.

Name aus Import	Typ	Offset	Erklärung
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

Name aus Import	Typ	Offset	Erklärung
			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.

Name aus Import	Typ	Offset	Erklärung
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.

7 Funktion Treiberkommandos

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

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

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

Achtung

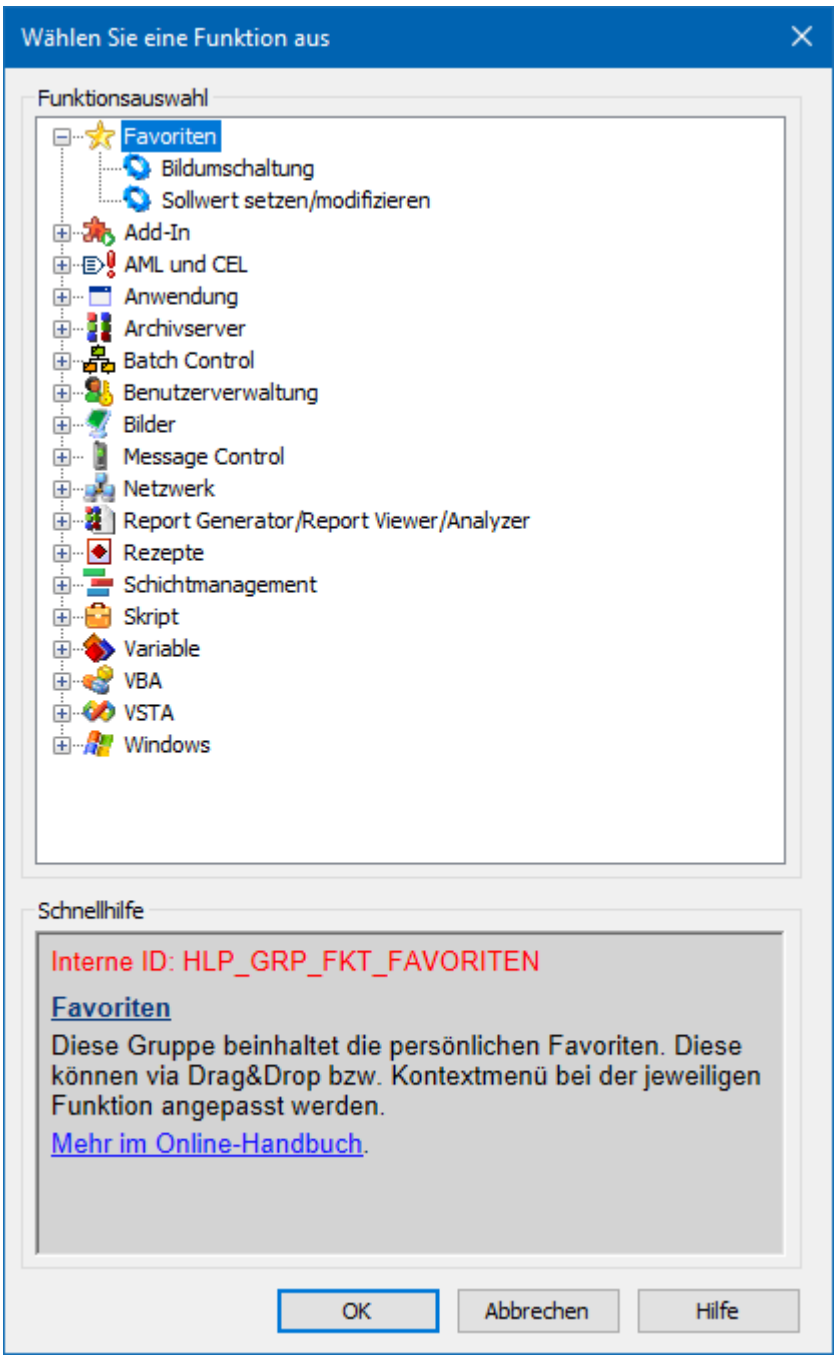
Die zenon Funktion **Treiberkommandos** ist nicht ident mit den Treiberkommandos, die bei Energy-Treibern in der Runtime ausgeführt werden können!

PROJEKTIERUNG DER FUNKTION

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

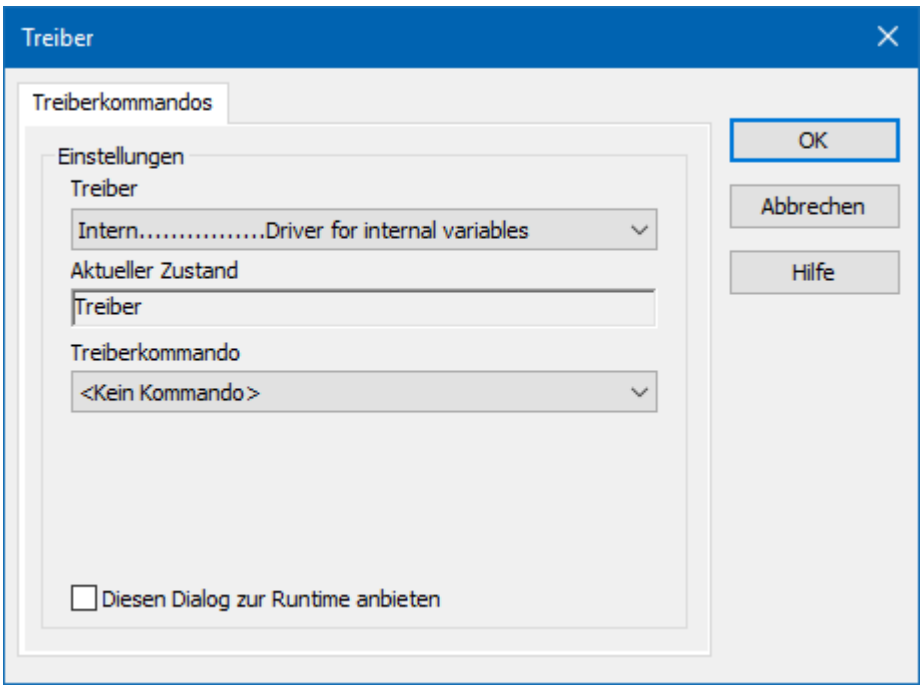
1. Legen Sie im zenon Editor eine neue Funktion an.

Der Dialog zur Auswahl einer Funktion wird geöffnet.



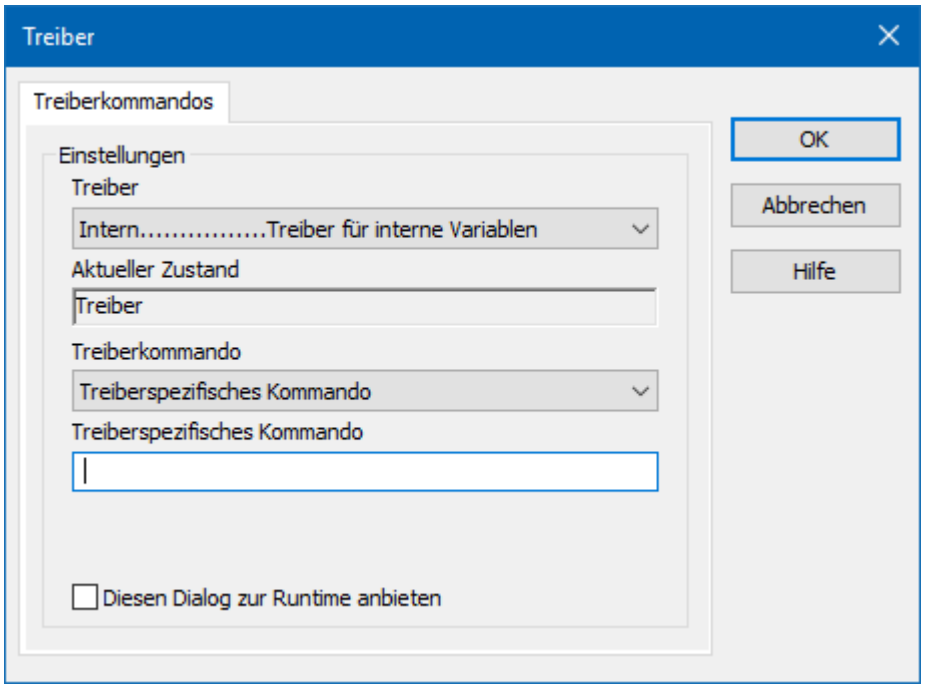
2. Navigieren Sie zum Knoten **Variable.**
3. Wählen Sie den Eintrag **Treiberkommandos**.

Der Dialog zur Konfiguration wird geöffnet.



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

DIALOG TREIBERKOMMANDOS



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

DIALOG BEENDEN

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

VERFÜGBARE TREIBERKOMMANDOS

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

Treiberkommando	Beschreibung
Kein Kommando	Es wird kein Kommando gesendet. Damit kann auch ein bereits bestehendes Kommando aus einer projektierten Funktion entfernt werden.
Treiber starten (Onlinemodus)	Treiber wird neu initialisiert und gestartet. **Hinweis:** Wenn der Treiber bereits gestartet wurde, muss er gestoppt werden. Erst dann kann der Treiber wieder neu initialisiert und gestartet werden.
Treiber stoppen (Offlinemodus)	Treiber wird angehalten, es werden keine neuen Daten angenommen. **Hinweis:** Ist der Treiber im Offline-Modus, erhalten alle Variablen, die für diesem Treiber angelegt wurden, den Status *Abgeschaltet* (*OFF*; Bit *20*).
Treiber in Simulationsmodus	Treiber wird in den Simulationsmodus gesetzt. Die Werte aller Variablen des Treibers werden vom Treiber simuliert. Es werden keine Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
Treiber in Hardwaremodus	Treiber wird in den Hardwaremodus gesetzt. Für die Variablen des Treibers werden die Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
Treiberspezifisches Kommando	Eingabe eines treiberspezifischen Kommandos. Öffnet Eingabefeld für die Eingabe eines Kommandos.
Treiber Sollwertsetzen aktivieren	Sollwert setzen auf Treiber ist möglich.
Treiber Sollwertsetzen deaktivieren	Sollwert setzen auf Treiber wird verhindert.
Verbindung mit Modem aufbauen	Verbindung aufbauen (für Modem-Treiber). Öffnet Eingabefelder für Hardware-Adresse und Eingabe der zu wählenden Nummer.
Verbindung mit Modem trennen	Verbindung beenden (für Modem-Treiber).
Treiber in Simulationsmodus zählend	Treiber wird in den zählenden Simulationsmodus gesetzt. Alle Werte werden mit *0* initialisiert und in der eingestellten Updatezeit jeweils um *1* bis zum Maximalwert inkrementiert und beginnen dann wieder

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

FUNKTION TREIBERKOMMANDOS IM NETZWERK

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

- Ein spezielles Netzwerkkommando wird vom Rechner zum Server des Projekts gesendet. Dieser führt dann die gewünschte Aktion auf seinem Treiber aus.
- Zusätzlich sendet der Server das gleiche Treiberkommando zum Standby des Projekts. Der Standby führt die Aktion auch auf seinem Treiber aus.

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

7.1 SwitchConnection

Das treiberspezifischen Kommando *SwitchConnection* schaltet zwischen primärer und sekundärer IP-Adresse um.

Ablauf:

1. Mit dem Kommando wird für die eingegebene **Netzadresse** die aktive IP-Adresse geschlossen.
2. Für die Variablen wird dabei das **INVALID**-Bit nicht gesetzt.
3. Dann wird die Verbindung über die andere IP-Adresse aufgebaut.
4. Falls der Verbindungsaufbau fehlschlägt, wird das **INVALID**-Bit gesetzt und es wird zur vorher aktiven Verbindung zurückgeschaltet.

Hinweis: Das Treiberkommando wird nur berücksichtigt, wenn die Verbindung zu einer IP Adresse erfolgreich aufgebaut ist. Besteht keine Verbindung oder läuft gerade ein Verbindungsaufbau, wird das Kommando ignoriert.

Syntax:

- *SwitchConnection <NetAddr>*
  <NetAddr> steht für die Nummer der **Netzadresse**.

Beispiel:

- **SwitchConnection 1**
  Schaltet für die Verbindung mit Netzadresse *1* zwischen primärer und sekundärer IP-Adresse um.

8 IEC850 Client Funktionen

Spezielle Spezifikationen des IEC 61850 Standard wurden als spezifische Funktionen im IEC850 Client Treiber implementiert.

Detaillierte Informationen erhalten Sie in den jeweiligen Kapitel:

- Verbindungaufbau und Verbindungsausfallerkennung (auf Seite 98)
- Befehle (Control Model) (auf Seite 101)

  Befehle - Übersicht

  - Select und Cancel (auf Seite 104)
  - Additional Cause Diagnosis (auf Seite 106)
  - Serviceparameter des Befehls (auf Seite 108)
- Service Tracking (auf Seite 111)
- Mapping von Double Point Values (auf Seite 115)
- Quality Bits, Zeitstempel und Statusbits der Variable (auf Seite 118)
  - TimeQuality und TimeAccuracy (auf Seite 121)
- Filetransfer (auf Seite 122)

  Für den Transfer der Datein sind drei Funktionen implementiert:

  - Ordnerinformation anfordern
  - Datei vom Server holen
  - Datei löschen

EINSCHRÄNKUNGEN

- Der Treiber unterstützt auch die auf **IEC 61400-25-4 Annex C** (von 2008-08) basierende Steuerungen, vorausgesetzt sie sind mit wenigstens Edition 1 des **IEC 61850** Standards kompatibel, mit berücksichtigung der **TISSUE**s (Technical Issues) die bis 2008-08 die Edition 1 korregiert haben.

- Datentyp: **ARRAY OF**
  Es wird nur eine Array-Ebene unterstützt. Die Unterstützung von Arrays ist auf die Datenattribute beschränkt. Das Beschreiben von Array-Elementen ist nicht möglich. Beim Lesen via Polling oder Reporting wird, wie bei allen anderen Datenattributen auch, grundsätzlich immer das gesamte Data Object gelesen (FCD-Referenz), das das Array-Attribut beinhaltet.

Datentyp **INT64(U)**:
Im (U)LINT Variablenwert werden nur die niedrigsten 52 Bits übernommen.
Restliche Bits werden abgeschnitten und nicht berücksichtigt.

- Alternativ kann die Variable mit dem Datentyp LREAL projektiert werden. Dadurch ist es möglich, den Gesamtwert zu übertragen.
  Beachten Sie dabei, dass für Werte die größer als 52 Bit sind, die Präzision der niedrigsten Stellen verloren geht.

- Beim Reports mit Service Tracking werden ausschließlich DataSets mit FCD-Referenzen unterstützt.

- **Conformance statement** (PICS, MICS):
  Die Informationen über unterstützte oder benötigte Dienste und Datentypen erhalten Sie im Anhang C (auf Seite 173).

8.1 Verbindungaufbau und Verbindungsausfallerkennung

Beim Start der zenon Runtime startet auch der Treiber und beginnt die Verbindungen zu IEDs herzustellen. Der Treiber öffnet nur dann eine Verbindung zu einem Logical Device, wenn mindestens eine Variable aus dem LD im Projekt angelegt ist. Der Treiber ignoriert aus Performancegründen auch die logische Knoten (Logical Nodes) für die keine Variablen im Projekt angelegt sind. **Ausnahme:** LLN0 - der Inhalt von LLN0 (seine RCBs) wird immer eingelesen.

Auf Windows Betriebssystemen kann die Information über das TCP-Timeout (20-30s) von Betriebssystem nur beim Senden (nicht beim Empfangen) zur Vergfügung gestellt werden. Somit kann der Treiber einen Verbindungsabbruch nicht erkennen, wenn alle Variablen per Reports empfangen werden. Nach einem Kommunikationsabbruch (z. B. Netzwerkausfall) wird die Verbindung zum Logical Device nur dann automatisch wiederhergestellt, wenn mindestens eine Variable des LD gerade gepollt wird.

Empfehlung: Um einen Verbindungsausfall zu erkennen, aktivieren Sie in der Treiberkonfiguration die Eigenschaft **Automatic Watchdog**.

Wenn **Automatic Watchdog** aktiviert wurde: der Treiber fängt automatisch an das Datenattribut **/LLN0/Mod/stVal[ST]* zu pollen sobald alle andere Daten des Logical Device nur noch über Reports ankommen, also während alle gepollte Variablen abgemeldet sind. Der Treiber pollt das Datenattribut unabhängig davon ob es in einen der RCBs vorhanden wäre. Die zyklisch, im Tackt der

Treibereinstellung **Polling rate** gesendete MMS.Read-Anfragen, zwingen das Betriebssystem die TCP-Verbindung erneut zu überprüfen und dem Treiber melden, wenn die Verbindung abgebrochen wurde.

TCP KEEP_ALIVE

Der Treiber öffnet in Windows das TCP Socket mit folgenden Keep-Alive-Parameter:

- Das Timeout für *keep_alive* beträgt 20 Sekunden.
- Das Intervall für *keep_alive* beträgt 1 Sekunde.
- Wiederholungen (*Retries*) sind vom Betriebssystem fix mit 10 Wiederholungen vorgegeben.
- Dies bedeutet:
  Sendet der Treiber innerhalb von 20 Sekunden kein TCP Telegramm (kein *MMS request*), so sendet das Windows Betriebssystem jede Sekunde eine TCP *keep_alive* Nachricht. Wird für 10 solcher *keep_alive* Nachrichten keine Antwort empfangen, wird die Verbindung zum 850-Server geschlossen.

ABBRUCH BEI AUSSTEHENDEN MMS REQUESTS

Erhält der IEC850 Treiber für die gesendeten *MMS requests* keine Antworten vom 850-Server, so bleiben diese ausstehend bis die Anzahl von *negociatedMaxServOutstandingCalled* (von *MMS Initiate*) erreicht ist. Wird dadurch der Treiber länger als 50 Sekunden blockiert, wird die Verbindung zum Server automatisch abgebrochen.
Dadurch wird verhindert, das ein fehlerhaftes IED die Kommunikationsprozesse des Treibers blockiert.
Dieser Abbruch erzeugt in der LOG-Datei folgende Fehlermeldung:
Outstanding requests are not answered for more than 50s

CONNECTION STATE-VARIABLE

Eine *Connection State*-Variable, wenn angelegt, gibt die Auskunft darüber, ob der Treiber eine TCP- und dann MMS-Verbindung zum Server hat und ob die Verbindung via primärer oder sekundärer IP-Adressen besteht. Weiters erhalten Sie darüber die Information, ob alle projektierten statischen RCB-Zuweisungen erfolgreich angemeldet worden sind.

Diese Variable muss mit dem **Treiberobjekttyp** (auf Seite 62) *Connection state* angelegt werden. Sie muss die korrekte **Netzadresse** der Verbindung beinhalten, die Referenz mit Syntax **!ConnectionState* haben und vom Datentyp **UDINT** sein.

Die Bits des Variablenwertes haben folgende Bedeutung:

Für die primäre TCP/IP Verbindung zum 850-Server:

Bit	Bedeutung	Wert (hex)
1	*TCP_CONNECTED*	0x02

Bit	Bedeutung	Wert (hex)
2	*TCP_CONNECTING*	0x04
3	*TCP_CONNECT_FAILED*	0x08
16	*MMS_ASSOCIATED*	0x10000
17	*MMS_RCB_ENABLE_FAILED*	0x20000

Für die sekundäre TCP/IP Verbindung:

Bit	Bedeutung	Wert (hex)
5	*TCP_CONNECTED*	0x20
6	*TCP_CONNECTING*	0x40
7	*TCP_CONNECT_FAILED*	0x80
24	*MMS_ASSOCIATED*	0x1000000
25	*MMS_RCB_ENABLE_FAILED*	0x2000000

Das Bit *MMS_RCB_ENABLE_FAILED* ist nur dann gesetzt, wenn einer (oder mehr) RCBs die in der Treiberkonfiguration in **RCB assignment** (auf Seite 47) konfiguriert wurde nicht aktiviert werden konnte. Dies geschieht z.B. dann, wenn der 850-Server das Schreiben auf RCB Datenattribute verweigert, da z.B. ein anderer Client diesen Report bereits nutzt.

Die *Connection State*-Variable gibt keine Auswertung darüber, ob es möglich war die URCBs über '**max. auto used URCBs** (auf Seite 29)' zu aktivieren.

Beispiel

Die Standardabfolge für den Wertänderung der Variable ist:

Start der Runtime (o. TCP_CONNECT_FAILED) => TCP_CONNECTING => TCP_CONNECTED => TCP_CONNECTED + MMS_ASSOCIATED

Oder, bei dauerhaften TCP-Fehler:

TCP_CONNECT_FAILED => TCP_CONNECTING => TCP_CONNECT_FAILED

Oder, wenn Verbindung korrekt hergestellt wurde, jedoch Aktivieren eines RCB fehlschlägt:

=> TCP_CONNECTED + MMS_ ASSOCIATED + MMS_RCB_ENABLE_FAILED

Beim Start der zenon Runtime ist es möglich dass die TCP Verbindungen zu manchen IEDs gelingen dem Treiber noch bevor die Runtime für die Auswertung der *Connection State*-Variablen bereit wird;

während die Variablen in der Runtime noch nicht angemeldet sind. In diesem Fall erst *TCP_CONNECTED*, bzw. *MMS_ASSOCIATED* Bits werden ausgewertet (und sichtbar z. B. in **CEL**).

REDUNDANTES ZENON NETZWERK

In redundanten zenon Netzwerk, sowohl der **Prozessführende Server** und der **Standby Server** starten beide eigenen IEC850 Treiber. Die beide Treiber sollen unterschiedliche RCB Instanzen anmelden (sonst stehen sie im Konflikt weil ein 850-Server nur einem Client die RCB Instanz vergeben darf).

Achtung: Wenn Sie in Ihrem zenon Projekt Redundanz verwenden (Eigenschaft **Netzwerk aktiv** aktiviert sowie **Server 1** und **Server 2** konfiguriert), müssen Sie in der Treiberkonfiguration - **RCB assiggment** - wenigstens zwei Einträge konfigurieren: einen **Hostname** für den **Server 1** und zweiten **Hostname** für den **Server 2** (Standby). Für diese Einträge müssen Sie dann unterschiedliche Instanzen der RCBs auswählen.
Hinweis: Per Default werden vom IEC850 Treiber bereits zwei Hostnamen angelegt: *$SCADA_SERVER1* und *$SCADA_SERVER2*.

Die *Connection State*-Variable kann für Treiber am Netzwerk **Server 1** andere Auswertung ergeben als am **Server 2**. Für dem Benutzer, auch am Standby, ist nur sichtbar der aktuelle Wert des aktuellen **Prozessführenden Servers**. Wenn Sie auch die aktuelle Auswertung für **Standby Server** sichtbar machen möchten - um z.B. Alarmieren falls der Standby seine RCBs nicht anmelden konnte, können Sie eine Kopie der *Connection State*-Variable erstellen - mit gleicher Referenz (in **Symbolische Adresse**), anderem Präfix in der Variableneigenschaft **Name** und mit aktivierter Eigenschaft **Nur von Standby Server anfordern**.

Hinweis: Für **Bewertetes Netzwerk** brauchen Sie keine von Standby angeforderte Variablen in der Eigenschaft **Bewertungen**. Die Bewertung findet am beiden Server separat statt, anhand der lokalen Werten der Variablen. Also am Standby wird immer der Wert der *Connection State*-Variable von seinen internen Abbild ausgewertet, auch wenn im Projekt keine Kopie erstellt wurde.

VERBINDUNGSAUFBAU MIT IED MIT KLEINER PDU-GRÖßE

Der Treiber liest in der zenon Runtime automatisch bei der *Association* (Verbindungsaufbau) das Datenmodell von einem IED aus. Verwendet das IED eine kleinere TCP Framegröße wie für das Auslesen des Datenmodells notwendig ist, wird dies automatisch erkannt und die Prozedur dynamisch angepasst. Basis sind die im zenon Projekt erstellten Variablen.
Für die Erstellung der Variablen im zenon Editor gilt: Kann das Datenmodell online nicht vollständig von IED importiert werden, verwenden Sie den Offline-Import aus einer SCL-Datei.

8.2 Befehle (Control Model)

Der Treiber unterstützt die Befehle 'Direct Operate' und 'Select Before Operate' (SBO) mit normaler und erhöhter Sicherheit (normal security und enhanced security) - also alle von IEC 61850 Standard vorgesehene Control Models *1...4*.

Um einen Befehl auszulösen, in beliebigen Control Model, setzen Sie einen Wert auf die Variable mit der Referenz **/Oper.ctlVal** , d.H:

- **/Oper.cltVal[CO]* für Befehle wo in Oper Struktur das Item ctlVal ein einfaches bType ist.
- **/Oper.ctlVal.f[CO]* oder **/Oper.ctlVal.i[CO]* für Befehle wo cltVal Item selbst auch eine Struktur ist, z.B. ein AnalogueValue.

Der Treiber überprüft dann am 850-Server das tatsächliche Control Model des Datenobjektes und führt eine vollständige Kommandosequenz entsprechend aus. Für den anstehende Befehl werden die erforderlichen Kommandosequenzen vom Treiber automatisch in der richtigen Reihenfolge ausgeführt.

Beispiel: Wenn der Benutzer der Wert der Variable mit einer Referenz **/CSWI1/Pos/Oper.ctlVal[CO]* ändert, führt der Treiber zuerst ein Select und erst anschließend ein Operate durch, oder nur ein Operate. Die Entscheidung ob ein Select erforderlich ist trifft der Treiber selbst. Dafür sind keine weiteren Konfigurationsschritte notwendig.

Achtung

Legen Sie im Projekt keine Variablen an, die den Datenattributen */SBO, */SBOw* oder */Cancel* entsprechen.

Um einen Befehl an den Server zu senden, setzen Sie ausschließlich den Wert der Variable */Oper.ctlVal*[CO] ab. Der Treiber wird automatisch erkennen, dass die Kommandosequenz auch ein Select beinhalten muss. Ein Select wird dann automatisch ausgeführt.

DATENATTRIBUTE MIT FUNCTIONAL CONSTRAINT CO

Mit Functional Constraint *CO* werden Variablen nur mit folgenden Referenzen unterstützt:

- **/Oper.ctlVal[CO]*, **/Oper.ctlVal.i[CO]* und **/Oper.ctlVal.f[CO]* - es sind die Auslöser eines Befehls
- **/Oper.Test[CO]* und **/Oper.Check[CO]* - es sind optionale Serviceparameters des Befehls, der mit Sollwertsetzen auf *ctlVal* ausgelöst wurde.

In der Runtime werden nur die Werte der *CO* Variablen *Oper.ctlVal**, *Oper.Check* und *Oper.Test* angezeigt, andere zenon Variablen mit Functional Constraint *CO*, auch wenn projektiert, werden vom Treiber ignoriert. Der Treiber unterstützt restliche *CO* Datenattribute in der *Oper*, *SBOw* und *Cancel* Strukturen ausschließlich für die Kommunikation mit einem 850-Server, nicht für Datenaustausch mit der Runtime.

