BECKHOFF New Automation Technology

Handbuch | DE

TS6511

TwinCAT 2 | PLC IEC 61850 Server

Supplement | Communication

08.11.2022 | Version: 1.1

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, für jede Installation und Inbetriebnahme die zu dem betreffenden Zeitpunkt veröffentliche Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiter entwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT®

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

1.2 Sicherheitshinweise

Sicherheitsbestimmungen

Beachten Sie die folgenden Sicherheitshinweise und Erklärungen!
Produktspezifische Sicherheitshinweise finden Sie auf den folgenden Seiten oder in den Bereichen Montage, Verdrahtung, Inbetriebnahme usw.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Erklärung der Symbole

In der vorliegenden Dokumentation werden die folgenden Symbole mit einem nebenstehenden Sicherheitshinweis oder Hinweistext verwendet. Die Sicherheitshinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

⚠ GEFAHR

Akute Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

⚠ WARNUNG

Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht Gefahr für Leben und Gesundheit von Personen!

⚠ VORSICHT

Schädigung von Personen!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen geschädigt werden!

HINWEIS

Schädigung von Umwelt oder Geräten

Wenn der Hinweis neben diesem Symbol nicht beachtet wird, können Umwelt oder Geräte geschädigt werden.

Tipp oder Fingerzeig

Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Produktübersicht

Das TwinCAT Supplementprodukt IEC 61850 Server besteht aus zwei wesentlichen Komponenten.

Zur Konfiguration des Datenmodells steht der so genannte "TwinCAT Telecontrol Configurator" zur Verfügung. Dieser erzeugt SPS Code, welcher in ein TwinCAT SPS Projekt eingebunden werden kann, wo die zweite Komponente, die SPS Bibliothek TcIEC61850Server.lib, zum Einsatz kommt. Diese SPS Bibliothek bindet den auf MMS (Manufacturing Message Specification) basierenden Kommunikationsstack ein und realisiert somit die Kommunikation nach IEC 61850.

Diese Dokumentation besteht entsprechend der beiden Komponenten aus den Teilen:

- TwinCAT Telecontrol Configurator [▶ 21]
- SPS Kommunikationsstack [▶ 43]

Systemvoraussetzungen

- Programmierumgebung:
  - XP, XPe;
  - TwinCAT Installation Level: TwinCAT PLC oder höher;
  - TwinCAT System Version 2.10.0 Build >= 1340 oder höher;
  - .NET Framework 3.5 für den TwinCAT Telecontrol Configurator
- Zielplattform:
  - PC or CX (x86): XP, XPe, CE 6 (image v3.06e oder höher);
  - CX (ARM): CE 5 (image v2.20e oder höher);
  - TwinCAT SPS-Laufzeitsystem Version 