In der Runtime, sind die Variablen mit Functional Constraint *CO* ausschließlich 'write-only'. Das bedeutet:

- Diese haben erst dann einen Wert, nachdem dieser in der Runtime - vom Benutzer - gesetzt wurde.

- Nach dem Start der Runtime haben diese Variablen noch keinen Wert.
- Nach Verbindungsausfall und Wiederverbindung sind sie auf Wert 0 zurückgesetzt.

ABLAUF NACH SOLLWERTSETZEN AUF OPER.CTLVAL

Wenn ein Wert auf eine Projektvariable gesetzt ist, die dem Item *ctlVal* in einem Struktur-Datenattribut mit dem Namen *Oper* entspricht und es die Functional Constraint *CO* oder *SP* (**/Oper.ctlVal[CO|SP]*, **/Oper.ctlVal.i[CO]* oder **/Oper.ctlVal.f[CO]*) hat, überprüft der Treiber automatisch welches Control Model für einen Befehl in diesem Data Object (z.B. *Pos* wenn die Referenz **/Pos/Oper.ctlVal[CO]* ist) benötigt wird.
Dafür wird zuerst das Datenattribut **/ctlModel[CF]* des Datenobjekts vom 850-Server ausgelesen (also: **/Pos/ctlModel[CF]*). Abhängig vom Wert dieses Datenattributs, wird der jeweilige Control Model benutzt und der Befehl im entsprechenden Ablauf ausgeführt. Das Control Model:

- *Wert 0* - status only*:*
  keine Aktion wird ausgeführt. Es ist laut IEC 61850 Standard verboten ein Befehl an 850-Server zu senden.
- *Wert 1* - direct control with normal security*:*
  ein 'Operate' Dienst wird ausgeführt - der Wert wird in die **/Oper[CO]* Struktur geschieben (und die Struktur an 850-Server gesendet).
- *Wert 2* - select before operate with normal security*:*
  die Dienste 'Select' und 'Operate' werden ausgeführt - lesen von **/SBO[CO]*, schreiben auf **/Oper[CO]*.
- *Wert 3* - direct control with enhanced security*:*
  ein 'Operate' Dienst wird ausgeführt - der Wert wird auf **/Oper[CO]* geschieben.
- *Wert 4* - select before operate with enhanced security*:*
  die Dienste 'SelectWithValue' und 'Operate' werden ausgeführt - schreiben auf **/SBOw[CO]*, schreiben auf **/Oper[CO]*.

Die geschriebene zum 850-Server **/Oper[CO]* und **/SBOw[CO]* sind Strukturen mit mehreren Datenattributen. Der Treiber befüllt sie automatisch, weitere Informationen dazu finden Sie im Kapitel Serviceparameter des Befehls (auf Seite 108).

Info

Der Treiber liest immer das *ctlModel* Datenattribut erneut - bevor jeden Dienst - vom 850-Server, um den Befehl korrekt auszuführen. Der Treiber überträgt den **ctlModel** Wert aber nicht automatisch zu der Runtime. Auch dann nicht, wenn für das **ctlModel** eine Variable erstellt wurde. Wenn eine Variable erstellt wurde, wird sie gemäß der **Polling Rate**, die in der Treiberkonfiguration definiert ist, aktualisiert.

BEFEHLE FÜR ANALOGWERTE

Handelt es sich um einen Befehl auf einen Analogwert - **/Oper.ctlVal.i[CO]* oder **/Oper.ctlVal.f[CO]* - überprüft der Treiber, ob das jeweilige Gegenstück - *f* bzw. *i* - zum gesetzten Wert im Datenmodell des Servers existiert.

Sind (trotz des **TISSUE 925**) beide Werte im Objekt vorhanden, wird der jeweils andere Wert anhand der zuvor gelesenen Konfiguration des Datenobjektes mit der Formel *f = ((i * scaleFactor) + offset) / 10^unitsMultiplier* (**Anmerkung:** *10* hoch *unitsMultiplier*) automatisch berechnet. Anschließend werden beide Werte (*i* und *f*) als Struktur in einem Befehl versendet. Dafür sind keine Konfigurationsschritte erforderlich.

Hinweis: das **TISSUE 925** für IEC 61850 Standard Edition 2 hat eingeschränkt, dass ein *Oper* mit *AnalogueValue* entweder Item *f* oder *i* beinhalten darf, nicht beide. Somit nur ältere Geräte können eventuell ein nicht mehr Standard konformes Datenmodell haben - mit *Oper.ctlVal.i* und *f* .

BEFEHLE FÜR MMS BITSTRINGS

Befehlsvariablen (**/Oper.ctlVal[CO]*), die einem MMS Bitstring entsprechen - wie Tcmd oder andere CODED ENUM basierende bTypes, werden bei Import als Variablen mit dem Datentyp *UDINT* angelegt. Der Datentyp kann nachträglich manuell auf den Datentyp *BOOL* geändert werden, z. B. für Variablen **/TapChg/Oper.ctlVal[CO]*. Hat die Befehlsvariable den Datentyp *BOOL*, wird der Treiber bei **Sollwert setzen** den Wert *FALSE* auf *0x40 (lower)* und den Wert *TRUE* auf *0x80 (higher)* mappen. Somit kann diese *BOOL* Variable direkt mit Schaltaktionen in der **zenon Befehlsgabe** verknüpft werden.

BEFEHLE - ÜBERSICHT

- Select und Cancel (auf Seite 104)
- Additional Cause Diagnosis (auf Seite 106)
- Serviceparameter des Befehls (auf Seite 108)

8.2.1 Select und Cancel

Der IEC850 Treiber arbeitet mit dem zenon Modul **Befehlsgabe** zusammen. Der Treiber stellt dabei der **Befehlsgabe** spezielle Funtkionalitäten für die Befehle Select und Cancel zur Verfügung.

So kann in der **Befehlsgabe** z.B. ein Select von einem Operate getrennt werden. Nach dem Select kann anstelle von Operate ein Cancel folgen.

Voraussetzung dafür ist, dass bei der Aktion in der Befehlsgabe die Eigenschaft **Zweistufig** aktiviert ist. Bei der Aktionsvariable muss die Eigenschaft **Select Before Operate** aktiviert sein.

Der Treiber unterstützt während eines Befehls ein Cancel ebenfalls nur mit der zenon **Befehlsgabe**.

ABLAUF

Der Treiber sendet automatisch ein Select zum IED, immer wenn dieses gemäß dem Control Model erforderlich ist.

- Sendet die **Befehlsgabe** einen zweistufigen Befehl und die Eigenschaft **Select Before Operate** der Aktionsvariable ist aktiv, dann führt der Treiber - gemäß dem *ctlModel* des Datenobjekts - ein Select bereits bei der ersten Stufe der Befehl-Aktion aus.
  - Für *ctlModel=2* oder *4* wird ein Select an das IED (850-Server) gesendet.
  - Ein Select für die Werte *ctlModel=1* oder *3* (direct control) wird vom Treiber sofort positiv für die **Befehlsgabe** beantwortet. Damit wird vom Treiber eine positive Ausführung des Select an die Aktion der Befehlsgabe bestätigt, ohne dass das IED in der Kommunikation involviert ist.
  - Ein Select für den Wert *ctlModel=0* (status only) wird negativ beantwortet, da ein IED keinen Befehl ausführen wird.

Hinweis: Select Before Operate bezieht sich auf die Eigenschaft bei der *Oper.ctlVal[CO]* Variable in zenon und darf nicht mit dem Datenattribut *SBO* in Datenmodell des 850-Servers verwechselt werden.

Antwortet der IEC 61850 Server auf das Select positiv, speichert der Treiber die Information über das ausgewählte Datenobjekt. Im folgenden Befehl (aus der zweiten Stufe der Befehlsgabe) führt der Treiber nur noch das Operate aus.

Die Eigenschaft **Select Before Operate** der *ctlVal* Variablen hat keinen Einfluss auf Kommandoausführung, wenn der Wert der *ctlVal* direkt gesetzt wurde und nicht durch eine Aktion der **Befehlsgabe**.

CANCEL OF SELECT

Wenn in der **Befehlsgabe** die Befehlsaktion auf der zweiten Stufe abgebrochen wird, sendet der Treiber ein Cancel (*ctlModel=2* oder *4*) oder bestätigt sofort den Cancel für die Aktion der Befehlsgabe (*ctlModel=1* oder *3*).

CANCEL OF OPERATE

Der Treiber unterstützt Cancel operate, nur über das Modul **Befehlsgabe** in zenon.
Wird bei der Kommandovariable die Eigenschaft **Operate abbrechen** aktiviert, kann damit bei Control Model mit erhöhter Sicherheit (*3* und *4*) ein noch nicht abgeschlossenes Operate abgebrochen werden. Wird ein laufendes Operate in der Befehlsgabe abgebrochen, sendet der Treiber zum Server ein Cancel.

Die Eigenschaft **Operate abbrechen** der *ctlVal* Variablen hat keinen Einfluss auf die Kommandoausführung, wenn der Wert der *ctlVal* direkt gesetzt wurde und nicht durch eine Aktion der **Befehlsgabe**.

Hinweis: nicht alle IEC 61850 Server unterstützen ein Cancel nachdem ein Operate begonnen wurde.

8.2.2 Additional Cause Diagnosis

Laut IEC 61850 Standard, soll ein 850-Server für negative Antworten eines Befehls ein zusätzliches Service mit *AdditionalCauseDiagnosis* an den Treiber senden (*AddCause* in *LastApplError* Struktur). Die möglichen Werte der *AddCause* Serviceparameter sind in der IEC 61850-7-2 gelistet. Der Treiber kann den empfangenen *AddCause* an zenon weitergeben. Dafür werden die Variablen vom **Treiberobjekt-Typ** *Command Info* verwendet.

Über weitere Variablen von Treiberobjekt *Command Info* kann der Treiber auch die Runtime informieren über den aktuellen Zustand des laufenden Befehls (z. B. Select/Operate/Cancel) und die Auskunft darüber geben, ob der Befehl erfolgreich war.

'COMMAND INFO' VARIABLEN

Bereits während des laufenden Befehls stellt der Treiber die zusätzliche Information in Variablen des Treiberobjekttyps *Command Info* zur Verfügung (siehe Treiberobjekte (auf Seite 62)), sobald diese im Projekt erstellt wurden. Variablen, die den Zustand des aktuellen Befehls wiedergeben, können für jede **/Oper*-Struktur erstellt werden. Sie geben dann die Auskunft nicht nur über Operate aber auch über Select und Cancel Befehle. Der Treiber unterstützt *Command Info* Variablen mit folgenden Referenzen:

- **/Oper.AddCause*:
  Nach einem gescheiterten Befehl, zeigt die Variable den Wert der Fehlerinformation - AddCause. Diese Information steht zur Verfügung nur wenn diese vom 850-Server verschickt wurde. Nicht jeder Server unterstützt diesen Dienst. In diesem Fall bleibt der Wert der Variable *0*.
- **/Oper.ControlRun*:
  Die Variable zeigt den Zustand oder das Ergebnis des aktuellen Befehls:
  - *1* = Befehl läuft
  - *0* = Befehl erfolgreich
  - *-1* = Befehl gescheitert
- **/Oper.ControlState*:
  Die Variable zeigt den Zustand des gerade laufenden Befehls (sequentiell verarbeitet):
  - *-1* = Befehl beendet
  - *0* = ctlModel lesen
  - *1* = 'Select' gesendet
  - *2* = 'Operate' gesendet
  - *3* = 'Cancel of Select' gesendet

- 4 = 'Operate response' empfangen (nur in ctlModel=3 o. 4)
- 5 = 'Cancel of Operate' gesendet (nur in ctlModel=3 o. 4)

Die Referenzen der *Command Info* Variablen benötigen kein Functional Constraint, sie müssen jedoch das Datenobjekt der entsprechenden **/Oper.ctlVal*[CO]* Variable adressieren.

Beispiel:

Für die Variable *PLC-Marker*, *BOOL*, mit der Referenz (z.B. **Symbolische Adresse**) *GE650!GEDeviceF650/CSWI1/Pos/Oper.ctlVal[CO]* können Sie die *Command Info* Variablen mit folgenden Referenzen erstellen:

- *Command Info*, *SINT*: *X!GEDeviceF650/CSWI1/Pos/Oper.AddCause*
- *Command Info*, *SINT*: *X!GEDeviceF650/CSWI1/Pos/Oper.ControlRun*
- *Command Info*, *SINT*: *X!GEDeviceF650/CSWI1/Pos/Oper.ControlState*

Fall 1:

Wenn Sie den Wert direkt auf **/Oper.ctlVal* setzen oder die Variable nicht die Eigenschaft **Select Before Operate** besitzt, ist die Reihenfolge der Zustände folgende:

1. */Oper.ControlRun = *1* und */Oper.ControlState = *0* - Befehl lauft, der Treiber überprüft Control Model.
2. */Oper.ControlState = *1* und dann */Oper.ControlState = *2* - Befehl lauft, der Treiber sendet Operate.
3. */Oper.ControlRun = *0* und */Oper.ControlState = *-1* - Befehl erfolgreich beendet.

Fall 2:

Wenn Sie die zweistufige Befehlsgabe für **/Oper.ctlVal* mit der Eigenschaft **Select Before Operate** verwenden, ist die Reihenfolge der Zustände für jeden Befehlsdienst - für Select, dann Operate und/oder Cancel - folgende:

1. */Oper.ControlRun = 1 und */Oper.ControlState = 0
2. */Oper.ControlState = x
3. */Oper.ControlRun = 0 und */Oper.ControlState = -1

Dabei gilt: x=1 oder 2 oder 3 ist, je nach Befehlsdienst.

Fall 3:

Wurde ein Befehl gesendet und der Wert */Oper.ControlRun bleibt auf dauer 1 ("lauft") - also ändert sich weder auf 0 noch -1, dann bedeutet es dass der Treiber immer noch warten muss auf die zugehörige Antwort vom 850-Server. Der 850-Server hat der Befehl noch nicht bestätigt - weder positiv noch negativ. Dies kann passieren unabhängig davon dass der sonstiger Datenaustausch mit dem Server weiterhin funktioniert. Es kann passieren während der 850-Server z.B. die zeitaufwendige

Interlocking überprüft oder wäre ein Hinweis auf eine Überlastung des 850-Servers (oder fehlerhaftes Verhalten bezüglich Befehlsbearbeitung).

> **Achtung**
>
> Die **Command Info** Variablen haben keinen Wert so lange kein Befehl gegeben wurde. Um diese Variablen auszuwerten, müssen Sie zuerst einen Wert auf die **/Oper.ctlVal[CO]* Variable setzen.

8.2.3 Serviceparameter des Befehls

Die Befehlsdienste (z.B. Operate), die der Treiber an den 850-Server sendet, bestehen aus definierten Strukturen mit zusätzlichen Parametern. Die Struktur ist in dem IEC 61850 Standard definiert.

Diese Parameter werden folgendermaßen behandelt.

CHECK-PARAMETER

Per Default sendet der Treiber alle Befehle mit *Check = 0*. Es gibt zwei Methoden, um *Check*-Wert für den folgenden Operate oder SelectWithValue Dienst zu setzen:

1. Der *Check*-Parameter kann durch das Setzen eines Werts für die **/Oper.Check[CO]* Variable vordefiniert werden. Der aktuelle Wert der Variable wird für den Befehlsdienst-Parameter verwendet.

   Achtung: Das Datenattribut *Check* ist eine PACKED LIST (MMS Bitstring) mit 2 Bits. Gemäß IEC61850-8-1 8.1.3.5 soll das erste Member der PACKED LIST auf Bit(0) gemappt werden und dieses Bit soll das höchstwertige (most significant) Bit des Oktet sein. Das heißt, dass die gültigen Sollwerte für das *Check* Datenattribut 128 (0x80 synchrocheck), 64 (0x40 interlock-check), 192 (0xC0 beide Check-Bits) und 0 sind.

2. In zenon durch Setzen in der Befehlsgabe: Aktion - Eigenschaft **Qualifier of Command**. Die gültigen Sollwerte sind 2 - synchrocheck (umgedreht 01b), 1 - interlock-check (umgedreht 10b), 3 (beide Check-Bits) und 0. Bei der Methode wird die eventuell angelegte **/Oper.Check[CO]* Variable ignoriert.

TEST-PARAMETER

Per Default sendet der Treiber alle Befehle mit *Test = 0*. Das *Test*-Flag für die folgenden Operate, SelectWithValue und Cancel Dienste, kann durch Setzen eines Werts für die **/Oper.Test[CO]* Variable vordefiniert werden. Der aktuelle Wert der Variable wird für Befehlsdienst-Parameter verwendet.

Achtung

Wenn Sie in der Runtime auf die *[CO] Variablen die neuen Werte absetzen, werden diese an den Treiber weitergegeben. Im Treiber warten die Werte dann auf das nächste Operate bzw. SelectWithValue.

Die neue Werte werden erst mit dem anschließenden Befehl zum Server geschrieben.

Ein Verbindungsausfall setzt die Werte, auch die Werte aller CO Variablen, auf den Status *INVALID*. Die Wiederherstellung der Verbindung setzt die Werte dann auf 0 zurück.

ORIGINATOR

Die *Originator* Datenattribute *orCat* (Category) und *orIdent* (Identification) zeigen, welcher Originator (z. B. ein Client der sich auf bay level befindet) die letze Änderung der Daten (z. B. ein Kommando ausgegeben) verursacht hat.

Der Treiber setzt den Originator beim Senden eines Befehls zusammen aus:

- **orCat**
  Dialog **Basic Settings**
- **orIdent**
  Dialog **Client configuration**

Der orIdent kann eingesetzt werden, basierend auf den Rechnernamen auf dem der Benutzer den Befehl ausgelöst hat. Aktivieren Sie dazu die Eigenschaft **Use Client orIdent** im Dialog **Server**. Legen Sie zustätzlich im Treiberdialog **Client configuration** einen **Hostname** für jeden Rechner im zenon Netzwerk an.

Der Default origin.orIdent *'zenon: <computer_name>'* wird in Befehlen verwendet, wenn:

- in der Client-Konfiguration für den entsprechenden Rechner (= **Hostname**) kein orIdent projektiert ist.
- in der Client-Konfiguration für den entsprechenden Rechner kein Eintrag mit **Hostname** projektiert wurde.

Ist die Eigenschaft **Use Client orIdent** nicht aktiv, wird der *orIdent* des Prozessführenden Servers eingesetzt.

Beispiel

Beim Ausführen eines Befehls setzt der Treiber für die *Originator* Datenattribute in SBOw, Oper und Cancel den Originator wie folgt:

Beispiel 1 - Defaulteinstellungen:

- **orCat =** *2* (station control)
- **orIdent =** leer

Ergebnis: *2,'zenon: <computer_name>'* (String zwischen Anführungszeichen - CHAR(27)) , e.g. 'zenon: MyPC'

Beispiel 2 - Checkbox "Use Client orIdent" deaktivert:

- **orCat =** *2* (station control)
- **orIdent =** "SCADA_850_Client"
  Konfiguriert im **Hostname** für **Server 1** und **Server 2** im Dialog **Client configuration**

Ergebnis: *2,'SCADA_850_Client'*

Beispiel 3 - Checkbox "Use Client orIdent" aktivert:

- **orCat =** *2* (station control)
- **orIdent =** "PC_Room01"
  Konfiguriert im **Hostname** für alle Rechner im zenon Netzwerk, welche sich physikalisch in einem Raum befinden.

Ergebnis: *2,'PC_Room01'*

Die *Origin* Attribute sind obligatorischer Teil der Oper, SBOw und Cancel Dienste und optional beim oberem Knoten eines Datenobjekt z.B. 'Pos'. Um in der Runtime den *Originator* der Wertänderungen **/Pos/stVal[ST]* zu analysieren, können Sie in zenon dazugehörige Projektvariablen anlegen, wenn in Datenmodell des 850-Servers die **/Pos/origin.orCat[ST]* und **/Pos/origin.orIdent[ST]* existieren.
Wenn der 850-Server kein **Service Tracking** unterstützt, dann ist in diesen Attributen ersichtlich, welcher Client als Letzter einen erfolgreichen Befehl gab. Die Attribute mit Functional Constraint *CO* sind dazu nicht geeignet, da es laut Standard nicht erlaubt ist, diese von einen 850-Server abzulesen.

Achtung: Um den Originator eines Datenobjekts zu analysieren, muss der 850-Server entsprechend programmiert sein. Der Server muss das *Origin* des empfangenen Befehls von **/Oper.origin.*[CO]* in diese *ST* Attribute übernehmen.

Beim Empfang eines Reports leitet der Treiber den *Originator* an die Runtime vorrangig weiter bevor er die anderen Datenattribute des empfangenen Datenobjekts weiterleitet.

Hinweis: Sie können *orCat* oder *orIdent* Variablen benutzen, um einen *dynamischen Grenzwerttext* für Wertänderungen der *stVal* Variable zusammenzustellen.

CTLNUM (CONTROL SEQUENCE NUMBER) PARAMETER

Der Treiber erstellt automatisch die Werte des *ctlNum* Parameters und sendet in Befehlen mit. Jedes Mal, wenn ein Wert auf die **/Oper.ctlVal[CO]* Variable gesetzt wurde, erhöht der Treiber automatisch den Wert des **/Oper.ctlNum[CO]*. Der Treiber fängt wieder bei *0* an, wenn der Wert *127* erreicht hat. Eine zenon Variable *ctlNum[CO]*, auch wenn projektiert, wird in der Runtime jedoch nicht aktualisiert. Der *ctlNum* Wert dient ausschließlich korrekten Befehlsausführung, nicht der Information für Benutzer. Der Treiber zählt auch **Sollwert setzen** auf *ctlVal* wenn *ctlModel=0* (keine Aktion) ist.

T (ZEIT) PARAMETER

Der Treiber sendet automatisch in Befehlen die Zeit der Befehlsausführung sowie *TimeQuality* und *TimeAccuracy* Informationen. Weitere Details finden Sie im Kapitel **TimeQuality und TimeAccuracy eines Befehls**.

8.3 Service Tracking

Service tracking wurde mit Edition 2 des IEC61850 Standards eingeführt. Als Grundvoraussetzung muss der IEC61850 Server mindestens eine Logical Node LTRK im Datenmodell beinhalten.

Um diese Funktionalität zu nutzen, müssen entsprechende zusätzliche Service-Tracking-Variablen angelegt werden.

Detailierte Informationen zum Anlegen dieser Variablen erhalten Sie in den Kapiteln Online-Import von einem IEC 61850 Server (auf Seite 70) bzw. Offline-Import (auf Seite 75).

Achtung

Die zusätzliche Service-Tracking-Variablen - von Treiberobjekttyp "Service tracking" - bekommen in der Runtime nur dann die Werte, wenn in dem Projekt wenigstens eine *[SR] Variable vom Treiberobjekttyp "SPS-Merker" von Logical Node LTRK in diesem Logical Device angelegt wurde.

Ein SPS-Merker von LTRK muss nur existieren und muss weder angefordert noch in einem Bild angezeigt werden.

Ein Service Tracking Objekt mit FC = *SR* kann pollend oder per Report empfangen werden. **Hinweis:** die Daten aus LTRK sollen laut Standard die Steuerungen (850-Server) in Reports zur verfügung stellen.

Der Treiber unterstützt auch Data Modelle mit mehr als eine Instanz des LTRK Knoten pro Logical Device.

Info: Beinhaltet ein Bild *Service Tracking* Variablen, welche bisher noch nicht angemeldet waren, bleiben die Variablen, somit auch die verknüpfte Bildelemente, so lange leer, bis eine Service Tracking Information vom Server empfangen wird.

SERVICE TRACKING DER DIENSTEN RELEVANTEN FÜR BEFEHLE

Wird ein SPS-Merker von LTRK angelegt, fordert der Treiber automatisch von der Steuerung alle relevanten Data-Attribute der folgenden Data Objekte mit Service tracking an:

- *SpcTrk*
- *DpcTrk*
- *IncTrk*
- *EncTrk*
- *ApcFTrk*
- *ApcIntTrk*
- *BscTrk*
- *IscTrk*
- *BacTrk*

Diese Data Objekte in LTRK gehören alle der CTS Common Data Class an und dienen Service tracking der Diensten relevanten für Befehle, wie: Select, SelectWithValue, Operate, etc.

Beispiel:

Um Service tracking über die Data Objects **CSWI1/Pos* und **CSWI2/Pos* zu empfangen, legen Sie folgende Variablen an:

- eine SPS-Merker Variable *LTRK*[SR], für beliebiges Datenattribut. Damit wird gewährleistet, dass der Treiber während der Association mit dem 850-Server das Datenmodell in diesem LTRK Knoten nicht überspringt.
- die *Service tracking* Variablen: *CSWI1/Pos_serviceType[SR] und *CSWI2/Pos_serviceType[SR] (und weitere, e.g. *_respAddCause[SR])

Führt einer der 850-Clients einen Befehl aus, über den dann der 850-Server in Service Tracking die Information liefert, so wird diese Information entsprechend ObjRef (Object Reference) auf die richtigen Variablen verteilt.

SERVICE TRACKING DER DIENSTEN RELEVANTEN FÜR CONTROL BLOCKS

Der IEC 61850 Standard sieht es auch vor Service tracking der Control Block relevanten Diensten, wie: SetBRCBValues, SetURCBValues, SetGoCGValues, SelectActiveSG, etc. Das Tracking über diese Dienste wird in 850-Servern in weiteren Data Objekten in LTRK abgebildet, welche einer anderen Common Data Class (als CTS) angehören, e.g. BTS, UTS etc.

Die Service Tracking Variablen für Control Blocks werden beim **Online Import** nicht angeboten, sie können jedoch manuell angelegt werden:

- Legen Sie die LTRK*[SR] Variablen von Treiberobjekttyp SPS-Merker an, wenigstens die LTRK*/objRef[SR], z.B. alle *LTRK1/UrcbTrk/*[SR] für Tracking über Unbuffered Report Control Blocks.
  Per Online- bzw. Offline-Import.
- Legen Sie zusätzliche Variablen vom Treiberobjekttyp Service Tracking an, z.B. alle *LLN0/urcbST01_*[SR] für Tracking über URCB mit den Namen "urcbST01".
  Dies ist manuell durchzuführen.

Beispiel:

Um Service tracking über Wertänderungen der Attribute RptEna[BR] im Report Control Blocks ‘brcbA01’ und ‘brcbB02’ zu empfangen, legen Sie folgende Variablen an:

- die SPS-Merker Variablen *LTRK*BrcbTrk/objRef[SR] und *LTRK*BrcbTrk/rptEna[SR]
- die *Service tracking* Variablen: *brcbA01_objRef[SR],*brcbB02_objRef[SR], *brcbA01_rptEna[SR] und *brcbB02_rptEna[SR]

ERRORCODE[SR] VARIABLEN - VERHALTEN FÜR STATUS-BIT M1

Wenn bei einem Data Objekt zusätzliche Service-Tracking-Variablen angelegt wurden, gibt es keine vordefinierte Reihenfolge, in der die Werte vom Treiber zur Runtime übertragen werden.

Die Datenkonsistenz, z.B. für das Erzeugen von dynamischen Limit Texten für die CEL, ist jedoch durch folgenden Prozess gewährleistet:

- Bei Jedem Empfang von Service Tracking setzt der Treiber das Statusbit **M1** bei der Service-Tracking-Variable **_errorCode[SR]*.
- Nachdem alle Service-Tracking-Werte an die Runtime übertragen wurden, setzt der Treiber das Statusbit M1 der **_errorCode[SR]* Variable zurück (*M1 = 0*).
- Dieses Rücksetzen des M1 Statusbits ist ein geeigneter Auslöser für die Erzeugung eines CEL/AML Eintrages.

> **Achtung**
>
> Wenn sie das Statusbit M1 mit einer Reaktionsmatrix auswerten und die Service-Tracking-Werte pollend und nicht über Reports empfangen werden, dann löst der Initialwert auch einen CEL/AML Eintrag aus.

ALLCTS[SR] VARIABLE

Eine **_allCTS[SR]* Variable beinhaltet die empfangenen Service-Tracking Information, bestehend aus den Datenattributen der Common Data Class CTS.

- Diese Variable ist eine String-Variable.
- Der Wert der Variable ist ein Text, bestehend aus Datenattributen der Common Data Class CTS in folgender Reihenfolge: *objRef*, *serviceType*, *errorCode*, *originatorID*, *t*, *ctlVal*, *origin.orCat*, *origin.orIdent*, *ctlNum*, *T*, *Test*, *Check*, *respAddCause*.
- Die einzelnen Datenattribute sind jeweils durch einen ; getrennt.
- Die Statusbits der Variable ergeben sich aus der Summe aller Statusbits der Datenattribute.
- Der Zeitstempel ergibt sich aus dem Zeitstempel des letzten Attributes des Service-Tracking Objektes. Ist kein Zeitstempel vorhanden, wird die aktuelle Zeit verwendet.

Wenn das empfangene Service-Tracking-Objekt kein Control Service (CTS) beinhaltet, sondern Common (CST) oder Control Blocks (UTS, BTS etc.), dann besteht die *_allCTS-Variable nur aus Datenattributen der Common Data Classe CST: *objRef*, *serviceType*, *errorCode*, *originatorID*, *t*.

LOG-EINTRÄGE

LOG Eintrag	Debug Level	Beschreibung
Variable %s has wrong format for a service tracking variable.	ERRORS	Die verwendete Variablenadressierung entspricht nicht dem erwarteten Format. Erwartet wird „*Server!Device/Node/Object_Attribute*“
Variable %s cannot be advised since no Service Tracking Variable from Driver Object Typ SPS-Merker was found.	ERRORS	Der Treiber kann die Variable nicht von der SPS anfordern, da im Editor keine [SR] Variable vom Typ “SPS-Merker" angelegt wurde. Legen Sie mindestens eine entsprechende Variable an.

8.4 Mapping von Double Point Values

Double Point Value Mapping ist eine Standardfunktion der zenon Energy Treiber. Es beeinflusst nur die zenon Runtime und hat keine Auswirkung auf die Treiberkommunikationen mit einem Gerät. Die Konfiguration erfolgt in den Treibereinstellungen in der Registerkarte **Basic Settings**.

Hinweis: Es wird empfohlen, die Option **Deactivate standard double point value mapping** in der Treiberkonfiguration default auf inaktiv zu belassen.

Mittels Double Point Value Mapping wandelt der Treiber Werte so um, dass sie benutzerfreundlich dargestellt werden. Das gilt allerdings nur für HMI.

Der Treiber kommuniziert mit einem Gerät immer mit Werten für Double Points (Doppel-Punkte) mit 2-Bit-Informationen. Dies entspricht den Definitionen der Energie-Standards. Das bedeutet:

Parameter	Double Point	Wert	Bedeutung
Intermediate	*00b*	*0*	Schalter weder offen noch geschlossen, z. B. die End-Position noch nicht erreicht.
Off	*01b*	*1*	Schalter offen
On	*10b*	*2*	Schalter schließen/geschlossen
Fault	*11b*	*3*	Fehler

Double Points werden im Energiesektor aus historischen Gründen mit 2 Bits kodiert: Die Übertragung eines Telegramms auf einer serielle Verbindung (RS232) mit einer Serie von Werten, die nur *0* enthalten, war nicht gegen Übertragungsfehler gesichert. Um die Sicherheit zu erhöhen, wurde in den ersten Standards beschlossen, den Wert für *OFF* nicht als *0* zu senden, sondern als *01b*, was dezimal *1* entspricht. Diese Double Point Values spiegeln auch exakt die Art wieder, wie zwei Sensoren die physikalische Position eines Schalters erfassen.

Allerdings sind die damit übermittelten Werte für Menschen eventuell verwirrend:

- *OFF = 1*
- *ON = 2*

Menschen sind von allen anderen Geräten und Systemen gewohnt:

- *OFF = 0*
- *ON = 1*

  Gleichzeitig werden im gleichen Standard die Single Point Values mit *OFF = 0* und *ON = 1* definiert.

Um gefährliche Fehlhandlungen des Benutzers zu vermeiden, bieten die zenon Energy-Treiber ein eigenes Double Point Value Mapping. Ohne DPI Mapping musste der Benutzer sich immer bewusst sein, auf welcher technischen Ebene er gerade agiert und Informationen erhält bzw. übermittelt. In

Stress-Situationen kann das sehr leicht zu schwerwiegenden Fehlhandlungen führen; zum Beispiel, wenn *ON* statt *OFF* gesendet wird.

MAPPING FÜR HMI

Mit dem Double Point Value Mapping haben in der zenon Runtime alle Double Points folgende Werte:

- *Intermediate = 2*
- *Off = 0*
- *On = 1*
- *Fault = 3*

> **Info**
>
> Diese Funktion kann in den Treibereinstellungen deaktiviert werden. Allerdings können dann einige Funktionalitäten wie Befehlsgabe oder ALC nicht mehr verwendet werden.
>
> **Empfehlung:** Verzichten Sie auf Numerische Elemente und Zahlenwerte, um *OFF/ON* oder *OPEN/CLOSE* anzuzeigen. Verwenden Sie Combi Elemente mit grafischen Symbolen oder Text-Elemente.

DPI-MAPPING IN IEC850 CLIENT TREIBER

Dem IEC 61850-7-3 Standard folgend hat in Common Data Classes: Double Point Status (DPS) und Controllable Double Point (DPC) das Datenattribut *stVal* den Datentyp *CODED ENUM* mit dem Wertebereich: intermediate-state (00) | off (01) | on (10) | bad-state (11). Daher wartet der Treiber, dass ein Double Point Attribut des Datentyp *CODED ENUM* auf dem Kommunikationsprotokoll als ein MMS des Datentyps *Bitstring* gemappt ist. Das Datenattribut *stVal* kann somit die folgenden Werte haben: *0x00*, *0x40*, *0x80* und *0xC0*. Dieser Basis-Datentyp ist in der SCL-Sprache mit dem Namen *Dbpos* zulässig (IEC 61850-6). In zenon entspricht dies einer Variable mit Datentyp *UDINT* (default) oder *STRING* (wenn manuell geändert) - siehe Zuordnung der Datentypen (auf Seite 65).

In der Treiberkonfiguration soll die Option **Deactivate standard double point value mapping** default auf inaktiv bleiben um die Module der zenon Energy Edition zu verwenden, wie z. B. die Funktionalitäten der **Befehlsgabe** und **ALC**.

Mit den Default-Einstellungen ordnet der Treiber Werte des *stVal* Datenattributs von DPS und DPC gemäß der folgenden Tabelle zu:

Positionswert im End-Gerät	zenon Dbpos Wert - STRING	zenon Dbpos Wert - UDINT ungemappt	zenon Wert - UDINT gemappt
Intermediate *(00b)*	*'00'*	*0*	*2*

Positionswert im End-Gerät	zenon Dbpos Wert - STRING	zenon Dbpos Wert - UDINT ungemappt	zenon Wert - UDINT gemappt
Aus (01b)	*'01'*	*64 (0x40)*	*0*
Ein (10b)	*'10'*	*128 (0x80)*	*1*
Störung (11b)	*'11'*	*192 (0xC0)*	*3*

Der Treiber konvertiert nur Werte für Variablen mit Namen **/stVal* und numerischen Datentyp, der in der SCL-Sprache (IEC 61850-6) den bType *Dbpos* entspricht. D.h. die Werte der **/stVal* Variable mit manuell auf STRING geändertem Datentyp sind von Mapping ausgeschlossen.

Zum Beispiel bedeutet in der Runtime:

Position des Schalters	UDINT gemappt	UDINT ungemappt	STRING
off	*0*	*64*	*'01'*
on	*1*	*128*	*'10'*

Die Variablen mit anderen Namen als *stVal* sind von **DPI Mapping** ausgeschlossen. **Beispiel:** in Datenobjekt *Pos* (von CDC *DPC*) sowohl die **/Pos/stVal[ST]* wie die (optionale) **/Pos/subVal[SV]* beide *Dbpos* sind, jedoch der Treiber wird der Wert der *subVal* Variable nicht mappen.

DPI-MAPPING IN BEFEHLSRICHTUNG

Im Unterschied zu IEC 60870 und DNP3 Protokollen, sind in dem IEC 61850 die Befehle in Common Data Class für Controllable Double Point (DPC) von Datentyp *BOOL*, also keine Double Point Values. **Beispiel:** obwohl das **CSWI/Pos/stVal[ST]* ein *Dbpos* ist, das **CSWI/Pos/Oper.ctlVal[CO]* ist im IEC 61850 Standard auf ein *MMS BOOLEAN* gemapt - *off (FALSE)* | *on (TRUE)*. Somit übergibt der Treiber die *ctlVal* Werte *False/True* ohne Zuordnung (mapping) an dem 850-Server.

Es gibt jedoch auch im IEC 61850 Standard die Common Data Classes wo ein Befehl einen *MMS Bitstring* entspricht, wie z.B. Binary controlled step position information (BSC) und Binary controlled analog process value (BAC). In diesen CDCs ist das *ctlVal* von bType *Tcmd* , also ein *CODED ENUM* mit drei gültigen Werten: *stop (0x00)* | *lower (0x40)* | *higher (0x80)*. Nichtsdestotrotz, die meisten regulierbaren Transformatoren werden nur mit Befehlen für Change tap position: *lower/higher* gesteuert, somit besteht dann kein Bedarf für auch *stop* Befehl.

Beispiel: die Variable für **ATCC/TapChg/Oper.ctlVal[CO]* wird beim **Offline** und **Online Import** mit Datentyp *UDINT* erstellt, weil sie einem *MMS Bitstring* Datenattribut entspricht und alle eventuellen Bits eines *Bitstring* Wertes umfassen soll.

Sie können nach dem **Import** für solche Befehlsvariablen der Datentyp von *UDINT* manuell auf *BOOL* ändern. Entsprechen die **/Oper.ctlVal[CO]* Datenattribute einen *MMS Bitstring*, mapt der Treiber bei Sollwert setzen den Wert *FALSE* auf *0x40* (lower) und den Wert *TRUE* auf *0x80* (higher).

8.5 Quality, Zeitstempel und Statusbits der Variable

Variablen in zenon mit **FC (auf Seite 157)=***ST* oder **FC=***MX* (z.B. **/Pos/stVal[ST]*, **/PhV.phsA/cVal.mag.f[MX]*) bekommen Zeitstempel und Statusbits aus den begleiten Datenattributen 'q' und 't'. Dies geschieht im Treiber automatisch. Eine zusätzliche Projektierung in zenon ist nicht notwendig.

Der Treiber ordnet immer ausgewählte Quality Bits (des Attributs 'q') und TimeQuality Bits (des Attributs 't') entsprechenden Statusbits jeder zenon Variable zu, die für ein Attribut des Datenobjekts erstellt wurden und eine **FC (auf Seite 157)=***ST* oder **FC=***MX* hat, z. B. für Statusbits von *stVal* Variable. Der Treiber ordnet auch den Wert des Attributs 't' dem Zeitstempel aller *ST/MX* Variablen des Datenobjekts zu.

Der Treiber benutzt automatisch den Zeitstempel und die Qualität des Datenobjekts für andere Variablen in diesem DO. Deshalb müssen keine Variablen für die Attribute 'q' und 't' separat erstellt werden, außer wenn Sie die Qualitätsbits analysieren möchten, die keinen Statusbits zugeordnet sind oder die Aktualisierungen des Zeitstempels ohne Änderung der Position (*stVal*) o. Messwertes (*mag*) erkennen möchten. Wurden im Projekt doch zusätzlich die zenon Variablen mit Referenzen zu Datenattributen 'q' oder 't' erstellt, wird der Treiber die empfangene Werte 'q' und 't' als Variablenwerte unterstützen.

Beispiel: Der Zeitstempel, empfangen als 't' Attribut, wird für die entsprechende **/stVal[ST]* Variable in der **CEL** verwendet. Wurde im Projekt auch die Variable **/t[ST]* erstellt, hat sie in der Runtime die empfangene Unix-Zeit als *LREAL* Wert, mit Millisekunden in Nachkommastellen.

Info

Der Treiber berücksichtigt per Default die TimeQuality Byte des 't' Attributes. Neben 3 Statusbits der Qualität der Zeit-Information enthält TimeQuality Byte auch 5 Bits für die TimeAccuracy Information - den Genauigkeitswert (N bits of accuracy - performance class). Die theoretisch mögliche Genauigkeitswerte sind zwischen 0..24 und Wert 31 als "unspecified" (alle Bits) definiert. Jedoch nur einige der Genauigkeiten darf ein gültiger 850-Server verwenden. Die niedrigste für ein 850-Produkt erlaubte Genauigkeit ist class T0 - 7 Bits - 10ms.

Wenn TimeAccuracy nicht 31 "unspecified" ist, wird das Feld FractionOfSeconds des Zeitwertes - die binäre Nachkommastellen - entsprechend der Genauigkeit - N+1, abgeschnitten. Anschließlich wird der Zeitstempel auf die Millisekunden gerundet.

Beispiel:

- TimeAccuracy = 10 Bits (class T1, 11 binäre Nachkommastellen): der Treiber kürzt den erhaltenen Zeitwert bis auf Genauigkeit 1/2048=0,00048828125 [s] und rundet auf die Millisekunden.
- TimeAccuracy = 0 Bits (keine gültige performance class im 850-7-2 Standard, 1 binäre Nachkommastelle): der Treiber kürzt den erhaltenen Zeitwert bis auf die 0.5 Sekunden.

Möchten Sie *TimeAccuracy* ignorieren, aktivieren Sie die Checkbox **Ignore time accuracy** im Treiber-Konfigurationsdialog Server (auf Seite 29).

Die ausgewählten Quality und TimeQuality Bits werden den Statusbits der Variablen im gleichen Datenobjekt wie folgt zugeordnet:

IEC61850 Quality oder TimeQuality Bits	Statusbit in zenon	Anmerkungen
Validity = **Invalid** oder **Reserved** oder **Questionable**	*INVALID*	Gekennzeichnet durch ein rotes Quadrat am verknüpften Bildelement.
Overflow	*OV_870* (*Overflow*)	Dies ist auch ein IEC 60870 Status
OutOfRange	*OR_DRV* (*Wert ausserhalb des gültigen Bereiches*)	
Source = Substituted	*SB_870* (*Substituted*)	Dies ist auch ein IEC 60870 Status
Test	*TEST* (*Test bit*)	Dies ist auch ein IEC 60870 Status
OperatorBlocked	*BL_870* (*Blocked*)	Dies ist auch ein IEC 60870 Status

IEC61850 Quality oder TimeQuality Bits	Statusbit in zenon	Anmerkungen
ClockFailure	*T_INVAL* (*Zeit ungültig*)	Der Wert des 't' Attributs wird ignoriert - **stVal** hat den Zeitstempel des lokalen Rechners.
ClockNotSynchronized	*T_UNSYNC* (*ClockNotSynchronized*)	Der Wert des 't' Attributs wird berücksichtigt - **stVal** hat den Zeitstempel von IED. **Anmerkung:** bis Version 7.50: *T_INVAL* und *T_EXTERN*, ab 7.60: *T_UNSYNC*.

Das TimeQuality Bit **LeapSecondsKnown** wird keinem Statusbits zugeordnet.

Die Variablen eines Datenobjekts ohne 't' Attribut (oder mit 't' Wert *0*) und die Variablen mit anderen Functional Constraint (auf Seite 157) als *ST/MX* erhalten den Zeitstempel der lokalen PC-Uhr.

Info

Sie können Statusbits (z. B. *T_INVAL*) auswerten, indem Sie Combi-Element oder *Reaktionsmatrix* (vom Typ Multi) verwenden. Die Statusbits können in **CEL** oder **AML** gefiltert werden.

CAUSE OF TRANSMISSION (COT)

Für jede Variable stellt der Treiber die Information zur Verfügung, ob der Wert der Variable pollend oder per Report empfangen wurde. Die Information wird auf die Statusbits *COTx* geschrieben.

COT Wert	Statusbits	Beschreibung
1	COT0	Der Wert wurde via Polling erhalten
2	COT1	Der Wert wurde über einen Integrity Report erhalten. **Hinweis:** Dies funktioniert nur, wenn Reason-for-inclusion im Dialog Optional Fields (auf Seite 40) aktiviert ist. Ansonsten ist die COT=3
3	COT0, COT1	Der Wert wurde über einen Report erhalten. **Hinweis:** Ist Reason-for-inclusion im Dialog Optional Fields (auf Seite 40) aktiviert, sind es nur Reports mit Triggeroption data-change,

COT Wert	Statusbits	Beschreibung
		data-update oder quality-change. Ist Reason-for-inclusion nicht aktiviert, sind es alle Reports - auch Integrity und GI.
20	COT2, COT4	Der Wert wurde über einen General Interogation Report erhalten. **Hinweis:** Dies funktioniert nur, wenn Reason-for-inclusion im Dialog Optional Fields (auf Seite 40) aktiviert ist. Ansonsten ist die COT=3

Info

Der COT-Wert für den IEC850 Treiber ist an den IEC60870 Standard angelehnt:

1 - periodic, cyclic

2 - background scan

3 - spontaneous

20 - general interrogation

Während der Befehlsausführung über das Modul **Befehlsgabe** aktualisiert der Treiber die Statusbits der */Oper.ctlVal[CO] Variable wie folgt:

- Select, SelectWithValue: *SE_870 + COT_act(6)*, dann *SE_870 + COT_actcon(7)* mit eventuell *N_CONF*
- Operate: *COT_act(6)*, dann *COT_actcon(7)* mit eventuell *N_CONF*, dann eventuell *COT_actterm(10)*
- Cancel: *SE_870 + COT_deact(8)*, dann *SE_870 + COT_deactcon(9)*

Die **Laufzeitüberwachung** der **Befehlsgabe** wertet diese Statusbits der **Aktionsvariable** aus.

8.5.1 TimeQuality und TimeAccuracy eines Befehls

Bei Senden eines Befehls in *SBOw*, *Oper* und *Cancel* sendet der Treiber auch ein T-Attribut. Dieses T-Attribut enthält Zeit der Befehlsausführung sowie *TimeQuality* und *TimeAccuracy* am letzten Oktett (gemäß Standard 61850-8-1, 8.1.3.7).

Der Benutzer kann manuell bestimmen, welche Werte für *TimeQuality* und *TimeAccuracy* für diesen Befehl verwendet werden. Projektieren Sie dazu zwei Variablen vom Treiberobjekttyp **Settings**. Benennen Sie diese mit *TimeQuality* und *TimeAccuracy*.

Wird ein Kommando mit Zeitstempel gesendet, werden die Werte für *TimeQuality* und *TimeAccuracy* aus diesen Variablen verwendet.

Wurden keine Variablen erstellt oder wurde niemals ein Wert auf diese Variablen *TimeQuality* und *TimeAccuracy* geschrieben, wird das letzte Oktett des T-Attributes mit 0 gesendet.

TIMEQUALITY:

Bit 0 = *LeapSecondsKnown*

Bit 1 = *ClockFailure*

Bit 2 = *ClockNotSynchronized*

TIMEACCURACY:

Wert 0 - 24: Bitanzahl-Genaugkeit der Nachkommastellen des T-Attributs

Wert 31: Genauigkeit nicht spezifiziert (*unspecified*)

8.6 Filetransfer

Filetransfer ist eine Funktionalität definiert im IEC 61850 Standard für Übertragung der Dateien zwischen 850-Server und Client. Damit können Sie z.B. die SCL-Datei des 850-Server hollen, oder disturbance data nach einem Störvorfall (typisch im *COMTRADE*-Format). Ob und welche Dateien für die Übertragung zur Verfügung stehen entscheidet allein der IED (850-Server).

Die übertragene Dateien werden mit gleichen Namen, Format und Inhalt abgespeichert wie der 850-Server sie zum Treiber weitergibt, z. B. stellt der IED die Dateien im *COMTRADE*-Format zu Übertragung bereits werden auf dem Laufwerk des Treibers die Dateien in *COMTRADE*-Format gespeichert mit Namen und Inhalt wie vom IED empfangen.

Für den **Filetransfer** sind drei Funktionen implementiert:

- Ordnerinformation anfordern (auf Seite 123)
- Datei vom Server holen (auf Seite 123)
- Datei am Server löschen (auf Seite 124)

FEHLERBEHANDLUNG

Wenn bei der Ausführung des Dateientransfers die Fehler auftreten, z. B. bei negativer Antwort des Servers, ändert die Kommandovariable ihren Wert auf "*XXX ERROR*" (*XXX* = *DIR*, *GET* oder *DEL*) und der Treiber schreibt optional einen Eintrag in die LOG-Datei (siehe auch Fehleranalyse (auf Seite 152)).

8.6.1 Ordnerinformation anfordern

Um Ordnerinformationen vom IED (850-Server) anzufordern:

1. erzeugen Sie in Ihrem zenon Projekt zwei Stringvariablen des **Treiberobjekttyps** *File Transfer* mit der entsprechenden **Netzadresse**:
   a) Die erste Variable, im weiteren Verlauf als Kommandovariable bezeichnet, mit der Referenz (**Name** / **Symbolische Adress**e) "**!Command*" kann auch dazu benutzt werden, um Dateien zu holen und zu löschen.
   a) Die zweite Variable mit der Referenz "**!Directory*" wird nur für das Ergebnis der Ordneranfrage benutzt. Sie empfängt den Ordnerinhalt als lesbaren Text. Deshalb sollte deren Größe der maximalen Größe des Dateinamen (inklusive des Odners) im IED entsprechen.
2. setzen Sie für die Kommandovariable den Wert "*DIR*" (für das Root-Directory) bzw. "*DIR <file_spec>*".
   <file_spec> ist ein String, der als Übergabeparameter an den IED gesendet wird.

Wenn der Ordner erfolgreich empfangen wurde:

- ändert die Kommandovariable ihren Wert auf "*DIR OK*"
- enthält die Ordnervariable den empfangenen Ordnerinhalt als lesbaren Text. Eine Zeile dieses Texts hat das folgende Format:

  <Dateiname>;<Dateilänge>;<Zeitstempel>.

8.6.2 Datei vom Server holen

Um eine Datei vom IED (850-Server) zu holen, setzen Sie bei der Kommandovariable den Wert "*GET <Dateiname>*".

Wenn die Datei erfolgreich empfangen wurde:

- wird sie in dem Ordner gespeichert, der als **Directory for file transfer** im Treiberdialog **Basic settings** (auf Seite 18) definiert wurde.
  Auch Dateien aus einem Unterordner werden im Hauptordner abgelegt.
- ändert die Kommandovariable ihren Wert auf "*GET OK*"

Die übertragene Dateien werden mit gleichen Namen, Format und Inhalt abgespeichert wie der 850-Server sie zum Treiber weitergibt.

8.6.3 Datei löschen

Um eine Datei im IED (850-Server) zu löschen, setzen Sie bei der Kommandovariablen den Wert "*DEL <Dateiname>*".

Wenn die Datei erfolgreich gelöscht wurde, ändert die Kommandovariable ihren Wert auf "*DEL OK*"

9 Reporting

Das Reporting wurde durch den IEC 61850 Standard für spontane Kommunikation eingeführt. Die Variablen, die mit einem erfolgreich aktivierten Report Control Block (RCB enable) verknüpft sind, werden nicht länger gepollt, sondern werden von der Steuerung (einen 850-Server, Intelligent Electronic Device - IED) gemeldet, wenn eine Wertänderung auftritt.

Welche Variablen spontan empfangen werden, wenn ein 850-Client einen bestimmten Report Control Block (RCB) anmeldet, ist im DatSet (Data Set) Attribut des RCB definiert. Während Association mit dem 850-Server, der Treiber liest die RCB und deren Data Set konfiguration von IED. Sind die Variablen in einem der Data Sets, der erfolgreich zugewiesenen RCB, so werden sie nicht mehr gepollt. Dieser Vorgang findet automatisch bei jedem Verbindungsaufbau erneut statt. In redundanten zenon Netzwerk, die **IEC850** Treiber am beiden Netzwerk Servern führen unabhängig diese Überprüfung durch. Somit das Ergebnis - ob eine Variable gepollt oder spontan antrifft - kann sich dynamisch je Association ändern; und im Netzwerk am **Server 1** und **Server 2** unterschiedlich enden. Der Treiber kennzeichnet die gepolte und spontane Werte mit unterschiedlichen Statusbits **Übertragungsursache [COTx];** mehr darüber finden Sie in RCB Aktivierung - Überprüfungsmöglichkeiten (auf Seite 138).

Hinweis: Ein 850-Server bekommt die Data Sets Konfiguration aus seiner SCL-Datei. Ein für Reporting geeignetes Data Set besteht aus Referenzen zu Datenobjekten, deren Daten - Wertänderungen, Quality, Zeitstempeln - vom 850-Server spontan gesendet werden.

Um die Aktivierung des Reporting zu ermöglichen, sollten die folgenden Bedingungen erfüllt sein:

- Mindestens eine Variable des Logical Device, in dem der RCB inkludiert ist, muss im Projekt erstellt worden sein. Anderenfalls berücksichtigt der Treiber diesen Logical Device erst gar nicht während des Verbindungsaufbaus.
- Mindestens eine Variable des Logical Node, in dem der RCB inkludiert ist, muss im Projekt erstellt worden sein. Anderenfalls wird der Treiber diesen Logical Node nicht berücksichtigen während des Verbindungsaufbaus.
  Ausnahme: Logical Node *LLN0*. Alle RCBs definiert im logischen Knoten *LLN0* werden für jedes Logical Device automatisch berücksichtigt, wenn das LD wenigsten eine Variable hat, die in der Projekt-Variablenliste erstellt wurde.

- Das Data Set und der Reporting Control Block, der das Data Set benutzt, müssen im selben Logical Device sein.
- Ein RCB wird nicht aktiviert:
  - wenn kein Datenset verknüpft ist.
  - wenn ein leeres Datenset verknüpft ist.
  - wenn das verknüpfte Datenset nicht existiert oder nicht gefunden werden kann.

    In der LOG-Datei werden dann entsprechende Warneinträge erstellt.

BEISPIEL:

1. RCB Name ist *IEDNameLD1/LLN0/urcbA* - somit liegt RCB in Logical Device *LD1* und Logical Node *LLN0;*

- sein Data Set beinhaltet eine Referenz (FCD) zu Datenobjekt: *IEDNameCTRL/Q0CSWI1/Pos* - somit die Daten befinden sich in Logical Device *CTRL*, in Logical Node *Q0CSWI1*.
  - dann wird der Treiber das RCB aktivieren, jedoch vorausgesetzt im zenon Projekt wurde auch wenigstens eine Variable erstellt in Logical Device *LD1*.

1. Ein RCB liegt in anderen Logical Node als *LLN0* - in *IEDNameLD1/GGIO1/brcbST*
   - so muss im Projekt, für diesen Logical Node GGIO1, wenigstens eine Variable erstellt sein, z.B. *IEDNameLD1/GGIO1/brcbST/RptEna[BR]*.

Hinweis: der IEC 61850 Standard verlangt dass ein System nur dann die Services (also z.B. Reporting) eines Logical Device nutzt, wenn es der Wert des *Mod*, *Beh* und *Health* überprüft hat. Somit wäre die aktivierung eines RCBs in Logical Device mit unbekannten Werten des *LLN0/Mod/stVal*, noch **/Beh/stVal*, noch **/Heath/stVal* nicht konform mit dem Standard. In Zweifelsfall legen Sie im Projekt die **/Mod/stVal[ST]* Variablen an, für alle Logical Devices deren Services Sie nutzen wollen.

AUTOMATISCHE VS STATISCHE ZUWEISUNG DER RCBS

Die Konfiguration des Reporting erfolgt in der Treiberkonfiguration - Server (auf Seite 29). Der Treiber unterstützt zwei Vorgehensweisen:

- automatisches Aussuchen und aktivieren der freien URCBs - wenn die Eigenschaft **Max. auto used URCBs** > *0* ist. Diese Methode gilt nur für Unbuffered Reports. Es erleichtert eine schnelle, erste Inbetriebnahme der spontanen Kommunikation.
- statische Zuweisung der gewünschten RCBs - in **RCB assignment/dynamic datasets** (auf Seite 47) Dialog. Diese Methode gilt für sowohl Buffered und Unbuffered Reports und ist für Verwendung in fertigen Systemen gedacht.

  Die statisch konfigurierte RCBs haben eine höhere Priorität als die eventuell automatisch ausgesuchte URCBs. Wenn die Aktivierung eines vordefinierten RCB misslingt, kann der Treiber stattdessen ein **URCB** automatisch aktivieren, das auch ein DataSets enthält mit

benötigten Daten. Wenn Sie nicht möchten, dass der Treiber automatisch die URCBs aktiviert, setzen Sie die Eigenschaft **Max. auto used URCBs** auf 0.

Wenn der Treiber aus irgendeinem Grund die Reports nicht aktivieren kann, wird stattdessen Polling benutzt. In diesem Fall wird in der **Connection State**-Variable (auf Seite 98) das Bit 17 oder 25 (*MMS_RCB_ENABLE_FAILED*) gesetzt.

Weiterführende Informationen über die verschiedenen Möglichkeiten die Aktivierung von *RCB*s zu überprüfen finden Sie im Kapitel RCB Aktivierung - Überprüfungsmöglichkeiten (auf Seite 138).

Sie können im Treiber **RCB assignment** manuell konfigurieren oder - in **Energy Edition** - mit Hilfe des **IEC850 Driver Configuration** Wizards, anhand einer SCL-Datei (SCD, CID, eventuell ICD etc).

Hinweis: Für vereinfachten Import der Variablen aus den Datasets der RCBs liefert der **IEC850 Driver Configuration** Wizard zusätzliche Möglichkeiten.

ZUWEISUNG ÜBER REPORTNAME ODER REPORT-ID

Per Default identifiziert der Treiber die RCB Instanzen des 850-Servers über die Namen. Dieses Verfahren wird von allen - mit IEC 61850 Standard konformen - IEDs (850-Servern) unterstützt.

Mit Aktivierung der Einstellung **Use Report-ID for RCB assignment** in der Treiberkonfiguration können Sie ein speziales Verfahren nutzen: die Report ID (RCB.RptId) wird anstatt des Reportnamens verwendet. Das Verfahren ist geeignet nur für spezielle Anwendungen. Damit wird der Treiber alle RCB Instanzen anmelden die gleiche ID haben. Der IEC61850 Standard erlaubt dass auch unterschiedliche RCBs, mit unterschiedlichen Data Sets, ein gleicher Wert in RptID haben. In der Runtime wird der Treiber alle RCBs mit der jeweiligen Report-ID aussuchen und anmelden. Damit kann man der Treiber so konfigurieren dass er mit nur einen Eintrag in der Liste mehrere RCBs anmeldet; und auch automatisch die, die zur Zeit der Treiberkonfiguration noch nicht existiert haben.

Achtung: haben also am 850-Server mehrere Instanzen eines RCBs gleiche ID (was oft vorkommt) wird der Treiber sie alle anmelden, auch wenn sie alle die gleiche Data Sets beinhalten. Im Folge dessen wird kein weiterer 850-Client den Zugriff auf die betroffene RCBs haben!

Verwendung von Report-ID erfordert entsprechende konfiguration in der SCL-Datei des 850-Servers:

- für jeden 850-Client im System muss eigene Report-ID definiert sein; also auch andere ID für je **Hostname** - andere für Prozessführenden Server, andere für Standby Server.
- bei unterschiedlichen RCBs, soll in SCL nur genau eine Instanz mit der ID konfiguriert sein. Damit werden Instanzen gekennzeichnet die ein Client benötigt, z. B. die erste Instanzen jedes RCB für Treiber am Prozessführenden Server, die zweiten für Treiber am Standby Server, die dritten für einen anderen 850-Client im System, etc.

Aktivieren Sie das Verfahren über Report-ID nur wenn Sie sicher sind dass Sie diese spezielle Feature im Treiber nutzen möchten und das IED auch entsprechend bereits vorkonfiguriert wurde.

Achtung: Wenn diese Option später geändert wird, müssen alle RCB Zuweisungen im Dialog **RCB assignment** gelöscht werden, die Treiberkonfiguration geschlossen und erneut geöffnet und **RCB assignment** erneut erstellt werden.

GENERALABFRAGE (GI)

Der Treiber sendet bei Aktivierung des RCBs auch ein General Interrogation (RCB.GI=TRUE) Kommando um die aktuellste Werte der Variablen zu holen. Er sendet bei der ersten Aktivierung der RCBs (= Start der Runtime) und bei Wiederaufbau der Verbindung (nach Verbindungsverlust) eine GI. Der Treiber sendet die GI auch parallel zur Anforderung die gepufferte Events über BRCB.EntryID zu übertragen, um die Aktualisierung der Prozessdaten sicherzustellen, auch wenn der Grund des Verbindungsverlustes der Ausfall des 850-Servers war.

Die Generalabfrage wird jedoch nicht gesendet wenn es sich um einen wiederholten Versuch handelt, einen Report zu anmelden (enable). Dies geschieht nach Ablauf der in der Eigenschaft **RCBs enable retries** konfigurierten Zeit. Damit werden Daten-Dubletten vermieden, da die Werte bereits durch das Polling empfangen wurden.

DATA SETS

Der IEC850 Treiber unterstützt Reports mit in dem IED vordefinierten Data Sets (in der SCL-Datei) oder mit Dynamic Data Sets - mit RCB.DatSet Attribut, das vom Treiber während der Aktivierung des RCB dynamisch definiert wurde.

Wie vom IEC 61850 Standard vorgesehen, sind für einen Report Datenobjekte mit Functional Constraints (FC) verfügbar:

- Status - *ST*
- Measurands - *MX*
- Service tracking - *SR*
- Configuration - *CF*
- und weitere: *SP*, *SV*

Wenn in einem Data Set eine nicht standardkonforme Referenz definiert wurde (andere als hier aufgelistete FCs), aktiviert der Treiber trotzdem den *RCB*. In den empfangenen Reports werden jedoch die Daten mit ungültigen FCs ignoriert. Die betroffenen Datenobjekte und Datenattribute werden weiterhin gepollt.

In einem Data Set:

- Datenobjekte mit unterschiedlichen FCs können zusammen verwendet werden, wenn der 850-Server dies unterstützt.
- Datenobjekte aus unterschiedlichen Logical Devices können enthalten sein.

Details über dynamische Datasets finden Sie im Kapitel Dynamische Datasets (auf Seite 141).

DATA SETS MIT FCD ODER FCDA

Die im Treiber konfigurierten dynamischen Datasets bestehen immer aus FCD-Referenzen:

- Functional Constraints (FC)
- Referenz zum Data Object (LD/LN/DO)

In einem IED bereits vordefinierte Datasets bestehen meistens aus FCD-Referenzen, sie können jedoch auch aus FCDA-Referenzen bestehen.

Eine FCDA-Referenz besteht aus:

- FCD-Referenz, und
- Data Attribute (daName)

Hinweis: In einer SCL-Datei beginnen die Elemente in Datasets immer mit dem XML-Keyword "FCDA". Dies gilt auch für Elemente, die eine FCD-Referenzen sind:

- FCD-Referenz:
  <FCDA ldInst="ANN" prefix="PLT" lnClass="GGIO" lnInst="2" doName="Ind01" fc="ST" />
- FCDA-Referenz:
  <FCDA ldInst="LD0" prefix="X" lnClass="GGIO" lnInst="110" doName="Ind1" daName="stVal" fc="ST" />
  - der Unterschied ist das Element "daName".

Um Dateninkonsistenzen auszuschließen, wird dringend empfohlen immer DataSets mit FCD-Referenzen zu verwenden. Dazu muss die SCL-Datei des Servers entsprechend konfiguriert sein. Die "daName" Elemente von Referenzen müssen entfernt und eventuale Duplikate gelöscht werden.

Beachten Sie weiters, dass nur bei FCD-Referenzen die Wertänderungen mit dem aktuellen Zeitstempel des 850-Servers im Report enthalten sind. Bei FCDA-Referenzen beinhalten nur die Reports mit GI oder Integrity die Zeitstempel des Servers.

Sollte der 850-Server die FCDA-Referenzen verwenden, welche nicht auf FCD-Referenzen umkonfiguriert werden können (z.B. da der Server keine Updates der SCL-Datei akzeptiert) wird der Treiber einen Report mit solchen DataSets doch aktivieren. **Ausnahme:** Reports mit Service Tracking werden ausschließlich mit FCD unterstützt!

REPORTS MIT DATA SET MIT FCDA

Bei Reports mit DataSet mit FCDA arbeitet der Treiber nach folgenden Regeln:

- Während Aktivierung des RCB wird die Struktur des DataSets erkannt. Sind hier einzelne Data Attribute vorhanden wird überprüft, ob in diesem DataSet gleichzeitig ein zugehöriges Quality-Attribut definiert ist. Wird keine FCDA-Referenz für das entsprechende Quality-Attribut gefunden, welches jedoch im Datenmodell vorhanden ist, schreibt der Treiber einen LOG-Eintrag:

LOG-Eintrag	Level	Beschreibung
Enabling scan of %s on device %s	MSG	Das angegebene Element (q-Attribut) wird im Zyklus des **Data consistency scan** abgefragt (Default: 5min).

Ab diesem Moment wird das im Report fehlende q-Attribut - je nach Treiberkonfiguration in der Einstellung **Data consistency scan** - automatisch gepollt. Damit wird gewährleistet, dass per Report empfangene Messwerte auch die Qualitätsinformation haben und somit dem IEC61850-Standard entsprechen.

- Ist das t-Attribut in einem empfangenen Report enthalten, wird dieses auch verwendet. Fehlt das t-Attribut im Report, enthalten die Werte der Variablen einen internen Zeitstempel.
- Ist das q-Attribut im empfangenen Report enthalten, werden die Statusbits der Variablen entsprechend aktualisiert. Fehlt dieses q-Attribut spiegeln die Statusbits der Variablen das q-Attribut wie im Treiber bereits bekannt. Dies kann von einem früheren Report, einem Polling des angemeldeten Datenobjekts oder von einem Datenscan (wie in **Data consistency scan** konfiguriert) stammen.
- Wird ein Report empfangen, der nur ein t-Attribut enthält (ohne Messwerte, ohne q-Attribut) wird ein LOG-Eintrag erzeugt, da dies im Standard nicht erlaubt ist. Dieser Fall kann nur dann eintreten, wenn das IED falsch konfiguriert oder aufgebaut wurde. Zum Beispiel bei fehlerhafter SCL-Datei. Der empfangene Zeitstempel wird ignoriert.
  Ausnahme: Wurde eine t-Variable im Projekt angelegt, und später wie der Reportempfang angemeldet, spiegelt der Wert der Variable das empfangene t-Attribut.

LOG-Eintrag	Level	Beschreibung
Received report contains only yyy$t; missing value and quality. Report will be ignored. Probable reason: mistake in IED configuration (SCL)	Warnin g	Es wurde ein Report empfangen, welcher weder Messwerte noch ein q-Attribut enthält. Dies deutet darauf hin, dass der Server mit einer fehlerhaften SCL-Datei konfiguriert wurde. **Beispiel:** In SCL.DataTypeTemplates wurde das t-Datenattribut mit eigener Trigger Options konfiguriert.

TRIGGER OPTIONS

Die im IEC 61850 Standard definierte Trigger Conditions bestimmen wann ein 850-Server ein neuer Report erstellt und zu Client sendet. Der Treiber unterstützt alle im Standard definierte Trigger Conditions. Die Trigger Conditions bestehen aus Trigger Options (**TrgOps**) die in der SCL-Datei des 850-Servers auf zwei Stellen definiert sind:

- per Report Control Block, z.B. in SCL:

  IED.AccessPoint.Server.LDevice.LN0.ReportControl <TrgOps dchg="true" qchg="true"/>

- für relevante Datenattribute (DA), z.B. in SCL:

  DataTypeTemplates.DOType <DA name="stVal" fc="ST" dchg="true" bType="Dbpos"/>

Ein 850-Server erstellt ein neuer *dchg*, *dupd* oder *qchg* Report nur wenn beide Bedingungen - für RCB und für DA - erfüllt sind, die Reports GI und Integrity - nur abhängig **TrgOps** im RCB. Laut Standard kann ein Client (der Treiber) nur auf TrgOps in RCB einen Einfluss haben, jedoch nicht auf die TrgOps der Datenattributen - die sind fix im SCL vorgelegt, bestimmen jedoch als erste ob das IED die Änderung des Wertes des Datenattributes als trigger für Reports berücksichtigen soll.

TRIGGER OPTIONS DES RCB

In der Treiberkonfiguration, mit der Checkbox **Use preconfigured (SCL) options** konfigurieren Sie ob der Treiber der RCB aktivieren (enable) soll mit bereits in der SCL-Datei des IED vordefinierten TrgOps (und IntgPg, BufTm und OptFlds) oder dass der Treiber die RCB Felder zuerst mit Einstellungen aus der Treiberkonfiguration überschreiben soll und erst dann - RCB anmelden.

Wenn also die Checkbox **Use preconfigured (SCL) options** nicht aktiviert ist, sendet der Treiber die konfigurierte Trigger Options zum 850-Server. Eventuelle Schreibfehler werden dabei ignoriert und der RCB wird weiterhin verwendet. **Hinweis**: beim Schreibfehler auf RCB.TrgOps würde zu erwarten dass der 850-Server dann anhand seiner TrgOps aus SCL-Datei agiert.

Die Trigger Options im RCB definieren, unter welchen Bedingungen ein IED (der 850-Server) ein neuer Report erstellt und sendet:

- **TrgOp: data-change** - ein IED erstellt ein neuer Report wenn einer der Datenattribute, die in deren SCL-Definition die Flag dchg oder dupd haben, der Wert ändert. Zum Beispiel: *stVal*.
- **TrgOp: quality-change** - ein IED erstellt ein neuer Report wenn einer der Datenattribute mit Flag qchg sein Wert ändert. Laut Standard nur Datenattribute *q* dürfen qchg haben.
- **TrgOp: data-update** - ein IED erstellt ein neuer Report wenn einer der Datenattribute mit Flag dupd aktualisiert wird; auch wenn der Wert gleich bleibt (und eventuell nur der Zeitstempel aktualisiert wurde). Zum Beispiel: *mag*.

  Hinweis: für TrgOps: dchg, qchg und dupd sammelt ein IED die Wertänderungen einzelnen Datenattributen über die **Buffer Time** (RCB.BufTm) lang, um die Anzahl der erstellten Reports zu minimieren. Das BufTm gilt für URCB (Unbuffered) und BRCB (Buffered). Bei Verbindungverlust erstellt ein IED die Buffered Reports, jedoch sendet sie nicht. Die gepufferte Reports werden erst nach Verbindungs-Wiederaufbau gesendet wenn der Client (der Treiber) sie - mit einer gültigen EntryID - anfordert.
- **TrgOp: integrity** - ein IED erstellt zyklisch - je **Integrity Period** - ein Report der alle Werte beinhaltet; also auch die, die bereits wegen dchg, qchg oder dupd gesendet worden sind und die, die sich inzwischen nicht geändert haben.
- **TrgOp: general-interrogation** - ein IED erstellt ein Report der alle Werte beinhaltet wenn der Client (der Treiber) die **Generalabfrage** anfordert (RCB.GI).

Hinweis: Nicht alle Server unterstützen TrgOps **data-change** und **data-update** zusammen. Auch TrgOp **integrity** kann zu unnötiger Überlastung der Kommunikation führen wenn für RCB eine zu kurze IntgPd (**Integrity Period**) im 850-Server definiert wurde. Im Zweifelsfall setzten Sie TrgOps: **data-change** + **quality-change** + **general-interrogation**. Achten Sie darauf, dass TrgOp: **general-interrogation** aktiviert ist. Anderenfalls kann der Treiber beim Initialstart der Runtime keine Werte für im Report vorhandene Variablen empfangen.

BTYPE TRIGGERCONDITIONS

Der *TrigerConditions* Datenattribut-Typ, der gemäß dem Standard dem *MMS Bitstring(6)* zugeordnet wird (wenn eine entsprechende *TrgOps[RP|BR] Variable beim Import angelegt wurde) entspricht in zenon per default dem Datentyp UDINT (motorola-format, Bit 0 höchstwertig in seinem Oktet). Zusätzlich ermöglicht der Treiber solche Variablen mit dem Datentyp STRING anstatt UDINT. Der Wert des STRING hat die Länge 6 und enthält nur die Zeichen "0" und "1"; in der gleichen Reihenfolge wie die Bits im Standard definiert sind.

Beispiel

Wert als String "011111" (als UDINT=124=01111100b) bedeutet, dass alle TriggerCondition-Bits (außer Bit 0) - '*Reserved*' gesetzt sind.

Der Wert "000010" (als UDINT=8=00001000b) bedeutet, dass nur das Bit *4* - '*intergity*' gesetzt ist.

REDUNDANTES ZENON NETZWERK

In redundanten zenon Netzwerk, sowohl der **Prozessführende Server** und der **Standby Server** starten beide eigenen IEC850 Treiber. Die beide Treiber sollen unterschiedliche RCB Instanzen anmelden (sonst stehen sie im Konflikt weil ein 850-Server nur einem Client die RCB Instanz vergeben darf).

Achtung: Wenn Sie in Ihrem zenon Projekt Redundanz verwenden (Eigenschaft **Netzwerk aktiv** aktiviert sowie **Server 1** und **Server 2** konfiguriert), müssen Sie in der Treiberkonfiguration - **RCB assiggment** - wenigstens zwei Einträge konfigurieren: einen **Hostname** für den **Server 1** und zweiten **Hostname** für den **Server 2** (Standby). Für diese Einträge müssen Sie dann unterschiedliche Instanzen der RCBs auswählen.
Hinweis: Per Default werden vom IEC850 Treiber bereits zwei Hostnamen angelegt: *$SCADA_SERVER1* und *$SCADA_SERVER2*.

Die *Connection State*-Variable kann für Treiber am Netzwerk **Server 1** andere Auswertung ergeben als am **Server 2**. Für dem Benutzer, auch am Standby, ist nur sichtbar der aktuelle Wert des aktuellen **Prozessführenden Servers**. Wenn Sie auch die aktuelle Auswertung für **Standby Server** sichtbar machen möchten - um z.B. Alarmieren falls der Standby seine RCBs nicht anmelden konnte, können Sie eine Kopie der *Connection State*-Variable erstellen - mit gleicher Referenz (in **Symbolische**

Adresse), anderem Präfix in der Variableneigenschaft **Name** und mit aktivierter Eigenschaft **Nur von Standby Server anfordern**.

Hinweis: Für **Bewertetes Netzwerk** brauchen Sie keine von Standby angeforderte Variablen in der Eigenschaft **Bewertungen**. Die Bewertung findet am beiden Server separat statt, anhand der lokalen Werten der Variablen. Also am Standby wird immer der Wert der *Connection State*-Variable von seinen internen Abbild ausgewertet, auch wenn im Projekt keine Kopie erstellt wurde.

9.1 Unbuffered Reporting

Die Nutzung von Unbuffered Report Control Blocks (URCB) kann im IEC850 Treiber in zwei Varianten konfiguriert werden:

1. automatische Zuweisung:
   Eigenschaft **max. auto used URCBs**:
   die automatische Zuweisung erleichtert die Nutzung von Reporting bereits in der Testphase des Projekts - der Treiber überprüft alle Vorhandenen Unbuffered Report Control Blocks (URCB) auf ihren Inhalt (Data Sets). Dann meldet der Treiber an, zuerst die URCBs, welche die meisten Variablen des Projektes enthalten;
2. statische Zuweisung - Dialog **statically assigned RCB** (auf Seite 47):
   der Treiber verwendet die Zuweisung von Report Control Blocks zu seinem Rechner - **Hostname** - wie es in Treiberkonfiguration - Client configuration (auf Seite 41) definiert ist. Dies gilt für URCB und BRCB.

AUTOMATISCHE ZUWEISUNG

Die automatische Zuweisung arbeitet mit Data Sets und verknüpfungen von Data Sets zu Report Control Blocks die im 850-Server bereits vordefiniert sind - anhand seiner SCL-Datei.

Während des Verbindungaufbaus zum 850-Server liest der Treibers alle Data Sets, die im 850-Server vorhanden sind, und ordnet sie gemäß ihrer Nutzbarkeit - entsprechend der Anzahl der inkludierten Variablen des Projekts. Diese Sortierung gibt die Reihenfolge an, in der der Treiber versucht freie URCBs fürs Reporting zu aktivieren (enable). Es werden höchstens so viele URCBs pro Logical Device aktiviert wie in der Treiberkonfiguration - Server (auf Seite 29) in Eigenschaft **max. auto used URCBs** definiert sind. Alle Datenobjekte des Data Sets, die erfolgreich mit einem aktivierten URCB verknüpft sind, werden nicht länger gepollt, sondern werden bei einer Wertänderung gemeldet.

Die **Dynamische Datasets** werden für automatische Zuweisung nicht unterstützt.

Hinweis: die automatische Zuweisung erleichtert die Nutzung von Reporting bereits in der Testphase des Projekts - noch bevor die statische Zuweisungen vorgenommen wurden. Jedoch bevor Sie das Projekt freigeben, konfigurieren Sie die statische Zuweisung auch für URCB - im Dialog **RCB assignment** - und setzen Sie die Eigenschaft **max. auto used URCBs** auf 0. Der Standard schlagt vor

dass in einem System sollte klar geregelt werden welcher Client bestimmte Instanz eines RCB verwendet. Es beugt vor eventuelle Konflikte zwischen mehreren 850-Clients.

REGELN FÜR AUFTRETENDE KONFLIKTE

Bei der Aktivierung des Reporting, erkennt der Treiber auch URCBs mit gleichem Inhalt; die Namen dieser Reports können unterschiedlich sein. Bevor die URCBs gemäß ihrer Nutzbarkeit für das zenon Projekt sortiert werden, sortiert der Treiber zuerst die Datenobjekte der Data Sets von jedem URCB und vergleicht sie, um Dubletten zu vermeiden. Das folgende gilt für Konflikte bei der Aktivierung der URCBs:

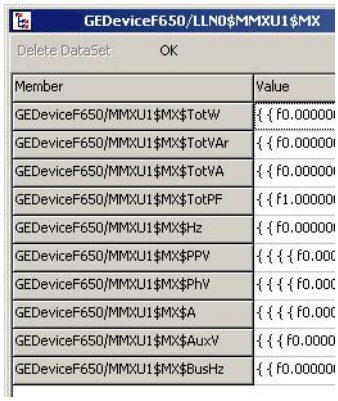
- Für gleiche Reports - mit identischer oder unterschiedlicher Reihenfolge der gleichen Datenobjekte: nur ein URCB wird aktiviert.
- Wenn ein URCB schon von einem anderen Client belegt ist, wird der nächste freie URCB aktiviert.
- Wenn alle URCBs schon von anderen Clients besetzt sind, werden die Datenpunkte gepollt.

BEISPIEL FÜR AUTOMATISCHE ZUWEISUNG

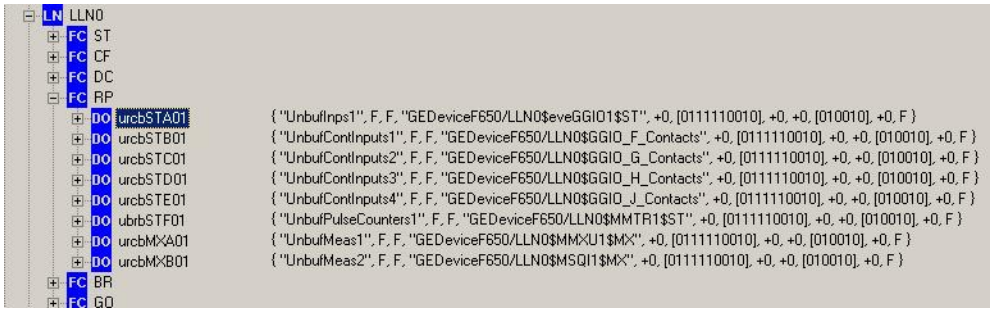
Die folgenden Variablen sind in ihrem Projekt verfügbar

Name	Datentyp
Filtertext	Filter...
Server1!GEDeviceF650/LLN0/urcbMXA01/BufTm[RP]	UDINT
Server1!GEDeviceF650/LLN0/urcbMXA01/ConfRev[RP]	UDINT
Server1!GEDeviceF650/LLN0/urcbMXA01/DatSet[RP]	STRING
Server1!GEDeviceF650/LLN0/urcbMXA01/GI[RP]	BOOL
Server1!GEDeviceF650/LLN0/urcbMXA01/IntgPd[RP]	UDINT
Server1!GEDeviceF650/LLN0/urcbMXA01/OptFlds[RP]	UDINT
Server1!GEDeviceF650/LLN0/urcbMXA01/Resv[RP]	BOOL
Server1!GEDeviceF650/LLN0/urcbMXA01/RptEna[RP]	BOOL
Server1!GEDeviceF650/LLN0/urcbMXA01/RptID[RP]	STRING
Server1!GEDeviceF650/LLN0/urcbMXA01/SqNum[RP]	USINT
Server1!GEDeviceF650/LLN0/urcbMXA01/TrgOps[RP]	UDINT
Server1!GEDeviceF650/MMXU1/Hz/mag.f[MX]	REAL
Server1!GEDeviceF650/MMXU1/PhV.phsA/cVal.ang.f[MX]	REAL
Server1!GEDeviceF650/MMXU1/PhV.phsA/cVal.mag.f[MX]	REAL
Server1!GEDeviceF650/MMXU1/PhV.phsB/cVal.ang.f[MX]	REAL
Server1!GEDeviceF650/MMXU1/PhV.phsB/cVal.mag.f[MX]	REAL
Server1!GEDeviceF650/MMXU1/PhV.phsC/cVal.ang.f[MX]	REAL
Server1!GEDeviceF650/MMXU1/PhV.phsC/cVal.mag.f[MX]	REAL
Server1!GEDeviceF650/MSQI1/SeqA.c1/mag.f[MX]	REAL
Server1!GEDeviceF650/MSQI1/SeqA.c2/mag.f[MX]	REAL
Server1!GEDeviceF650/MSQI1/SeqA.c3/mag.f[MX]	REAL
Server1!GEDeviceF650/XCBR1/Pos/Oper.ctlVal[CO]	BOOL
Server1!GEDeviceF650/XCBR1/Pos/stVal[ST]	UDINT

Die **/urcbMXA01/** Variablen haben rein informativen Zweck. Damit Reporting funktioniert, müssen die Variablen für die RCB Attribute nicht zwingend im zenon Projekt existieren. Das IED enthält einige Data Sets, z.B. ein Data Set *GEDeviceF650/LLN0$MMXU1$MX* mit dem folgenden Inhalt:



Im IED ist dieses Data Set dem RCB *LLN0RPurcbMXA01* zugewiesen - der Name des Data Set steht im Attribut DatSet dieses RCBs.



Beim Treiberstart werden die URCBs *LLN0RPurcbMXA01* und *LLN0RPurcbMXB01* angemeldet (wenn sie noch frei sind).
Da es keine weiteren geeigneten URCBs gibt, die Variablen des Projekts in ihren Data Sets haben, bleiben die übrigen URCBs ungenutzt. Wenn das Projekt auch eine Variable enthält, die im Data Set *GEDeviceF650/LLN0$eveGGIO1$ST* inkludiert ist, würde das bedeuten, dass *LLN0RPurcbSTA01* auch aktiviert wird.

STATISCHE ZUWEISUNG

Für statischer Zuweisung ist in der Treiberkonfiguration - **statically assigned RCB** (auf Seite 47) fest definiert, welche Report Control Blocks benutzt werden. Der Treiber versucht, alle gelisteten RCBs, unabhängig deren Inhalt, zu aktivieren.
Wenn in der Treiberkonfiguration auch ein Dynamic Data Set eingegeben wurde, weist der Treiber in der Steuerung diese Data Set dem RCB zu, indem er den Wert des DatSet Attribut dieses RCB ändert bevor der Report aktiviert wird.
Wenn kein Dynamic Data Set eingegeben wurde, ist das in der Steuerung vordefiniert Data Set gültig (genauso wie bei der automatischen Zuweisung).

Wenn ein RCB erfolgreich aktiviert wurde, werden alle Datenobjekte, die in seinem Data Set enthalten sind, nicht länger gepollt. In diesem Fall erwartet der Treiber, dass das IED die Wertänderungen spontan meldet.

Wenn im empfangenen Report kein Datasetname mitgeschickt wird, wird das entsprechende Dataset anhand von RptID gefunden. Dies ist nur dann möglich, wenn die RptIDs eindeutig sind.

Info

Der Treiber kann die in einem Report empfangen Daten nur dann Variablen zuordnen, wenn im Report ein Datasetname vorhanden ist oder wenn die RptID des Reports eindeutig ist.

Enthält der Report keinen Datasetnamen, haben Sie zwei Möglichkeiten:

- Ein IED kann Reports ohne Datsetnamen nur dann schicken, wenn im RCB das **Optional Field** *Data set name* nicht aktiviert ist.
  Ob **Optional Field** *Data set name* dynamisch aktivierbar ist, ist abhängig vom 850-Server.
- Jeder Report beinhaltet RptID.
  Wenn die IDs bei jedem RCB eindeutig sind, ist dies ausreichend um die Daten korrekt auszuwerten.
  Ob RptID dynamsich änderbar ist, ist abhängig vom 850-Server.
- In der SCL-Datei des 850-Servers, in der Sektion für Services, wird bestimmt, ob die Optional Fields und RptID dynamisch änderbar sind. Zur Laufzeit kann ein Client nur dann Änderungen durchführen, wenn der Server dies zulässt.

Weiterführende Informationen über statische Zuweisung der RCBs finden Sie im Kapitel über **Buffered Reporting**.

9.2 Buffered Reporting

Bei Buffered Reports speichert ein 850-Server (IED) die Änderungen in der Zeit, in der ein angeschlossener 850-Client nicht erreichbar ist. Wenn dann der Client (der IEC850 Treiber) wieder online ist (z.B. Netzwerkverbindung wiederhergestellt), liest der Treiber alle gepufferte Änderungen von IED und verarbeitet sie, z.B. erzeugt entfallene Alarme.

Bei Buffered Reporting gibt es keine **automatische Zuweisung**. Die Zuweisung von BRCBs zum Treiber ist immer statisch:

- entweder über die Treiberkonfiguration, so wie im Abschnitt über die statische Zuweisung für Unbuffered Reporting (auf Seite 132) beschrieben;
- oder aus Kompatibilitätsgründen zu älteren Versionen des Treibers/IEDs (aus Anfangszeiten der Edition 1 des Standards), über die *Report ID* der BRCBs die in der SCL-Datei projektiert wurden und bestimmten Text beinhalten. Dafür die Eigenschaft **Use Report-ID for RCB**

assignment in der Treiberkonfiguration sollte aktiviert sein.
Wenn die Zuweisungen in der SCL Datei projektiert werden sollen, muss die zugewiesene Report ID (RptID) der BRCB einen String mit folgender Nomenklatur enthalten 'zenon_Computername_*'; z. B.: zenon_MyPC_Measurands1. Dann meldet der Treiber das RCB an und benutzt das Data Set von RCB, das in der SCL-Datei definiert ist.

IEC 61850 Standard definiert dass ein Buffered Report kann nur vom einen Host (einen 850-Client) aktiviert werden. Um den Treiber davon abzuhalten die Instanzen des Buffered Report Control Blocks zu aktivieren, die den Zugriff für andere 850-Clients blockieren, müssen die richtigen Instanzen explizit in der Treiberkonfiguration (auf Seite 9) zugewiesen werden.

Abgesehen von gepufferten Events ist das grundlegende Verhalten gleich wie im Abschnitt über **Unbuffered Reporting - statische Zuweisung** beschrieben.

Info

Nach dem Start der Runtime liest der Treiber zuerst die RCB-Atrribute und schreibt zum Enablen nur, wenn der RCB:

- frei ist
- noch nicht enabled ist (z.B. von anderen 850-Client)
- nicht reserviert ist
- ein gültiges Dataset beinhaltet

Bei Wiederholungen - von Eigenschaft **RCBs enable retries** - liest der Treiber den RCB nicht mehr. In diesem Fall schreibt er immer RCB.RptEna=TRUE und überprüft dann, ob die IED den Erfolg bestätigt oder einen Fehler zurückgibt.

BRCB PUFFER HOLEN ODER LÖSCHEN

Bei erneuter RCB Aktivierung (**RCB enable**) des vorher benutzten BRCB (= erneuter Aufbau der Verbindung nach Verbindungsverlust), sendet der Treiber zum 850-Server ein gespeichertes **EntryID** um die gepufferte Wertänderungen vom IED zu holen. Diese Vorgehensweise ist so im IEC 61850 Standard definiert. Die Option **Do not purge BRCB buffer at start** hat keinen Einfluss darauf.

Beim ersten Versuch (= Start der Runtime) einen Report zu enablen, kann der Treiber den Wert der **EntryID** ebenfalls senden. Der Treiber macht dies jedoch nur dann, wenn die Option **Do not purge BRCB buffer at start** aktiviert ist und wenn der Treiber vorher ein **EntryID** gespeichert hat.

Tipp: Der Treiber speichert die zuletzt erhaltene **EntryID** per jedes *ObjectReference* zu BRCB zuerst intern und dann - jede Minute oder wenn die Runtime geschlossen wird - in einer TXT-Datei im Verzeichnis der Runtime-Daten.

In folgenden Situationen jedoch, schickt der Treiber das Kommando **purge buffer** um den Puffer am IED zu löschen:

- Es handelt sich um die erste Verbindung nach dem Start der Runtime und die Option **Do not purge BRCB buffer at start** im Basic settings (auf Seite 18) ist deaktiviert.
- Der Treiber hat kein gültiges **EntryID** Wert vorher empfangen (bzw. die TXT-Datei beinhaltet kein EntryID).
- Es handelt sich um einen wiederholten Versuch, einen Report zu anmelden - nach Ablauf der **RCBs enable retries** Zeit. Bei einem wiederholten Versuch wird das BRCB.PurgeBuf Kommando immer geschickt. Damit werden Daten-Dubletten vermieden, da die Werte bereits durch das Polling empfangen wurden. Die Option **Do not purge BRCB buffer at start** hat keinen Einfluss auf Wiederholungen.

> **Achtung**
>
> Ein 850-Server ist verpflichtet zuerst das Schreiben von **EntryID** zu bestätigen und erst dann die gepufferte Werte zu senden. Solange die BRCB Aktivierung noch nicht bestätigt wurde, werden die Reports ignoriert.

BUFFERED REPORTS UND ZENON REDUNDANZ

Da zenon Redundanz unterstützt (siehe zenon Netzwerk) und ein BRCB nur einmal von einem PC aktiviert werden kann, müssen Sie für das Projekt **Server 1** und **Server 2** (Standby Server) getrennt festlegen, welcher PC welchen Report aktivieren soll. Der Treiber wird sowohl auf dem Prozessführenden Server als auch auf dem Standby gestartet und überprüft die Treiberkonfiguration (Eigenschaft **Hostname**), ob er eine Konfiguration findet, die dem Computernamen entspricht, auf dem die Runtime läuft.

Darum benötigen Sie in dem IED (850-Server) zwei Kopien des Buffered Report im Fall einer redundanten Konfiguration; eine für den Treiber, der auf dem Projekt-Server läuft und eine für den Treiber, der auf dem Standby Server läuft. Und in der Treiberkonfiguration erstellen Sie zwei Einträge in der Liste RCB assignment (auf Seite 47). Einen mit dem Computernamen des Projekt-Servers und einen mit dem Computernamen des Standby Servers.

Hinweis: Beim Konfigurationseintrag für Rechnername von **Server 1** können Sie die erste Instanz des BRCB eintragen (z.B. "brcbA01") und beim Konfigurationseintrag von **Server 2** die zweite Instanz (z.B. "brcbA02").

> **Achtung**
>
> Auch wenn Sie keine Redundanz verwenden, müssen Sie in der Treiberkonfiguration (Eigenschaft **Hostname**)dennoch festlegen welche Reports für den PC verwendet werden, auf dem die zenon Runtime läuft.

BRCB.RESVTMS HANDLING

Der Treiber behandelt das Datenattribut *BRCB.ResvTms* nur dann, wenn in der Treiberkonfiguration kein **ClientLN.iedName** definiert wurde. Diesen **ClientLN.iedName** konfigurieren Sie im Treiberdialog Client configuration (auf Seite 41) oder mit dem **IEC850 Driver Configuration** Wizard aus einer SCD-Datei.

Wurde **ClientLN.iedName** konfiguriert, dann wird angenommen, dass auch der 850-Server mit der gleichen SCD-Datei bereits vorkonfiguriert wurde, also ist der 850-Server verpflichtet für alle exklusiv reservierte BRCBs das *ResvTms* auf -1 setzen und die BRCBs nur auf die vorgesehene Clients zu vergeben.

Ist in der Treiberkonfiguration kein **ClientLN.iedName** definiert, werden beim Start der Verbindung zum Server, jene BRCBs, die zu enablen sind, auf das Vorhandensein des optionalen Attributs *ResvTms* geprüft. Dabei bedeutet der Wert -1 im *ResvTms*, dass dieser BRCB, in der SCD-Datei für einen anderen Client exklusiv reserviert ist. In diesem Fall liegt eine Fehlkonfiguration vor und eine Warnung wird im LOG ausgegeben.

Mögliche Fehlerursache:

- Treiber wurde nicht mit richtigem *Client IED name* konfiguriert.
- Im Treiber wurden BRCBs statisch konfiguriert, welche jedoch laut SCD-Datei für einen anderen Client reserviert sind.

Wurde im Treiber kein **ClientLN.iedName** konfiguriert, setzt der Treiber das Attribut *ResvTms* auf 32767 Sekunden (7fff hex) um diesen BRCB im Falle eines Verbindungsausfalls für die längst mögliche Zeit zu reservieren. Diese Zeit beträgt ca. 9 Stunden und sorgt dafür dass während Verbindungsabbruch kein andere 850-Client diesen BRCB anmelden könnte.

URCB.RESV HANDLING

Der Treiber beachtet das Datenattribut *URCB.Resv* nur dann, wenn in der Treiberkonfiguration kein **ClientLN.iedName** definiert wurde.

9.3 RCB Aktivierung - Überprüfungsmöglichkeiten

Um herauszufinden, ob ein bestimmter RCB aktiviert wurde, lesen Sie den Wert des Datenattributs **RptEna[RP/BR]* aus.

Ist der Wert dieser Variable *TRUE*, dann ist der RCB von einem Client zugeordnet. Diese Information ist nur dann aussagekräftige und zuverlässig, wenn nur ein einziger Client mit dem 850-Server (IED) kommuniziert.

Anderenfalls sehen Sie zwar, dass der Report enabled ist, es gibt aber keine Information von welchem Client dieser Report aktiviert wurde. Eine zusätzliche Information darüber, dass einer der RCBs nicht aktiviert werden konnte, kann aus den Werten der **Connection State**-Variable (auf Seite 98) ausgelesen werden.

Wenn der Treiber aus irgendeinem Grund die statisch zugewiesene Reports nicht aktivieren kann, wird stattdessen - für Variablen aus fehlenden RCB - Polling benutzt. In diesem Fall wird in der **Connection State**-Variable das Bit 17 oder 25 (*MMS_RCB_ENABLE_FAILED*) gesetzt und der Treiber wird - nach Ablauf **RCBs enable retries** Zeit - erneut versuchen das Datenattribut *RptEna* mit *true* zu beschreiben. Sobald der 850-Server, bei einen erneuten Versuch, die fehlende Anmeldungen für RCB akzeptiert, pollt der Treiber die Variablen nicht mehr. Nach Anmeldung aller RCBs wird das Bit *MMS_RCB_ENABLE_FAILED* der **Connection State**-Variable zurückgesetzt.

DIAGNOSE VIEWER

Mehr Details erhalten sie, wenn Sie die LOG-Datei mit dem **Diagnose Viewer** auslesen. In der LOG-Datei werden die entsprechenden Informationen über das Enabling der Reports vom Treiber (IEC850 Client) auf der *MSG*-Ebene protokolliert. Der dazugehörige LOG-Eintrag:

- "*Trying to enable xxx on device nnn*".
  Beispiel für xxx: *LLN0BRbrcbA01*.

Standardmäßig im **Diagnose Viewer** wird nur der ERROR-Level protokolliert. Sie müssen im **Diagnose Viewer** den entsprechenden **Messagelevel** konfigurieren. Dies Konfiguration im **DiagViewer** ist nur dann möglich, wenn die entsprechenden Treiber-Applikationen bereits gestartet wurde. Dies ist dann der Fall, wenn die Runtime läuft oder im zenon Editor die Treiberkonfiguration geöffnet ist.

Ist der *MSG*-Level im **Diagnose Viewer** aktiviert aber der LOG-Eintrag "*Trying to enable xxx on device nnn*" nicht vorhanden, dann stellen Sie sicher, dass die Eigenschaft **Hostname** der Treiberkonfiguration korrekt konfiguriert ist.

Diese Konfiguration muss entweder mit einem Schlüsselwort *$SCADA_SERVER1* (*$SCADA_SERVER2*) oder mit dem Rechnernamen, auf dem der Treiber läuft (= zenon Runtime) konfiguriert sein.
Hinweis: Die **Hostname** Konfiguration projektieren Sie im Treiberdialog **Client configuration**. (auf Seite 41)

Stellen Sie auch sicher, dass in ihrem zenon Projekt mindestens eine **[ST|MX]* Variable vorhanden ist deren Werte per diesem Report gemeldet werden.

Weitere LOG-Einträge:

- Enabling des Reports wird vom 850-Server verweigert - 'ERROR'-LOG-Eintrag mit der Nummer des MMS Fehlercodes, wie vom Server empfangen.
- Enabling des Reports wird vom 850-Server akzeptiert:
  - 'MSG'-LOG-Eintrag mit Text: "*Successfully enabled xxx on device yyy*"
  - 'DEBUG'-LOG-Einträge mit der Liste der Datenobjekte mit Text: "*Disabled ST polling for zzz on device nnn*".
    Beispiel für zzz: **/XCBR1STPos*.
    Diese Datenobjekte werden nun über Report erwartet und nicht mehr gepolled.

> **Info**
>
> Ist der Treiber für die Nutzung von bestimmten RCBs konfiguriert und haben diese RCBs falsche Dataset-Definition, wird der Report zwar aktiviert, aber einige Datenobjekte müssen immer noch gepollt werden. In der LOG-Datei können Sie Einträge über das Enabling von RCB finden, jedoch nicht die Einträge über deaktiviertes Polling.

CAUSE OF TRANSMISSION (COT)

Für jede Variable stellt der Treiber die Information zur Verfügung, ob der Wert der Variable pollend oder per Report empfangen wurde. Die Information wird auf die Statusbits *COTx* geschrieben.

COT Wert	Statusbits	Beschreibung
1	COT0	Der Wert wurde via Polling erhalten
2	COT1	Der Wert wurde über einen Integrity Report erhalten. **Hinweis:** Dies funktioniert nur, wenn Reason-for-inclusion im Dialog Optional Fields (auf Seite 40) aktiviert ist. Ansonsten ist die COT=3
3	COT0, COT1	Der Wert wurde über einen Report erhalten. **Hinweis:** Ist Reason-for-inclusion im Dialog Optional Fields (auf Seite 40) aktiviert, sind es nur Reports mit Triggeroption data-change, data-update oder quality-change. Ist Reason-for-inclusion nicht aktiviert, sind es alle Reports - auch Integrity und GI.
20	COT2, COT4	Der Wert wurde über einen General Interogation Report erhalten. **Hinweis:** Dies funktioniert nur, wenn Reason-for-inclusion im Dialog Optional Fields (auf Seite 40) aktiviert ist. Ansonsten ist die COT=3

> **Info**
>
> Der COT-Wert für den IEC850 Treiber ist an den IEC60870 Standard angelehnt:
>
> 1 - periodic, cyclic
>
> 2 - background scan
>
> 3 - spontaneous
>
> 20 - general interrogation

> **Info**
>
> Sie können *COTx* Statusbits auswerten, indem Sie Combi-Element oder Reaktionsmatrix (vom Typ Multi) verwenden. Sie sind auch in **Variablen-Diagnose** Bild ersichtlich.

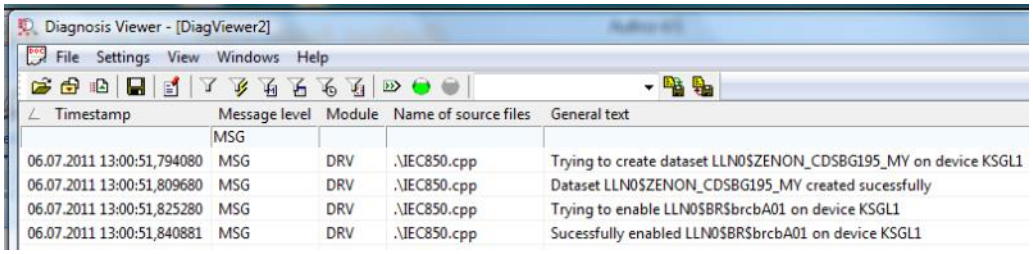
9.4 Dynamische Datasets

Um Dynamic Data Sets zu benutzen, müssen die beiden Dienste *CreateDataSet* und *DeleteDataSet* vom IEC 61850 Server unterstützt werden und das Attribut DatSet muss in dem Reporting Control Blocks (RCB) schreibbar sein (SCL: ReportSettings.datSet="Dyn"). Beim Projektieren müssen Sie mögliche Einschränkungen des Servers berücksichtigen wie zum Beispiel maximale Anzahl von Elementen im Data Set (SCL: DynDataSet.max und maxAttributes).

Dynamic Data Sets listen Datenobjekte auf und werden im Dialog dynamic DataSet configuration (auf Seite 53) konfiguriert. Sie können für Unbuffered Reporting und für Buffered Reporting verwendet werden, aber nur, wenn in der Treiberkonfiguration - Client-Konfiguration (auf Seite 41) statische Zuweisungen mit RCBs projektiert wurden. Wenn kein Dynamic Data Set in der RCB Konfiguration des Treibers verknüpft wurde, ist das in dem RCB des IEDs vordefiniertes, statisches Data Set gültig.

Wenn in der Treiberkonfiguration ein Dynamic Data Set für ein RCB eingeben wurde, weist der Treiber in dem IED (im 850-Server) dieses Data Set dem RCB zu, indem er den Wert des DatSet Attribut dieses Blocks ändert bevor der Report aktiviert wird.

Im Diagnosis Viewer (auf Seite 152) kann das Anlegen eines dynamischen Datasets überprüft werden:

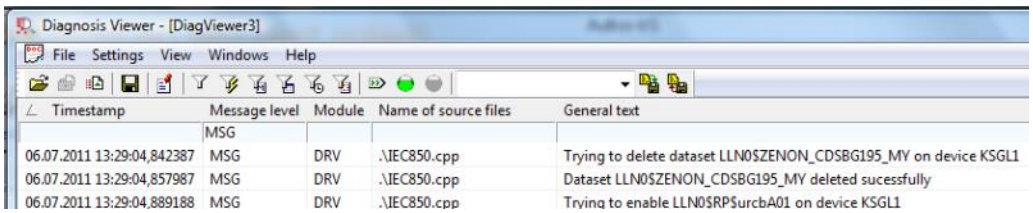


Kann ein dynamisches Datenset nicht erstellt werden, so werden alle RCBs, die dieses dynamische Datenset verwenden, in der Runtime nicht aktiviert. Wenn das Datenset erfolgreich in dem IED erstellt wurde, jedoch kann der Treiber das Attribut DatSet in RCB nicht aktualisieren, wird der zugewiesene RCB aktiviert mit bereits vorhandenen Datenset.

Bevor die statisch zugewiesenen Reporting Control Blocks aktiviert sind, werden alle dynamischen Data Sets, die mit ihnen via Treiberkonfiguration - Client-Konfiguration (auf Seite 41) verknüpft sind, mit denen auf dem 850-Server verglichen. Alle beim Server verfügbaren Dynamic Data Sets, die nicht (mehr) benötigt werden, werden gelöscht. Alle beim Server nicht verfügbaren Dynamic Data Sets werden erstellt und alle Data Sets die auf beiden Seiten verfügbar sind, werden auf selben Inhalt

überprüft. Wenn der Inhalt abweicht, werden sie gelöscht und neu erstellt. Der Treiber löscht aber nur die Data Sets die er selbst mal erstellt hat.

Im Diagnosis Viewer (auf Seite 152) kann das Löschen eines dynamischen Datasets überprüft werden:



Die Dynamic Data Sets, die der Treiber am IED erstellt (via *CreateDataSet* Dienst), erhalten einen Client-abhängigen Namen am IED, um sie leicht wieder zu finden. Die Nomenklatur des Dynamic Data Set Namens ist wie folgt:

LLN0$ZENON_hostname_datasetname. Hierbei:

Hostname ist der Computername des Clients, so wie er in der Treiberkonfiguration definiert ist. Diese Konfiguration finden Sie in der Client-Konfiguration (auf Seite 41) unter **Hostname**.
datasetname ist der Name, der in der Treiberkonfiguration definiert wurde.
Diese Konfiguration finden Sie in der Client-Konfiguration (auf Seite 41) bei der Dynamic DataSet configuration (auf Seite 47).

Hinweis: Definieren Sie kurze Namen der Datensets und verwenden Sie nur alphanumerische Zeichen. Der RCB.DatSet Parameter hat laut IEC 61850 Standard Ed.2 die maximale Länge 32 (genauer: 64/16$32 - LD/LN$Name). Diese 32 Zeichen werden mit Präfix, Hostname und Name des Datensets belegt, somit bei längeren Namen können die übrige Zeichen abgeschnitten werden und nicht mehr eindeutig sein.

LOG-EINTRÄGE

LOG Eintrag	Debug Level	Beschreibung
Trying to delete dataset xxx on device yyy	MSG	der Treiber ist dabei ein vorher angelegtes und nicht mehr gebrauchtes dynamisches Dataset von 850-Server zu löschen.
Delete dataset xxx failed	ERRORS	Der 850-Server hat das Löschen des Datasets xxx verweigert.
Dataset xxx deleted sucessfully	MSG	Das Dataset xxx wurde vom Server erfolgreich gelöscht.

10 Treibermodus zur Runtime ändern

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

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

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

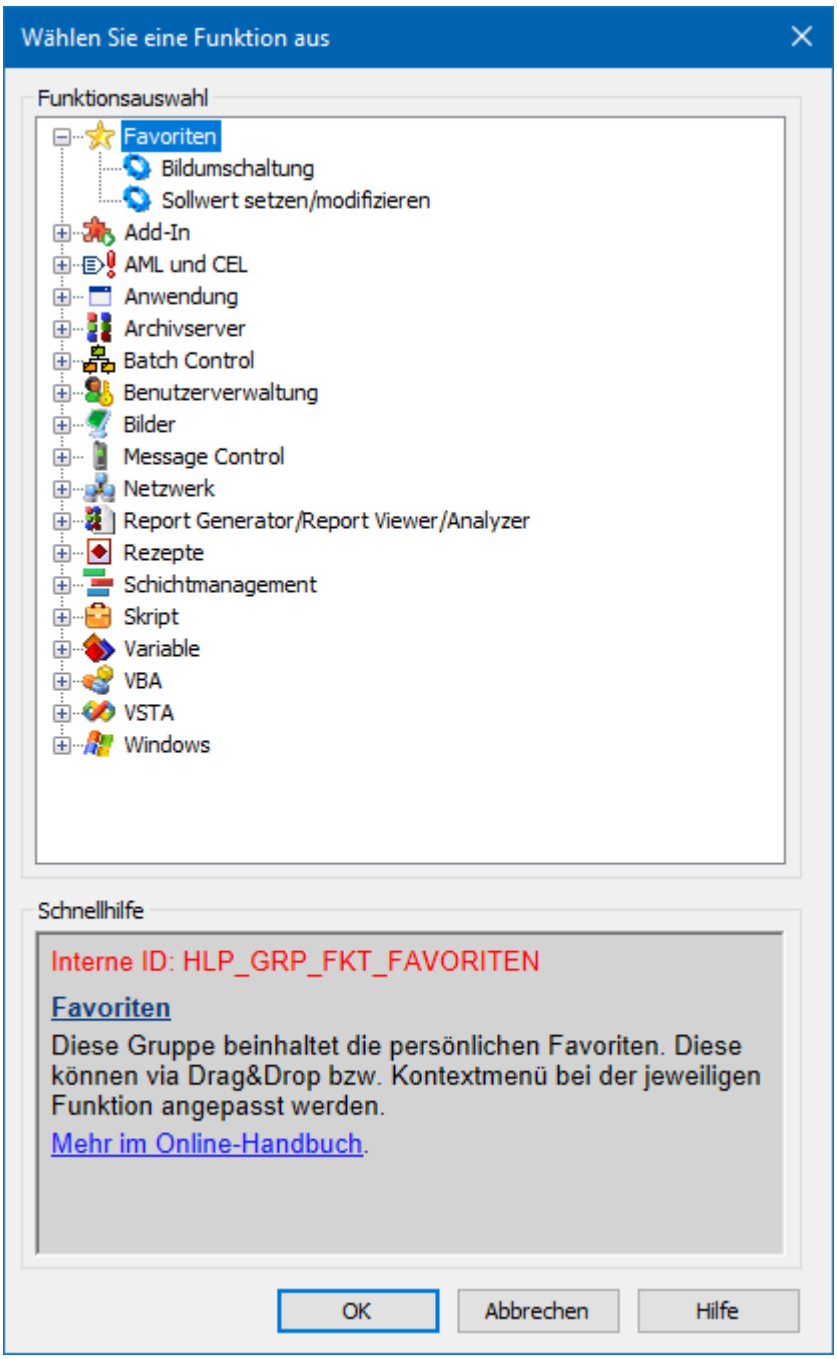
> **Achtung**
>
> Die zenon Funktion **Treiberkommandos** ist nicht ident mit den Treiberkommandos, die bei Energy-Treibern in der Runtime ausgeführt werden können!

PROJEKTIERUNG DER FUNKTION

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

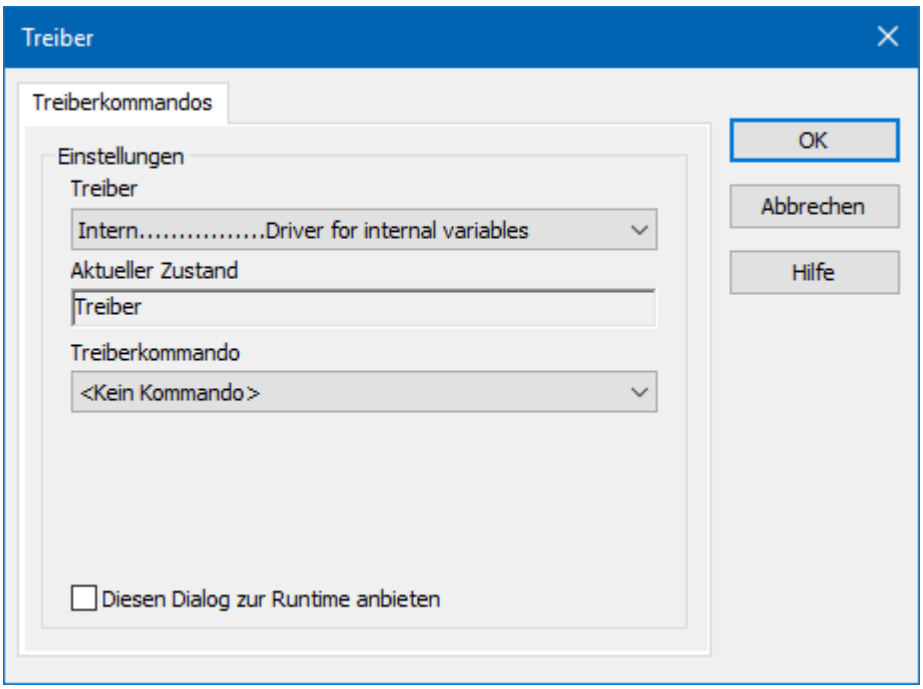
1. Legen Sie im zenon Editor eine neue Funktion an.

Der Dialog zur Auswahl einer Funktion wird geöffnet.



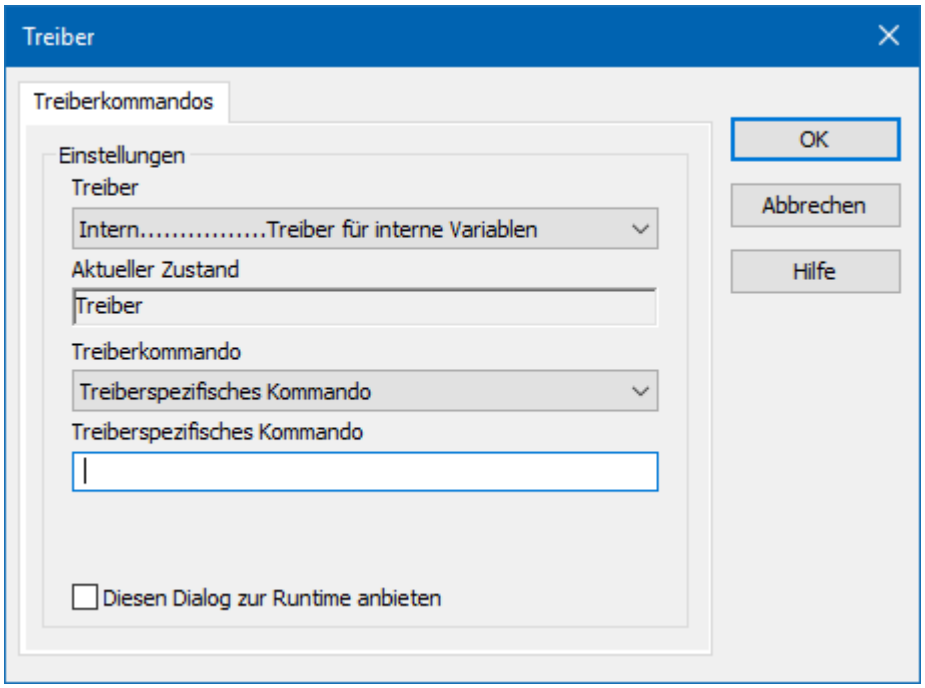
2. Navigieren Sie zum Knoten **Variable.**
3. Wählen Sie den Eintrag **Treiberkommandos**.

Der Dialog zur Konfiguration wird geöffnet.



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

DIALOG TREIBERKOMMANDOS



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

DIALOG BEENDEN

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

VERFÜGBARE TREIBERKOMMANDOS

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

Treiberkommando	Beschreibung
Kein Kommando	Es wird kein Kommando gesendet. Damit kann auch ein bereits bestehendes Kommando aus einer projektierten Funktion entfernt werden.
Treiber starten (Onlinemodus)	Treiber wird neu initialisiert und gestartet. **Hinweis:** Wenn der Treiber bereits gestartet wurde, muss er gestoppt werden. Erst dann kann der Treiber wieder neu initialisiert und gestartet werden.
Treiber stoppen (Offlinemodus)	Treiber wird angehalten, es werden keine neuen Daten angenommen. **Hinweis:** Ist der Treiber im Offline-Modus, erhalten alle Variablen, die für diesem Treiber angelegt wurden, den Status *Abgeschaltet* (*OFF*; Bit *20*).
Treiber in Simulationsmodus	Treiber wird in den Simulationsmodus gesetzt. Die Werte aller Variablen des Treibers werden vom Treiber simuliert. Es werden keine Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
Treiber in Hardwaremodus	Treiber wird in den Hardwaremodus gesetzt. Für die Variablen des Treibers werden die Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
Treiberspezifisches Kommando	Eingabe eines treiberspezifischen Kommandos. Öffnet Eingabefeld für die Eingabe eines Kommandos.
Treiber Sollwertsetzen aktivieren	Sollwert setzen auf Treiber ist möglich.
Treiber Sollwertsetzen deaktivieren	Sollwert setzen auf Treiber wird verhindert.
Verbindung mit Modem aufbauen	Verbindung aufbauen (für Modem-Treiber). Öffnet Eingabefelder für Hardware-Adresse und Eingabe der zu wählenden Nummer.
Verbindung mit Modem trennen	Verbindung beenden (für Modem-Treiber).
Treiber in Simulationsmodus zählend	Treiber wird in den zählenden Simulationsmodus gesetzt. Alle Werte werden mit *0* initialisiert und in der eingestellten Updatezeit jeweils um *1* bis zum Maximalwert inkrementiert und beginnen dann wieder

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

FUNKTION TREIBERKOMMANDOS IM NETZWERK

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

- Ein spezielles Netzwerkkommando wird vom Rechner zum Server des Projekts gesendet. Dieser führt dann die gewünschte Aktion auf seinem Treiber aus.
- Zusätzlich sendet der Server das gleiche Treiberkommando zum Standby des Projekts. Der Standby führt die Aktion auch auf seinem Treiber aus.

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

11 TLS-verschlüsselte Kommunikation

Eine verschlüsselte Kommunikation des IEC850 Treibers mit der PLC wird in Konfigurations-Datei des Treibers konfiguriert. Diese Konfigurationsdatei liegt per Default im Projektordner im Unterordner *\zenon\custom\drivers*.

Der Ablageort und die Benennung der Konfigurationsdatei kann auch im Treiberdialog in der Registerkarte **Basic settings** in der Option **Configuration File name** konfiguriert werden.

Tipp: Sie können im zenon Editor mit der Tastenkombination **Strg + Alt + E** in den Projektordner wechseln.

Diese Konfiguration muss folgende Einträge enthalten:

***** LINK *****

Die Parameter einer abgesicherten Verbindung können für jede Verbindung extra konfiguriert werden.

Parameter	Beschreibung
[TLS aktiviert]	Art der abgesicherten Kommunikation. ▸ *0: deaktiviert* Keine verschlüsselte Kommunikation. ▸ *1: aktiviert* Kommunikation erfolgt verschlüsselt, auf Basis der folgenden Parameter. Default: *0*
[Liste Verschlüsselungsverfahren]	Liste der unterstützten Verschlüsselungsverfahren. Die Liste enthält die Kürzel im openSSL-Format.
[Zertifikat]	Name des TLS-Zertifikates. Der Default-Ablageort wird mit dem Eintrag für den *[Ablageort der Zertifikate]* (**TLS_CERTIFICATE_STORE_PATH=)** projektiert. Das TLS-Zeritifikat muss dort im Ordner **PRIVATE** vorhanden sein.
[Zertifikat Identifizierung]	Verwendete Zertifikats-Identifizierung des Clients. Das Client-Zertifikat muss das hier konfigurierte **SUBJECT** enthalten. Bei Nicht-Übereinstimmung wird die Verbindung abgebrochen.
[Zeit bis zur Erneuerung des Schlüssels]	Maximale Dauer eines benutzten Schlüssels, bevor er erneuert wird. Es erfolgt nur eine Erneuerung des Schlüssels. Das Zertifikat wird dabei nicht aktualisiert. Nach Ablauf der projektierten Zeit wird das Zertifikat erneuert. Zeitangabe in Sekunden. Default: *43200*
[Datenmenge bis zur Erneuerung des Schlüssels]	Datenmenge, für die ein Schlüssel benutzt wird, bevor er erneuert wird. Nach Übertragung der projektierten Datenmenge wird die Verschlüsselung erneuert. Das Zertifikat wird dabei nicht aktualisiert. Angabe in Bytes. Default: *10485760*
[maximale Dauer einer verschlüsselten Verbindung]	Maximale Dauer einer verschlüsselten Verbindung, bevor diese erneuert wird. Nach Ablauf der projektierten Zeit wird die

Parameter	Beschreibung
	Verschlüsselung neu initialisiert. Dabei werden Zertifikate und Schlüssel neu ausgetauscht. Zeitangabe in Sekunden. Default: *86400*
[maximale übertragene Datenmenge einer verschlüsselten Verbindung]	Datenmenge, für die eine verschlüsselte Verbindung benutzt wird, bevor diese erneuert wird. Nach Übertragung der projektierten Datenmenge wird die Verschlüsselung neu initialisiert. Dabei werden Zertifikate und Schlüssel neu ausgetauscht. Angabe in Bytes. Default: *1048576*
[Überprüfungsintervall der REVOCATION LIST]	Zeitintervall für die Prüfung, ob das aktuell verwendete Zertifikat in der **REVOCATION LIST** enthalten ist. Ist das Zertifikat in der **REVOCATION LIST** enthalten, ist die verschlüsselte Verbindung nicht mehr sicher. Die Verbindung wird abgebrochen. Zeitangaben in Sekunden. Default: *21600*

*** TLS CONTEXT ***

Die Konfiguration des Ablageortes der Zertifikate gelten für den Treiber und können nicht einzeln pro Verbindung projektiert werden.

Parameter	Beschreibung
[Ablageort der Zertifikate]	Absoluter Ablageort der Zertifikate. Dieses Basisverzeichnis muss die beiden Unterverzeichnisse **CA** und **PRIVATE** enthalten. ▸ **CA** für vertrauenswürdige Zertifikate und ▸ **PRIVATE** für eigene Zertifikate

BEISPIEL

1

*** LINK ***

29

1

Link1

0

192.168.0.230

2404

1

1

2

2

3

0.0.0.0

0

15000

10000

20000

12

8

0

0

0

0 [TLS aktiviert]

[Liste Verschlüsselungsverfahren]

[Zertifikat]

[Zertifikat Identifizierung]

0 [Zeit bis zur Erneuerung des Schlüssels] in Sekunden

0 [Datenmenge bis zur Erneuerung des Schlüssels]

0 [maximale Dauer einer verschlüsselten Verbindung]

0 [maximale übertragene Datenmenge einer verschlüsselten Verbindung]

0 [Überprüfungsintervall der REVOCATION LIST] in Sekunden

1

*** SECTOR ***

2

Sector1

1

1

*** TLS CONTEXT ***

1

C:\Certificates [Ablageort der Zertifikate]

12 Fehleranalyse

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

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

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

%ProgramData%\COPA-DATA\LOG.

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

Im Diagnosis Viewer kann man auch:

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

Hinweise:

1. Der Diagnosis Viewer zeigt alle Einträge in UTC (Koordinierter Weltzeit) an und nicht in der lokalen Zeit.
2. Der Diagnosis Viewer zeigt in seiner Standardeinstellung nicht alle Spalten einer LOG-Datei an. Um mehr Spalten anzuzeigen, aktivieren Sie die Eigenschaft **Add all columns with entry** im Kontextmenü der Spaltentitel.
3. Bei Verwendung von reinem **Error-Logging** befindet sich eine Problembeschreibung in der Spalte **Error text**. In anderen Diagnose-Ebenen befindet sich diese Beschreibung in der Spalte **General text**.
4. Viele Treiber zeichnen bei Kommunikationsprobleme auch Fehlernummern auf, die die SPS ihnen zuweist. Diese werden in **Error text** und/oder **Error code** und/oder **Driver error parameter(1** und **2)** angezeigt. Hinweise zur Bedeutung der Fehlercodes erhalten Sie in der Treiberdokumentation und der Protokoll/SPS-Beschreibung.
5. Stellen Sie am Ende Ihrer Tests den Diagnose-Level von **Debug** oder **Deep Debug** wieder zurück. Bei **Debug** und **Deep Debug** fallen beim Protokollieren sehr viele Daten an, die auf der Festplatte gespeichert werden und die Leistung Ihres Systems beeinflussen können. Diese werden auch nach dem Schließen des Diagnosis Viewers weiter aufgezeichnet.

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

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

12.2 Treiberüberwachung

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

Mögliche Ursachen für Auslösen des Watchdogs:

- Der Treiberprozess läuft nicht mehr.

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

- Betriebssystem ist mit höher priorisierten Prozessen ausgelastet.

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

WATCHDOG KONFIGURATION

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

Dafür werden entsprechende LOG-Einträge erstellt.

LOG-EINTRAG

Bei Auslösen des Watchdogs wird eine Fehlermeldung im LOG protokolliert:

Parameter	Beschreibung
Communication with driver:<drvExe>/<drvDesc>(id:<drvId>) timed out. No communication for <time> ms.	Keine Kommunikation mit Treiber innerhalb der angegeben Zeit. - *<time>*: Zeitangabe in Millisekunden - *<drvDesc>*: Treibername - *<drvExe>*: Treiber EXE-Name - *<drvId>*: Treiber-ID im zenon Projekt
Communication with %s timed out. Invalid-Bit will be set.	Die Kommunikation zum Treiber *%s* konnte bei 5 Versuchen nicht innerhalb von 60 Sekunden aufgebaut werden. Bei der Variable wird das *INVALID*-Bit gesetzt.
Communication with %s timed out. Timeout happened %d times	Die Kommunikation zum Treiber *%s* konnte *%d* Mal nicht innerhalb von 60 Sekunden aufgebaut werden.

12.3 Checkliste

Überprüfen Sie bei Fehlern Ihr System anhand der Treiberdokumentation auf korrekte Treiberkonfiguration, Adressierung der Variablen in zenon Projekten, Reporteinstellungen in der Steuerung etc.

Überprüfen Sie vor allem:

1. Ist das Gerät (IEC 61850 Server) für die Kommunikation mit dem Stromnetz verbunden und bereit für den Einsatz?
   a) Kann eine TCP-Verbindung zum Gerät aufgebaut werden (überprüfen Sie das via 'ping') über den definierten Port (überprüfen Sie das via Telnet)?
   a) Ist eine Kommunikation zwischen diesem Gerät und einem anderen IEC 61850 Client möglich?
   b) Befindet sich der Treiber im Hardwaremodus und nicht im Simulationsmodus?
2. Haben Sie die Fehler, die in der LOG-Datei (mehr im Handbuch Tools im Kapitel Diagnosis Viewer) der Treiber protokolliert werden, ausgewertet?
   a) Welche ERROR-Einträge sind aufgetreten?
   b) Haben Sie die Informationen in Spalten: **Error Text** und **General Text** überprüft?
   - In Spalte **General Text**, die Information "... failed due to error N" zeigt mit welcher Fehlernummer (N) - definiert in MMS Protokoll - der IED geantwortet hat, z.B.: 3 - "access-violation", 5 - "parameter-value-inconsistent", 11 - "value-out-of-range" etc.
   a) Wurde das Logging: für den richtigen Treiber, wenigstens für das Error-Level und für alle Treibermodule aktiviert?
   b) Haben Sie berücksichtigt, dass die Zeiten im **Diagnosis Viewer** immer UTC sind?
   c) Sind die Einträge im LOG mit Aufzeichnungen in **Wireshark** vergleichbar?
   - Der Treiber loggt ergänzend (nicht anstatt) zu Informationen die **Wireshark** mit Filter auf "mms" darstellt.
   a) Ist der **Diagnosis Viewer** so eingestellt, dass alle registrierten Einträge und alle Spalten mit dem Inhalt (ohne Filter) dargestellt werden?
3. Variablenadressierung (wenn die Variablen manuell oder per Offline-Import aus der SCL-Datei erstellt wurden):
   a) haben Sie versucht die Variable online zu importieren?
   b) Haben alle Variablen die richtigen **Netzadressen**?
   c) Sind die **Symbolische Adressen** (oder Namen, Kennungen) der Variablen gültig (SERVER!LD/LN/DO/DA[FC])?
   d) Haben die Variablen den richtigen **Treiberobjekttyp** (*PLC-marker*), **Datentyp** und **Wertebereich**?

4. Variablen ohne Wert in der Runtime, die aber beim Online-Import existieren:
   a) ein RCB das in Dataset diese Variablen liefen soll wurde aktiviert jedoch der 850-Server sendet kein Report - überprüfen Sie ob **TrgOps.general-interrogation** und entweder **TrgOps.data-change** oder **TrgOps.data-update** aktiviert sind.
   b) die Variablen sind in Dataset eines URCB (nicht BRCB) und der 850-Server sendet diese Unbufered Reports (**Wireshark**: mms.InformationReport), jedoch der Treiber kann sie nicht auswerten - weisen Sie alle URCB statisch in der Treiberkonfiguration zu und deaktivieren Sie für diese URCBs das **Optional Fields** - **Entry ID**. Jedoch bei BRCB muss das **Entry ID** aktiv sein.
5. Allgemein und Reporting:
   a) Wurde wenigstens eine Variable von je Logical Device im Projekt erstellt?
   - Aus Performance-Gründen überspringt der Treiber in der Runtime alle Logical Devices für die keine Variablen im Projekt existieren, auch die statisch konfigurierte RCBs können dabei ignoriert werden.
   a) Wurde wenigstens eine Variable vom Logical Device gepollt oder ist **Automatic Watchdog** aktiviert?
   - Wird bei aktivierten **Automatic Watchdog** kein MMS.Read periodisch gesendet, dann ist MMS blockiert (MaxOustandingCallings wurden erreicht) - der 850-Server hat mit TCP.ACK reagiert jedoch die MMS Request PDUs nicht beantwortet. Ein 850-Client darf dann keine weitere MMS Requests senden, muss warten. Die Situation trift zu wenn IED fehlerhaft ist (reagiert verzögert und sendet kein Abort by Reset).
     Der Treiber erzeugt einen Error-Eintrag in der LOG-Datei und die Verbindung wird abgebrochen.
   a) Wenn ein RCB, der nicht im *LLN0* liegt, nicht berücksichtigt wurde: wurde wenigstens eine Variable vom Logical Node des RCBs im Projekt erstellt?
   - Aus Performance-Gründen überspringt der Treiber in der Runtime alle Logical Nodes außer *LLN0* für die keine Variablen im Projekt existieren.
   a) Wurde in der Treiberkonfiguration die Eigenschaft **use Report-ID** aktiviert jedoch nicht gebraucht oder nicht mit SCL-Datei des IEDs abgeglichen?
   - Falls **use Report-ID** umgestellt wurde: wurde nachher die **statische Zuweisung** der RCBs erneut konfiguriert wie erforderlich?
   a) Wurden in der Treiberkonfiguration - in **Hostname** - die Schlüsselworte *$SCADA_SERVER1* oder *$SCADA_SERVER2* oder der Name des Runtime Rechners eingetragen und je mit anderen Instanzen der RCBs verknüpft?
   - Die Netzwerk Server und Standby Server dürfen nicht die gleiche Instanzen verwenden.
   - Auch im Projekten ohne Netzwerk muss **Hostname** eingetragen werden.
   a) Hat die Eigenschaft **max. auto used URCBs** den richtigen Wert?

- Wenn Sie bereits im Treiber die **statische Zuweisung** für RCBs konfiguriert haben - dann 0;
- wenn die **statische Zuweisung** noch nicht konfiguriert wurde - ein Wert der allen 850-Clients im System genügend RCB Instanzen übrig lässt.

a) Ist im Gerät das Reporting verfügbar?

- Ein Standard-konformer 850-Server kann, jedoch muss nicht, Reporting unterstützen. Z.B. seine ICD-Datei (Datei des Herstellers) kann unvollständige Konfiguration für RCBs beinhalten und es wird erwartet dass für fertige Systeme die vervollständigte CID-Dateien in IED hochgeladen werden.

a) Beinhalten die RCB, in DatSet Parameter, die Namen von existierenden Datesets?

- In zweifelsfall legen Sie die */DatSet[RP|BR] und */RptEna[RP|BR] Variablen an und überprüfen Sie deren Wert z.B. in **Variablen-Diagnose** Bild.

a) Ist die **TrgOp general interrogation** aktiv?

- Ohne GI ist es dem Treiber unmöglich die Initialwert der Variablen von IED anzufordern.

a) Ist das IED imstande auch kurze Verbindungsabbruche zu erkennen und setzt dann korrekt alle RCBs automatisch frei (*RptEna* auf false)?

6. Dynamic Data Set

a) ist das DataSet-Attribut im RCB am IED beschreibbar?

b) Haben Sie Einschränkungen des IEDs wie z.B. die maximale Anzahl der Elemente im Data Set oder die maximale Länge der Name etc. berücksichtigt?

13 Anhang A - Beschreibung der Functional Constraints (FCs):

FC	Semantics IEC 61850-7-2	Beschreibung
ST	Status information	steht für eine Statusinformation, e.g. Position eines Schalters. Der Wert kann gelesen, ersetzt, reported aber nicht geschrieben werden. In Datenmodell eines 850-Servers, in *ST* Datenattributen eines Datenobjekt muss *t* (Zeitstempel) und *q* (Quality) vorhanden sein. Diese müssen aber nicht als zenon Variablen erstellt werden. **Beispiel:** Variable für **/stVal[ST]* hat automatisch die Statusbits aus *q* und Zeitstempel aus *t*.

FC	Semantics IEC 61850-7-2	Beschreibung
MX	Measurands (analogue values)	steht für eine Messgröße, deren Wert gelesen, ersetzt, reported aber nicht geschrieben werden kann. In Datenmodell eines 850-Servers, in *MX* Datenattributen eines Datenobjekt muss *t* (Zeitstempel) und *q* (Quality) vorhanden sein. Diese müssen aber nicht als zenon Variablen erstellt werden. **Beispiel:** Variable für **/cVal.mag.f[MX]* hat automatisch die Statusbits aus *q* und Zeitstempel aus *t*.
CO	Control	steht für Control Information eines Befehles (entsprechend aktuellen Control Model des Datenobjektes). Der Wert kann ausschließlich geschrieben werden, nicht gelesen. **Beispiel:** ▸ Sollwert setzen auf eine Variable, die einen Datenattribut */Oper.ctlVal[CO] entspricht, wird im IEC850 Treiber automatisch auf ein Operate Befehl oder der Select Before Operate Ablauf umgesetzt. ▸ Sollwert setzen auf eine Variable, die einen Datenattribut */Oper.Test[CO] entspricht, wird im IEC850 Treiber für Test-Parameter des nächsten Befehles verwendet.
CF	Configuration	steht für Konfigurationsinformation, deren Wert gelesen, reported und geschrieben werden kann. **Beispiel:** Der Wert der Variable, die einen Datenattribut **/cltModel[CF]* entspricht, zeigt aktuelles Control Model des Datenobjektes.
SP	Setting (outside setting group)	steht für einen Parameter, dessen Wert gelesen, reported und geschrieben werden kann. **Beispiel:** ▸ Ein Setting Group Control Block (*SGCB*) beinhaltet *SP* Datenattribute, um die *SG/SE* Datenattribute zu steuern. ▸ Einige der Common Data Classes beinhalten das Datenattribut *setVal[SP]* um das Verhalten von 850-Server zu parametrieren.

FC	Semantics IEC 61850-7-2	Beschreibung
SG /SE	Setting Group (/editable)	steht für zusätzliche Datenattribute deren Werte, gesteuert über ein Setting Group Control Block (*SGCB*), als Parametern dienen. Ein *SG* Wert kann gelesen werden, der *SE* auch geschrieben. **Hinweis:** in Datenmodell eines 850-Servers können die *SG* und *SE* Datenattribute in beliebigen Datenobjekten vorkommen, um diese Datenobjekte zu parametrieren.
RP/ BR	Unbuffered / Buffered Report Control Block	steht für Datenattribute in Report Control Block: ▸ *RP* - Unbuffered ▸ *BR* - Buffered Der Wert kann gelesen werden, und - bedingt (61850-7-2) - geschrieben. **Beispiel:** Sollwert setzen *1* (*true*) auf eine Variable, die einen Datenattribut **/GI[RP|BR]* entspricht, ist manuelles Auslösen der Generalabfrage. **Hinweis:** die *RP* und *BR* Variablen können via **Online Import** angelegt werden.
LG	Log Control Block	steht für Datenattribute in Log Control Block. Der Wert kann gelesen werden, und - bedingt (61850-7-2) - geschrieben.
GO	GOOSE Control Block	steht für Datenattribute in GOOSE Control Block. Der Wert kann gelesen werden, und - bedingt (61850-7-2) - geschrieben. **Beispiel:** Sollwert setzen *0/1* (*false/true*) auf eine Variable, die einen Datenattribut **/GoEna[GO]* entspricht, ist manuelles Deaktivieren/Aktivieren eines GOOSE Publishers.
MS/ US	Multicast / Unicast Sampled Values Control Block	steht für Datenattribute in Sampled Values Control Block: ▸ *MS* - Multicast ▸ *US* - Unicast Der Wert kann gelesen werden, und - bedingt (61850-7-2) - geschrieben.
SV	Substitution	steht für zusätzliche Datenattribute deren Werte als Ersatzwerte des Prozesses dienen können. Der Wert kann

FC	Semantics IEC 61850-7-2	Beschreibung
		gelesen und geschrieben werden. **Hinweis:** in Datenmodell eines 850-Servers können die *SV* Datenattribute in beliebigen Datenobjekten vorkommen, um, falls nötig, die betroffene Prozessdaten in diesen Datenobjekt zu ersetzen.
DC	Description	steht für Beschreibungen, üblicherweise die *STRINGs*. Der Wert kann gelesen werden; in seltenen Fällen auch geschrieben - wenn *DA* nicht als '*RO*' (read-only) in *SCL* konfiguriert wurde.
EX	Extended definition	steht für Referenzen im den Datenmodel. Der Wert kann gelesen werden.
SR	Service Response	steht für Service Tracking Informationen, deren Wert gelesen, reported aber nicht geschrieben werden kann. **Hinweis:** • in Datenmodell eines 850-Servers sind *SR* Daten ausschließlich in Logical Node *LTRK* möglich. • In IEC850 Treiber gibt es zusätzliche Variablen von **Treiber Objekttyp** *Service Tracking* mit FC=*SR*. Diese zusätzlichen *SR* Variablen können in beliebigen Datenobjekten und Control Blocks erstellt werden, um Service Tracking über diese auszuwerten.
OR	Operate received	steht für Informationen relevant während Befehlsausführung; spezifisch für das Datenobjekt. Der Wert kann gelesen werden.
BL	Blocking	steht für Datenattribute für *operator-block* Verriegelungen im IED. Der Wert kann gelesen und geschrieben werden.

14 Anhang B - Abkürzungen für data object/data attribute

Info

Anhang B - Datenobjekt/Datenattribut liegt nur in englischer Sprache vor.

Term	Beschreibung
f	floating point (analog value)
i	integer (analog value)
d	Visible String(255)
q	Quality
t	Timestamp
A	Current
Acs	Access
ACSI	Abstract Communication Service Interface
Acu	Acoustic
Age	Ageing
Alm	Alarm
Amp	Current non phase related
To	Analogue
Ang	Angle
Auth	Authorization
Auto	Automatic
Aux	Auxiliary
Av	Average
B	Bushing
Bat	Battery
Beh	Behaviour
Bin	Binary

Term	Beschreibung
Blk	Block, blocked
Bnd	Band
Bo	Bottom
Cancel	Cancel
Cap	Capability
Capac	Capacitance
Car	Carrier
CB	Circuit Breaker
CDC	Common Data Class
CE	Cooling Equipment
Cf	Crest factor
Cfg	Konfiguration
CG	Core Ground
Ch	Channel
Cha	Charger
Chg	Change
Chk	Check
Chr	Characteristic
Cir	Circulating
Clc	Calculate
Clk	Clock, clockwise
Cls	Close
Cnt	Counter
Col	Coil
Cor	Correction

Term	Beschreibung
Crd	Coordination
Crv	Curve
CT	Current Transducer
Ctl	Control
Ctr	Center
Cyc	Cycle
Dea	Dead
Den	Density
Det	Detected
DExt	De-excitation
Diag	Diagnostics
Dif	Differential, difference
Dir	Richtung
Dis	Distance
Dl	Delay
Dlt	Delete
Dmd	Demand
Dn	Down
DPCSO	Double point controllable status output
DQ0	Direct, Quadrature, and zero axis quantities
Drag	Drag hand
Drv	Drive
DS	Device State
Dsch	Discharge

Term	Beschreibung
Dur	Duration
EC	Earth Coil
EE	External Equipment
EF	Earth Fault
Ena	Enabled
Eq	Equalization, Equal
Ev	Evaluation
Ex	External
Exc	Exceeded
Excl	Exclusion
Ext	Excitation
FA	Fault Arc
Fact	Factor
Fan	Fan
Flt	Fault
Flw	Flow
FPF	Forward Power Flow
Fu	Fuse
Fwd	Forward
Gen	General
Gn	Generator
Gnd	Ground
Gr	Group

Term	Beschreibung
Grd	Guard
Gri	Grid
H	Harmonics (phase related)
H2	Hydrogen
H2O	Water
Ha	Harmonics (non phase related)
Hi	High, highest
HP	Hot point
Hz	Frequency
IEEE	Institute of Electrical and Electronic Engineers
Imb	Imbalance
Imp	Impedance non phase related
In	Input
Ina	Inactivity
Incr	Increment
Ind	Indication
Inh	Inhibit
Ins	Insulation
Int	Integer
ISCSO	Integer status controllable status output

Term	Beschreibung
Km	Kilometer
L	Lower
LD	Logical Device
LDC	Line Drop Compensation
LDCR	Line Drop Compensation Resistance
LDCX	Line Drop Compensation Reactance
LDCZ	Line Drop Compensation Impedance
LED	Light Emitting Diode
Len	Length
Lev	Level
Lg	Lag
Lim	Limit
Lin	Line
Liv	Live
Lo	Low
LO	Lockout
Loc	Local
Lod	Load, loading
Lok	Locked
Los	Loss
Lst	List
LTC	Load Tap Changer

Term	Beschreibung
M	minutes
M/O	Data Object is Mandatory or Optional
Max	Maximum
Mem	Memory
Min	Minimum
Mod	Mode
Mot	Motor
Ms	Milliseconds
Mst	Moisture
MT	Main Tank
N	Neutral
Nam	Name
Net	Net sum
Ng	Negative
Nom	Nominal, Normalizing
Num	Number
Ofs	Offset
Op	Operate, Operating
Oper	Operate*
Opn	Open
Out	Output
Ov	Over, Override, Overflow

Term	Beschreibung
Pa	Partial
Par	Parallel
Pct	Percent
Per	Periodic
PF	Power Factor
Ph	Phase
Phy	Physical
Pls	Pulse
Plt	Plate
Pmp	Pump
Po	Polar
Pol	Polarizing
Pos	Position
POW	Point on wave switching
PP	Phase to phase
PPV	Phase to phase voltage
Pres	Pressure
Prg	Progress, in progress
Pri	Primary
Pro	Protection
Ps	Positive
Pst	Post
Pwr	Power
Qty	Quantity

Term	Beschreibung
R	Raise
R0	Zero sequence resistance
R1	Positive sequence resistance
Rat	Winding ratio
Rcd	Record, recording
Rch	Reach
Rcl	Reclaim
Re	Retry
React	Reactance; Reactive
Rec	Reclose
Red	Reduction
Rel	Release
Rem	Remote
Res	Residual
Ris	Resistance
Rl	Relation, relative
Rms	Root mean square
Rot	Rotation, Rotor
Rs	Reset, Resetable
Rsl	Result
Rst	Restraint
Rsv	Reserve
Rte	Rate
Rtg	Rating
Rv	Reverse
Rx	Receive, received

Term	Beschreibung
S1	Step one
S2	Step two
SBO	Select before operate*
SBOw	Select with value*
Sch	Scheme
SCO	Supply change over
SCSM	Specific Communication Service Mapping
Sec	Security
Seq	Sequence
Set	Setting
Sh	Shunt
Spd	Speed
SPI	Single Pole
SPCSO	Single point controllable status output
Src	Source
St	Status
Stat	Statistics
Stop	Stop
Std	Standard
Str	Start
Sup	Supply
Svc	Service
Sw	Switch
Swg	Swing

Term	Beschreibung
Syn	Synchronisation
Tap	Tap
Td	Total distortion
Tdf	Transformer derating factor
Test	Test
Thd	Total Harmonic Distortion
Thm	Thermal
TiF	Telephone influence factor
Tm	Time
	• Tmh = Time in h
	• Tmm = Time in min
	• Tms = Time in s
	• Tmms = Time in ms
Tmp	Temperature (°C)
To	Top
Tot	Total
TP	Three pole
Tr	Trip
Trg	Trigger
Ts	Total signed
Tu	Total unsigned
Tx	Transmit, transmitted
Typ	Type
Un	Under

Term	Beschreibung
V	Voltage
VA	Volt Amperes
Vac	Vacuum
Val	Value
VAr	Volt Amperes Reactive
Vlv	Valve
Vol	Voltage non phase related
VT	Voltage Transducer
W	Active Power
Wac	Watchdog
Watt	Active Power non phase related
Wei	Weak End Infeed
Wh	Watt hours
Wid	Width
Win	Window
Wrm	Warm
X0	Zero sequence reactance
X1	Positive sequence reactance
Z	Impedance
Z0	Zero sequence impedance
Z1	Positive sequence impedance

Term	Beschreibung
Zer	Zero
Zn	Zone
Zro	Zero sequence method

* SBO, SBOw, Oper, und Cancel sind in IEC 61850-8-1 definiert:

Data attribute name	Semantics
SBO	Select – returns ACSI name of control
SBOw	SelectWithValue – receives service parameters
Oper	Operate – receives service parameters and control values
Cancel	Cancel – receives service parameters and control values

15 Anhang C - Conformance statement

> **Info**
>
> Das Confirmance Statement des **IEC850 Treibers** zu dem Protokoll IEC61850 liegt nur in englischer Sprache vor.

15.1 Protocol implementation conformance statement (PICS)

This document specifies the PICS of the IEC 61850 interface in the client system: "**zenon Supervisor**/*Energy Edition*", Version *8.00*, further referred to as "Client". The zenon Editor and Runtime are further referred to as "HMI".

A.1 - BASIC ACSI CONFORMANCE STATEMENT

Client-Server roles		Client / subscriber	Server / publisher	Value / comments
B11	Server side (of TWO-PARTY-APPLICATION-ASSOCIATION)	--	N	for Server see zenon Logic
B12	Client side of (TWO-PARTY-APPLICATION-ASSOCIATION)	Y	--	
SCSMs supported		**Client / subscriber**	**Server / publisher**	**Value / comments**
B21	SCSM: IEC 61850-8-1 used	Y		
B22	SCSM: IEC 61850-9-1 used	N		deprecated
B23	SCSM: IEC 61850-9-2 used	N		
B24	SCSM: other	N		
Generic substation event model (GSE)		**Client / subscriber**	**Server / publisher**	**Value / comments**
B31	Publisher side	--	Y	see zenon Logic
B32	Subscriber side	Y	--	see zenon Logic
Transmission of sampled value model (SVC)		**Client / subscriber**	**Server / publisher**	**Value / comments**
B41	Publisher side	-	N	
B42	Subscriber side	N	-	for 90-5 see zenon Logic

- = not applicable

Y = supported

N or **empty** = not supported

A.2 - ACSI MODELS CONFORMANCE STATEMENT

IF SERVER SIDE (B11) AND/OR CLIENT SIDE (B12) SUPPORTED

		Client / subscriber	Value / comments
M1	Logical device	Y	
M2	Logical node	Y	
M3	Data	Y	
M4	Data set	Y	
M5	Substitution	Y	
M6	Setting group control	Y	
Reporting		**Client / subscriber**	**Value / comments**
M7	Buffered report control	Y	
M7-1	sequence-number	Y	
M7-2	report-time-stamp	Y	
M7-3	reason-for-inclusion	Y	
M7-4	data-set-name	Y	
M7-5	data-reference	Y	
M7-6	buffer-overflow	Y	
M7-7	entryID	Y	
M7-8	BufIm	Y	
M7-9	IntgPd	Y	
M7-10	GI	Y	
M8	Unbuffered report control	Y	
M8-1	sequence-number	Y	
M8-2	report-time-stamp	Y	
M8-3	reason-for-inclusion	Y	

		Client / subscriber	Value / comments
M8-4	data-set-name	Y	
M8-5	data-reference	Y	
M8-6	BufIm	Y	
M8-7	IntgPd	Y	
M8-8	GI	Y	
Logging		**Client / subscriber**	**Value / comments**
M9	Log control	N	
M9-1	IntgPd	N	
M10	Log	N	
general		**Client / subscriber**	**Value / comments**
M11	Control	Y	
M17	File Transfer	Y	
M18	Application association	Y	
M19	GOOSE Control Block	Y	
M20	Sampled Value Control Block	Y	

IF GSE (B31/B23) IS SUPPORTED

		publisher & subscriber	Value / comments
M12	GOOSE	Y	
M13	GSSE	N	

IF SVC (B41/B42) IS SUPPORTED

		Client / subscriber	Value / comments
M14	Multicast SVC	N	

		Client / subscriber	Value / comments
M15	Unicast SVC	N	

FOR ALL IEDS

		Client / subscriber	Value / comments
M16	Time	Y	Time of Operating System must be available with required accuracy

Y = service is supported

N or **empty** = service is not supported

A.3 - ACSI SERVICE CONFORMANCE STATEMENT

SERVER

		AA: TP/MC	Client (C)	Comments
S1	GetServerDirectory (LOGICAL-DEVICE)	TP	Y	driver start
Application association		**AA: TP/MC**	**Client (C)**	**Comments**
S2	Associate		Y	driver start
S3	Abort		Y	stop / exit
S4	Release		N	
Logical device		**AA: TP/MC**	**Client (C)**	**Comments**
S5	GetLogicalDeviceDirectory	TP	Editor: Y Runtime: N	only for **Online Import** in Editor
Logical node		**AA: TP/MC**	**Client (C)**	**Comments**
S6	GetLogicalNodeDirectory	TP	Y	only for data sets
S7	GetAllDataValues	TP	N	

DATA

		AA: TP/MC	Client (C)	Comments
S8	GetDataValues	TP	Y	variable(s) value ▸ polling: FCD ▸ watchdog, scan, SBO: FCDA
S9	SetDataValues	TP	Y	write set value
S10	GetDataDirectory	TP	Y	driver start
S11	GetDataDefinition	TP	N	
Data set		**AA: TP/MC**	**Client (C)**	**Comments**
S12	GetDataSetValues	TP	N	
S13	SetDataSetValues	TP	N	
S14	CreateDataSet	TP	Y	dynamic data set in RCB
S15	DeleteDataSet	TP	Y	dynamic data set in RCB
S16	GetDataSetDirectory	TP	Y	driver start
Substitution		**AA: TP/MC**	**Client (C)**	**Comments**
S17	SetDataValues	TP	Y	write set value
Setting group control		**AA: TP/MC**	**Client (C)**	**Comments**
S18	SelectActiveSG	TP	Y	write set value */LLN0/SGCB/ActSG[SP]
S19	SelectEditSG	TP	Y	write set value */LLN0/SGCB/EditSG[SP]
S20	SetSGValues	TP	Y	write set value *[SG], *[SE]
S21	ConfirmEditSGValues	TP	Y	write set value */LLN0/SGCB/CnfEdit[SP]
S22	GetSGValues	TP	Y	variable(s) value

		AA: TP/MC	Client (C)	Comments
				*[SG], *[SE]
S23	GetSGCBValues	TP	Y	variable(s) value */LLN0/SGCB/*[SP]

REPORTING

Buffered report control block (BRCB)		AA: TP/MC	Client (C)	Comments
S24	Report	TP	Y	variable(s) value
S24-1	*data-change (dchg)*		Y	
S24-2	*quality-change (qchg)*		Y	
S24-3	*data-update (dupd)*		Y	
S25	GetBRCBValues	TP	Y	variable(s) value *[BR]
S26	SetBRCBValues	TP	Y	write set value *[BR]
Unbuffered report control block (URCB)		**AA: TP/MC**	**Client (C)**	**Comments**
S27	Report	TP	Y	variable(s) value
S27-1	*data-change (dchg)*		Y	
S27-2	*quality-change (qchg)*		Y	
S27-3	*data-update (dupd)*		Y	
S28	GetURCBValues	TP	Y	variable(s) value *[RP]
S29	SetURBCValues	TP	Y	write set value *[RP]

LOGGING

Log control block		AA: TP/MC	Client (C)	Comments
S30	GetLCBValues	TP	Y	variable(s) value *[LG]
S31	SetLCBValues	TP	Y	write set value *[LG]
Log		**AA: TP/MC**	**Client (C)**	**Comments**
S32	QueryLogByTime	TP	N	
S33	QueryLogAfter	TP	N	
S34	GetLogStatusValues	TP	Y	variable(s) value

GENERIC SUBSTATION EVENT MODEL (GSE)

GOOSE-CONTROL-BLOCK		AA: TP/MC	publisher, subscriber	Comments
S35	SendGOOSEMessage	MC	Y	see zenon Logic
S36	GetGoReference	TP	N	
S37	GetGOOSEElementNumber	TP	N	
S38	GetGoCBValues	TP	client Y	variable(s) value *[GO]
S39	SetGoCBValues	TP	client Y	write set value *[GO]
GSSE-CONTROL-BLOCK		**AA: TP/MC**	**Client (C)**	**Comments**
S40	SendGSSEMessage	MC	N	deprecated
S41	GetGsReference	TP	N	deprecated
S42	GetGSSEDataOffset	TP	N	deprecated
S43	GetGsCBValues	TP	Y	deprecated

GOOSE-CONTROL-BLOCK		AA: TP/MC	publisher, subscriber	Comments
S44	SetGsCBValues	TP	Y	deprecated

TRANSMISSION OF SAMPLED VALUE MODEL (SVC)

Multicast SVC		AA: TP/MC	Client (C)	Comments
S45	SendMSVMessage	MC	N	
S46	GetMSVCBValues	TP	Y	variable(s) value *[MS]
S47	SetMSVCBValues	TP	Y	write set value *[MS]
Unicast SVC		**AA: TP/MC**	**Client (C)**	**Comments**
S48	SendUSVMessage	TP	N	
S49	GetUSVCBValues	TP	Y	variable(s) value *[SV]
S50	SetUSVCBValues	TP	Y	write set value *[SV]

CONTROL

		AA: TP/MC	Client (C)	Comments
S51	Select	TP	Y	automatically before Operate
S52	SelectWithValue	TP	Y	automatically before Operate
S53	Cancel	TP	Y	cancel of first-stage of an **Action** if DO selected
S54	Operate	TP	Y	write set value */Oper.ctlVal[CO]

		AA: TP/MC	Client (C)	Comments
S55	Command-Termination	TP	Y	automatically
S56	TimeActivated-Operate	TP	N	

FILE TRANSFER

		AA: TP/MC	Client (C)	Comments
S57	GetFile	TP	Y	FT!Command *GET*
S58	SetFile	TP	N	
S59	DeleteFile	TP	Y	FT!Command *DEL*
S60	GetServerDirectory (FILE-SYSTEM)	TP	Y	FT!Command *DIR*

TIME

		Client (C)	Comments
T1	Time resolution of internal clock	Operating System (T1)	Microsoft Windows PC resolution, default 1ms
T2	Time accuracy of internal clock	Operating System (T1)	T0 (10ms), T1 (1ms), T2 (100µs), T3 (25µs), T4 (4µs), T5 (1µs)
T3	Supported TimeStamp resolution	T1 (1ms)	or decimal places of LREAL when t-attribute created as variable (T3)

15.2 Model implementation conformance statement (MICS)

This document specifies the MICS of the IEC 61850 interface in the client system: "**zenon Supervisor**/*Energy Edition*", Version *8.00*, further referred to as "Client". The zenon Editor and Runtime are further referred to as "HMI".

- All logical nodes, common data classes and data attribute types - as defined in IEC 61850-7-3 and IEC 61850-7-4 - are supported.
- Data types INT64 and INT128 are fully supported on MMS, but their values in the Runtime only in range of lowest 52 bits as mapped to **LINT** or mapped to **LREAL** with eventual loss of precision.
- Data type ARRAY OF - only single-level arrays on data attribute level are supported; items of this data type are read-only. **Hint:** deactivate variable property **Set value**.
- Object References in received Service Tracking data are not supported with '@' wildcard instead the IED Name.

16 Anhang D - PIXIT

> **Info**
>
> Das PIXIT Dokument zum Zertifizierungstest des IEC850 Treibers "**zenon Supervisor**/*Energy Edition*", Version *8.00*, liegt nur in englischer Sprache vor.
>
> PIXIT basiert auf einer von der UCA International Users Group gelieferten Vorlage.

INTRODUCTION

This document specifies the protocol implementation extra information for testing (PIXIT) of the IEC 61850 interface in the client system: "**zenon Supervisor**/*Energy Edition*", Version *8.00*, further referred to as “Client”. The zenon Editor and Runtime are further referred to as "HMI".

Together with the PICS and the MICS the PIXIT forms the basis for a conformance test according to IEC 61850-10.

The following chapters specify the PIXIT for each applicable ACSI service model as structured in IEC 61850-10 and the “Conformance Test Procedures for Client System with IEC 61850-8-1 Edition 2 interface” **by UCA International Users Group**.

16.1 PIXIT for Configuration

ID	Description	Value / Clarification
Cf1	Describe how the client handles nameplate configuration revision mismatches	**The client reads out data model at each associate and therefore does not need to check configuration revision.**

ID	Description	Value / Clarification
Cf2	Describe how the client handles report control block configuration revision mismatches	**RCB configuration is read out at each associate.**
	<additional items>	**The client does not use items of pre-defined data model if online read of data model fails. Preconfigured items are used according current online definition or omited if not available online.** **The SCL-file is used in Editor for pre-configuration of HMI projects, e.g. in driver configuration Wizard and in offline import of variables. Please refer client documentation for more details.**

16.2 PIXIT for Association model

ID	Description	Value / Clarification
As1	Garanteed number of servers that can set-up an association simultaneously (one association per server)	**per design: 65535** **tested: 1000**
As2	Lost connection detection time range (default range of TCP_KEEPALIVE is 1 – 20 seconds)	**current TCP timeout of Operating System and Ethernet equipment or, while no new MMS requests - TCP KeepAliveTime is 20s with 10 retries.**
As3	Describe the behavior when association fails	**Retry each 30 seconds** - **if was associated longer as 30s: the first retry is immediately;** - **else: 30s since previous TCP socket (re)open**
As4	Is authentication supported?	**Y**
As5	What is the maximum and minimum MMS PDU size?	Max MMS PDU size **65535** Min MMS PDU size **depends on size of largest DO in server's data model**

ID	Description	Value / Clarification
As6	What is the typical startup time after a power supply interrupt?	**The client works only together with the zenon Runtime, these are Windows applications.** **zenon Runtime + client may work on redundant PCs then by power loss or Hardware problem on main PC the secundary (standby) system takes (seamless) over.** 1. **Windows does not guarantee the files will be not corrupted after power failure. We recommend to use Redundant systems and/or an UPS to prevent loss of data stored on HDD.** 2. **The start-up time for the client depends of HMI project design and PC performance. In typical HMI projects client opens TCP/IP session to a server after few (1-4) seconds since application start.**
As7	How does the client disconnect from the server?	**service: Abort**

16.3 PIXIT for Server model

ID	Description	Value / Clarification
Sr1	Maximum object identification length?	**LD: 128 chars/octets** **FC: 2 chars/octets** **LN: 16 chars/octets** **DO: 128 chars/octets** **DA: 128 chars/octets** **in sum max 256 chars/octets (Symbolic Address)**
Sr2	Does client support autodescription?	**Y** **During association client always reads current data model of server via online**

ID	Description	Value / Clarification
		services: GetServerDirectory and GetDataDirectory. During this process client skips not present LD and reads LN of created HMI variables; then skips not present DO and DA. **After association client does not use items not present on server. The HMI variables, if created but mirroring non-existing items - have I-Bit*.**
Sr3	Describe how to view data values	**as value of HMI variable**
Sr4	What analogue value (MX) quality bits are used in the client?	▸ Y Good, ▸ Y Invalid, ▸ Y Reserved, ▸ Y Questionable ▸ Y Overflow ▸ Y OutofRange ▸ Y BadReference ▸ Y Oscillatory ▸ Y Failure ▸ Y OldData ▸ Y Inconsistent ▸ Y Inaccurate ▸ Y Process ▸ Y Substituted ▸ Y Test ▸ Y OperatorBlocked
Sr5	Which status value (ST) quality bits are used in the client?	▸ Y Good, ▸ Y Invalid, ▸ Y Reserved, ▸ Y Questionable ▸ Y BadReference ▸ Y Oscillatory

ID	Description	Value / Clarification
		▸ Y Failure ▸ Y OldData ▸ Y Inconsistent ▸ Y Inaccurate ▸ Y Process ▸ Y Substituted ▸ Y Test ▸ Y OperatorBlocked
Sr6	Describe how to view/display quality values	1. **the status bits of */stVal variable in HMI, if created (only selected bits; please refer client documentation for more details)** 2. **the value of */q variable in HMI, if created**
Sr7	Describe how to force a SetDataValues request	**Set value to the variable created in HMI, e.g. via screen element or write set value function.**
Sr8	Describe how to force a GetDataValues request	**no need to force - client does automatically:** ▸ **client does GetDataValues periodically for all not reported FC$DO when at least one DA is mirrored to currently advised HMI variable.** ▸ **when no variables are currently advised, client uses GetDataValues(LLN0STMod$stVal) as automatic watchdog for connection loss detection.**
Sr9	Describe how to force a GetAllDataValues request	**service not used**
Sr10	Does the client support writing blkEna values?	Y
Sr11	Describe how the client behaves in case of:	**see also Sr2**

ID	Description	Value / Clarification
	▸ GetDataDefinition response-	▸ **service not used**
	▸ GetDataDefinition response+ with more or less attributes as expected	▸ **service not used**
	▸ GetLogicalDeviceDirectory response-	▸ **service not used**
	▸ GetAllDataValues response-	▸ **service not used**
	▸ GetAllDataValues response+ with more or less attributes as expected	▸ **service not used**
	▸ GetDataValues response-	▸ **log error entry; value of HMI variable will be not updated - have previously received value or null**
	▸ GetDataValues response+ with more or less attributes as expected	**to request process data the client uses GetDataValues on FC$DO, not DA level (see Sr8).** ▸ **HMI variables mirroring items not present in server's data model have I-Bit* (see Sr2) and are not requested** ▸ **items present in data model are requested but if not included in GetDataValue response then adequate HMI variables are not updated** ▸ **additional items are ignored.**
	▸ SetDataValues response-	▸ **log error entry**
Sr12	Which time quality attributes from the server are used in the client	▸ N Leap Second Known, ▸ Y ClockFailure ▸ Y Clock not synchronized ▸ Y Accuracy
Sr13	Describe how to view time quality attributes	**client maps time quality bits received in t-attribute to status bits of HMI variables responding the same FC$DO as following:** ▸ **ClockNotSynchronised - status bit T_UNSYNC**

ID	Description	Value / Clarification
		▸ **ClockFailure - status bit T_INVAL and HMI variable values have local timestamp** ▸ **Accuracy < 11111b - client truncates fraction of seconds of timestamp of HMI variable (option)**
	***I-Bit is status bit INVALID of HMI variables mirroring DO/DA. Please refer HMI documentation for more details.**	

16.4 PIXIT for Data set model

ID	Description	Value / Clarification
Ds1	Describe how to force a GetDataSetValues request	**service not used**
Ds2	Describe how to force a SetDataSetValues request	**service not used**
Ds3	Describe how to force a DeleteDataSet request	**client does it automatically when in client configuration user changes name and members for Dynamic Data Sets.**
Ds4	Describe how the client handles following dataset mismatches between the SCL and the data sets exposed via MMS: ▸ new dataset element ▸ missing dataset element ▸ Reordered dataset members in a dataset of a different data type ▸ Reordered dataset members in a dataset of the same data type	**No mismatch possible - client uses autodescription (see Sr2)**
Ds5	Describe how the client behaves in case of: ▸ GetLogicalNodeDirectory(DATA-SET) response- ▸ GetDataSetDirectory response- ▸ GetDataSetValues response-	**LOG entry + data will be polled (no reporting)**

ID	Description	Value / Clarification
	▸ SetDataSetValues response-	
Ds6	Maximum name length: ▸ for dataset ▸ for dataset member, including LD and FC	▸ **MMS: 128/16$128, HMI: 256** ▸ **MMS: 128/128, HMI: 256**
Ds11	Describe how to force a CreateDataSet request: 1. persistent 2. non-persisten	1. **client creates automatically new dataset before it enables the RCB where client is configured to use Dynamic Data Set; and when this dataset does not exist in IED yet.** 2. **not used**
Ds12	Describe how to force a DeleteDataSet request: 1. persistent 2. non-persisten	1. **before RCB enabling, client deletes automatically previously created datasets (having name with client's prefix) when this dataset is not to be used anymore.** 2. **not used**
Ds13	Describe how the client behaves in case of: ▸ CreateDataSet response- ▸ DeleteDataSet response-	▸ **log entry, RCB will be not enabled and data will be polled** ▸ **client retries to delete the data set by each next association**

16.5 PIXIT for Substitution model

ID	Description	Value / Clarification
Sub1	Describe how to substitute a value	**following steps:** 1. **create HMI variables for attributes with FC=SV (*/subEna, subQ, subID and e.g. subVal)** 2. **set values in HMI variables (*/subVal, subQ, subID) to values the IED then shall use while**

ID	Description	Value / Clarification
		substituting 3. **set value TRUE to HMI variable */subEna.** **Please refer HMI documentation for more details about setting variable values**
	<additional items>	**Hint: when IED does not support substitution there is an alternative method in the HMI to substitute HMI variable values: a HMI variable can be switched to "alternative value" and than the HMI - and not IED - substitutes the value.** **Please refer description of status bit ALT_VAL in the HMI documentation for more details.**

16.6 PIXIT for Setting group control model

ID	Description	Value / Clarification
Sg1	How can the client be forced to send a GetLogicalNodeDirectory(SGCB) request	**service not used** **Note: to (re)read data model via GetDataDirectory force reconnect, e.g. using HMI "driver commands" to stop and start client**
Sg2	Describe how to change the active setting group	**set value of adequate variable in HMI project (ActSG or EditSG)**
Sg3	Describe how to get the actual setting group values	**create variables in the HMI project for adequate attributes and advise these variables**
Sg4	Describe how to edit setting group values	**set value of adequate variables in HMI project**
Sg5	Describe how the client behaves in case of: - GetSGCBValues response- - SelectEditSG response-	**client uses autodescription and then requests only attributes present in server's data model. By write failure client logs error; by negative responses HMI**

ID	Description	Value / Clarification
	▸ SetEditSGValue response- ▸ SelectActiveSG response- ▸ ConfirmEditSGValues response-	**variables are not updated.**
Sg6	Does the client read the optional ResvTms value?	**Yes, only if attribute is present and adequate variable is created in the HMI project**
	<additional items>	

16.7 PIXIT for Reporting model

ID	Description	Value / Clarification
Rp1	Does the client search for RCB in all logical nodes? If not, specify the logical nodes	**Y - All logical nodes** ▸ **LLN0 automatically by sufficient PDU size** ▸ **any LN when RCB name is defined in the client configuration and at least one variable in this LN with FC=BR/RP exists in HMI** ▸ **in addition: BRCBs in all LDevices/LN - if BRCB.RptID=zenon_computername_***
Rp2	Which dynamic RCB attributes are/can be configured by the client	**by client configuration:** ▸ **Y RptID** ▸ **Y DataSet** ▸ **Y Optional fields** ▸ **Y Trigger conditions** ▸ **Y Buffer time** ▸ **Y Integrity period**
Rp3	Does the client support IED's with indexed and non-indexed report control blocks (RCB)	**Y Buffered RCB indexed** **Y Buffered RCB not indexed** **Y Unbuffered RCB indexed** **Y Unbuffered RCB not indexed**

ID	Description	Value / Clarification
Rp4	The supported trigger conditions are	**Y integrity** **Y data change** **Y quality change** **Y data update** **Y general interrogation**
Rp5	The minimum required optional fields are	**N sequence-number** **N report-time-stamp** **N reason-for-inclusion** **Y* data-set-name (*if RptID is not unique)** **N data-reference** **N buffer-overflow** **Y** entryID (**BRCBs only)** **N conf-rev** **client can process reports with any additional optional field, not only the minimum; the optional fields not required may be ignored.**
Rp6	Does the client support segmented reports	**Y**
Rp7	Does the client support pre-assigned RCB	**Y** **for more details see manual of IEC850 driver configuration wizard**
Rp8	Does the client support reported data set containing structured data objects or data attributes?	**Y reporting of data objects** **Y* reporting of data attributes** ***not recommended as this kind of dataset does not assure that each trigged report will contain current q and t attribute values.**
Rp9	Describe how the client does respond when an URCB is reserved by another client	**log entry + trying to use free one but only if configured to use automatic (not statical) enabling of URCBs.** **for statically assigned URCB see Rp10**
Rp10	Describe how the client does respond when a BRCB is reserved by another client	**log entry and 'connection state' indicator about failure; data will be polled** **optionally, per default each 7s, the client tries periodically to enable the RCB.**

ID	Description	Value / Clarification
Rp11	Describe how the client does respond on a SetBRCBValues(EntryID) respond-	**error log entry; as next client tries to enable BRCB anyway and sends GI; and process buffered data if IED reports whole contents of buffer**
Rp12	Describe how the client does respond when a report has an unknown: dataset, RptId, unexpexted number of dataset entries, and/or unexpexted data type format entries	**mismatches are not possible - client uses autodescription - see Sr2**
Rp13	Describe how the client detect reporting configuration changes (mismatches). Does it check the "configuration revision" attributes and/or does it check the dataset members? Is the dataset update done online or offline?	▸ **N* Check ConfRev** ▸ **Y Check dataset members** ▸ **online update at each associate** ***optionally, an unexpected change of ConfRev value, if created adequate HMI variable, may produce e.g. an user alarm**
Rp14	Describe how to force the client to change the RCB buffertime	**BufTm is a setting in the client configuration** **see also <addititonal items>**
Rp15	Does client set server TrgOps.GI prior to first issuance of GI command?	**Yes, if use of client's TrgOps is configured in the client configuration**
Rp16	Describe how to force the client to send the GI request	▸ **client does it automatically by enabling of RCB and by reconnect.** ▸ **GI on user demand - see <addititonal items>**
Rp17	Describe how to force the client to enable a RCB	**settings in the client configuration**
Rp18	Describe how the client does respond when a report control block is renamed or deleted ▸ Does it prevent reading the deleted RCB ▸ If it reads the missing RCB, how does it handle the GetURCBValues or GetBRCBValues response-	▸ **no reading of deleted RCB** **during autodescription (see Sr2) client checks if all in client (statically) configured RCBs are existing in the IED's data model; if not: log entry and 'connection state' indicator about failure; data will be polled**
Rp19	Describe how the client behaves in case of:	▸ **if SetURCBValues(RptEna) fails -> see Rp9 and Rp10**

ID	Description	Value / Clarification
	▸ SetURCBValues response- ▸ Unsupported optional fields ▸ Unsupported trigger condition(s)	▸ **if SetURCBValues(DataSet) fails -> error log entry; even when RCB is enabled data will be polled** ▸ **if SetURCBValues fails for other attributes -> client uses enabled RCB (like it is).** ▸ **client supports all OptFlds and TrgOps**
Rp20	Describe how the client behaves in case of: ▸ Buffer overflow	**LOG entry** **Note: by reconnect client acquires buffered data (EntryID) automatically and sends also GI. Buffer purge when EntryID not stored or by an option.**
Rp21	Describe how to force the client to send SetBRCBValues request for ▸ EntryID ▸ PurgeBuf	▸ **client sends by reconnect the last received EntryID auromaticaly** ▸ **client sends PurgeBuf automatically, but only by HMI start; this may be deactivated.** ▸ **PurgeBuf on user demand - see <addititonal items>**
Rp22	Does the client support writing resvTms	**Y** **only when client configuration ClientLN.iedName is empty**
Rp23	Does the client support reading owner	**N*** **see <additional items>; *compare Rp13**
Rp24	Does the device function only as test equipment?	**N**
	<additional items>	**In addition to settings for RCBs in the client configuration, a user may create HMI variables for any attribute with FC=BR|RP (manually or by online import). Then in HMI the current value of RCB's attribute may be read; and may be written (e.g. GI)**

ID	Description	Value / Clarification
		Note: written eventually only while RptEna is set to FALSE as else IED should refuse the updates of e.g. RptId, ResvTms etc.

16.8 PIXIT for Logging model

ID	Description	Value / Clarification
	<additional items>	logging services are not supported

16.9 PIXIT for GOOSE control block model

ID	Description	Value / Clarification
	<additional items>	system user may create HMI variables for any attribute with FC=GO (manually or by online import). Then in HMI the current values of GoCB's attributes may be read; and may be written (e.g. GoEna) Note: written eventually only while GoEna is set to FALSE as else IED should refuse the updates of e.g. GoID etc.

16.10 PIXIT for Control model

ID	Description	Value / Clarification
Ctl1	What control modes are supported?	Y status-only Y direct-with-normal-security Y sbo-with-normal-security Y direct-with-enhanced-security Y sbo-with-enhanced-security client checks current ctlModel value automatically and then executes

ID	Description	Value / Clarification
		command in corresponding control model; no need for additional configuration steps or pre-configuration from SCL **see client documentation for more details**
Ctl2	Is Time activated operate (operTm) supported?	**N**
Ctl3	Is "operate-many" supported?	**N**
Ctl4	Can the client set the test flag?	**Y**
Ctl5	What check conditions can be set?	**Y** synchrocheck **Y** interlock-check **there are two methods to set check conditions:** **1. by setting a value to HMI variable */Oper.Check[CO] if created** **2. by configuring in Command Processing modul of HMI property 'Qualifier of Command'** **see HMI documentation for details**
Ctl6	Which originator categories are supported and what is the originator identification?	**orCat =2 (station control)*** **orIdent ='zenon: nnn'*** **where: nnn - computer name** **' - apostrophe (CHAR(27))** ***default value; is a setting in the Client configuration**
Ctl7	Describe if and how the client sets/increments the ctlNum	**by each use* of ctlVal the client increments ctlNum from 0 to 127 and next turns to 0.** *** also by ctlModel=0 (no control possible)**
Ctl8	What does the client when its receives a LastApplicationError and describe how to view the additional cause?	**HMI user may create an internal "Command Info" variable (pro */Oper structure) for received AddCause item value and variables ControlRun and ControlState showing current state and success/fail of command. These internal variables can be displayed and/or stored**

ID	Description	Value / Clarification
		see client documentation for more details
Ctl9	What does the client when its receives a Select, SelectWithValue or Operate respond negative ?	**client sets corresponding values to intern "Command Info" (see Ctl8) variables ControlRun (-1=error) and AddCause; optionally writes a log entry and sets by HMI variable Oper.ctlVal status bit N_CONF** **see client documentation for more details**
Ctl10	Can the client change the control model via online services?	**Y**
Ctl11	What does the client when the ctlModel is not initialized in the SCL?	**client does not need ctlModel value in SCL; client checks current ctlModel value automatically via online services each time before is executing command; no mismatch possible**
Ctl12	What does the client when the ctlModel in SCD and in SERVER SIMULATOR is different?	**see Ctl11**
Ctl13	Describe how to view a ▸ CommandTermination request+ ▸ CommandTermination request- ▸ TimeActivatedOperateTermination request+ and request-	▸ **CommandTermination reception is visible in 'Command Info' internal HMI variables (see Ctl8); it may ends action timeout in HMI modul Command Processing, if used** ▸ **service TimeActivatedOperate is not supported** Note: **there is no special handling for negative response, only AddCause update (if server sends some). By use of Command Processing module for final success (or failure) of commnand the change of related ST/MX attribute cares.**
	<additional items>	

16.11 PIXIT for Time and time synchronization model

ID	Description	Value / Clarification
Tm1	Described how to view the internal time & quality or how to expose the timestamp and timestamp quality via the IEC 61850 interface	**View: client uses Windows time (local PC clock) and does not cooperate directly with a SNTP application. In HMI a user may create additional dynamical elements to view internal PC time and check clock quality using tools of own choice, e.g. SCADA Logic.** **Expose: in Operate and SelectWithValue services, client uses current value of additional HMI variables 'Settings': TimeQuality and TimeAccuracy.**
Tm 2	What time quality bits are supported?	**In Operate and SelectWithValue services variable TimeQuality may* update:** **Y LeapSecondsKnown** **Y ClockFailure** **Y ClockNotSynchronized** ***depending e.g. SNTP agent installed on the PC with HMI and user-implemented solution.**
Tm 3	What is the behavior when the time synchronization signal/messages are lost?	**PC (Windows) internal clock runs not synchronized anymore** **local SNTP agent can detect the signal loss and this information can be processed in HMI** **as result in HMI the variable TimeQuality may change value, if user adequate designed HMI project**
Tm 4	When is the quality bit "Clock failure" set?	**Optionaly HMI may force the Client to set ClockFailure (TimeQuality bit 1) when PC clock was reseted or for test purpose.** **see <additional items>**
Tm	When is the quality bit "Clock not	**TimeQuality bit 2 set if SNTP agent**

ID	Description	Value / Clarification
5	synchronised" set?	**informs HMI that it detects the loss of time synchronization signal**
	<additional items>	**Note: practical use for commands given from HMI on Windows PC has only ClockNotSynchronized:** ▸ **PC having severe Hardware problem - resulting in ClockFailure - is probably no more able to work in general. We recommend to use Redundant systems - compare As6.**

16.12 PIXIT for File transfer model

ID	Description	Value / Clarification
Ft1	Describe when or how to force the client to request GetServerDirectory(FILE) and what it does with the responded filenames	**client gives possibility to create two internal 'File transfer' string variables: a Command and a Directory variable. Setting value of Command to: "DIR <fileSpec>" results with the name or list of files in Directory variable** **see client documentation for more details**
Ft2	Does the client uses a wildcard in the GetServerDirectory(FILE) request	**No**
Ft3	Does the client support IED's that include the path in the file name in the GetServerDirectory(FILE) respond?	▸ **Y path included** ▸ **Y path not included** ▸ **Y list of files with a char separator**
Ft4	Does the client support IED's that use the fileseparator	▸ **N** "/" ▸ **Y** "\"
Ft5	What is the maximum file name size including path	**Operating System limit (Windows)**
Ft6	Can the client read a file with size 0	**Y**
Ft7	Are directory/file name case sensitive	**Not case sensitive** **only Command value is case sensitive: "DIR <fileSpec>" or "GET <fileSpec>",**

ID	Description	Value / Clarification
		the <fileSpec> is not case sensitive
Ft8	Maximum file size	**per design 4 GB** **tested* 25 MB** *** transfer of 25 MB may take about 30 seconds, bigger transfer per single file wasn't tested**
Ft9	Describe how the client behaves in case of: ▸ GetFile response- ▸ GetFileAttributes response- ▸ SetFile response-	▸ **Command variable value changes to "GET ERROR" or "DIR ERROR" (or "DEL ERROR" by service DeleteFile)** ▸ **SetFile is not supported**
	<additional items>	**For DeleteFile set Command variable value to "DEL <fileSpec>"**

16.13 PIXIT for Service Tracking model

ID	Description	Value / Clarification
Tr1	Which tracking services are supported by the client: ▸ BrcbTrk ▸ UrcbTrk ▸ LocbTrk ▸ GocbTrk ▸ MsvcbTrk ▸ UsvcbTrk ▸ SgcbTrk ▸ SpcTrk ▸ DpcTrk ▸ IncTrk ▸ EncTrk ▸ ApcFTrk	▸ Y* - BrcbTrk ▸ Y* - UrcbTrk ▸ Y* - LocbTrk ▸ Y* - GocbTrk ▸ Y* - MsvcbTrk ▸ Y* - UsvcbTrk ▸ Y* - SgcbTrk ▸ Y - SpcTrk ▸ Y - DpcTrk ▸ Y - IncTrk ▸ Y - EncTrk ▸ Y - ApcFTrk** ▸ Y - ApcIntTrk**

ID	Description	Value / Clarification
	▸ ApcIntTrk ▸ BscTrk ▸ IscTrk ▸ BacTrk ▸ GenTrk	▸ Y - BscTrk ▸ Y - IscTrk ▸ Y - BacTrk ▸ Y - GenTrk ***) per default only CST attributes, the rest of data attributes optionally.** ****) hint: attributes ctlVal.i and ctlVal.f are not included in "allCTS[SR]", create separatelly.** **See Tr2 and client documentation for more details.**
Tr2	Describe how to view the tracking objects or their attributes	**in additional HMI variables with reference: *LN/DO_xxx[SR] and 'driver object type' = 'service tracking', where xxx is "allCTS" or name of data attribute in CTS class or in another common data class possible in LTRK.** **See client documentation for more details**
	<additional items>	

17 Appendix E - TICS

This document specifies the TICS of the IEC 61850 interface in the client system: "**zenon Supervisor**/*Energy Edition*", Version *8.00*, further referred to as “Client”. The zenon Editor and Runtime are further referred to as "HMI".

MANDATORY EDITION 2 TISSUES

Below tables give an overview of the applicable mandatory Tissues.

SUPPORTED BY CLIENT:

- **"Y":** means that the client supports servers that have implemented the respective tissue.
- **"ni":** No impact on testing

- **"na":** not applicable if the client does not support the corresponding ACSI service(s)

17.1 Part 6

Tissue	Description	Supported by client
658	Tracking related features	Y
663	FCDA element cannot be a "functionally constrained logical node"	Y
668	Autotransformer modeling	Y
719	ConfDataSet - maxAttributes definition is confusing	na
721	Log element name	na
768	bType VisString65 is missing	Y
779	object references	na
788	SICS S56 from optional to mandatory	Y
789	ConfLdName as services applies to both server and client (supportsLdname)	Y
804	valKind and IED versus System configuration (valimport)	na
806	Max length of log name inconsistent between -6 and -7-2	ni
807	Need a way to indicate if "Owner" present in RCB	Y
822	Extension of IED capabilities	Y
823	ValKind for structured data attributes	Y
824	Short addresses on structured data attributes	Y
825	Floating point value	Y
845	SGCB ResvTms	Y
853	SBO and ProtNs	Y
855	Recursive SubFunction	na
856	VoltageLevel frequency and phases	ni
857	Function/SubFunction for ConductingEquipment	na
886	Missing 8-1 P-types	Y

Tissue	Description	Supported by client
901	tServices as AP or as IED element	Y
936	SupSubscription parameter usage is difficult	na
1175	IPv6 address lowercase only	na

17.2 Part 7-1

Tissue	Description	Supported by client
828	Data model namespace revision IEC 61850-7-4:2007[A]	Y
1151	simulated GOOSE disappears after 1st appearance when LPHD.Sim = TRUE	na
1196	Extensions to standardized LN classes made by third parties	na

17.3 Part 7-2

Tissue	Description	Supported by client
778	AddCause values – add value not- supported	Y
780	What are unsupported trigger option at a control block?	na
783	TimOper Resp- ; add Authorization check	na
786	AddCause values 26 and 27 are switched	Y
820	Mandatory ACSI services (use for PICS template)	Y
858	typo in enumeration ServiceType	ni

Tissue	Description	Supported by client
861	dchg of ConfRev attribute	Y
876	GenLogiclNodeClass and SGCB, GoCB, MsvCB, UsvCB	ni
1038	Loss of Info Detection After Resynch	Y
1062	Entrytime not used in CDC	ni
1071	Length of DO name	ni
1091	The sentence "The initial value of EditSG shall be 0", has to be stated in part 7.2 not in 8.1	Y
1127	Missing owner attribute in BTS and UTS	Y
1163	Old report in URCB	ni
1202	GI not optional	ni

17.4 Part 7-3

Tissue	Description	Supported by client
697	persistent command / PulseConfig	Y
698	Wrong case is BAC.dB attribute	Y
722	Units for 'h' and 'min' not in UnitKind enumeration.	ni
919	Presence Condition for sVC	ni
925	Presence of i or f attribute - Problem with writing	Y
926	Presence Conditions within RangeConfig	Y

17.5 Part 7-4

Tissue	Description	Supported by client
671	mistake in definition of Mod & Beh	Y

Tissue	Description	Supported by client
674	CDC of ZRRC.LocSta is wrong	Y
675	SIML LN	ni
676	Same data object name used with different CDC	Y
677	MotStr is used with different CDC in PMMS and SOPM LN classes	Y
679	Remove CycTrMod Enum	Y
680	SI unit for MHYD.Cndct	ni
681	Enum PIDAlg	ni
682	ANCR.ParColMod	ni
683	Enum QVVR.IntrDetMth	ni
685	Enum ParTraMod	ni
686	New annex H - enums types in XML	Y
694	Data object CmdBlk	ni
696	LSVS.St (Status of subscription)	ni
712	interpretation of quality operatorBlocked	Y
713	DO Naming of time constants in FFIL	Y
724	ANCR.Auto	Y
725	Loc in LN A-group	Y
734	LLN0.OpTmh vs. LPHD.OpTmh	Y
735	ISAF.Alm and ISAF.AlmReset	Y
736	PFSign	Y
742	GAPC.Str, GAPC.Op and GAPC.StrVal	Y
743	CCGR.PmpCtl and CCGR.FanCtl	Y
744	LN STMP, EEHealth and EEName	Y
772	LPHD.PwrUp/PwrDn shall be transient	Y
773	Loc, LocKey and LocSta YPSH and YLTC	Y

Tissue	Description	Supported by client
774	ITCI.LocKey	Y
775	KVLV.ClsLim and OpnLim	ni
776	LPHD.OutOv/InOv and LCCH.OutOv/InOv	ni
800	Misspelling in CSYN	Y
802	CCGR and Harmonized control authority	Y
808	Presence condition of ZMOT.DExt and new DOs	Y
831	Setting of ConfRevNum in LGOS	Y
838	Testing in Beh=Blocked	Y
844	MFLK.PhPiMax, MFLK.PhPiLoFil, MFLK.PhPiRoot DEL->WYE	Y
849	Presence conditions re-assessing in case of derived statistical calculation	Y
877	QVUB -settings should be optional	Y
909	Remove ANCR.ColOpR and ColOpL	Y
920	Resetable Counter is NOT resetable	Y
932	Rename AVCO.SptVol to AVCO.VolSpt	Y
939	Change CDC for ANCR.FixCol	Y
991	LGOS: GoCBRef (as well as LSVS.SvCBRef) should be mandatory	ni
1007	PTRC as fault indicator - Update of description required	ni
1044	TapChg in AVCO	Y
1077	Rename DOnames within LTIM	Y

Info

Tissues 675, 735, 772, 775, 776, 878 are not relevant for conformance testing

17.6 Part 8-1

Tissue	Description	Supported by client
784	Tracking of control (CTS)	Y
817	Fixed-length GOOSE float encoding	Y
834	File dir name length 64	Y
951	Encoding of Owner attribute	Y
1040	More associate error codes	Y
1178	Select Response+ is non-null value	Y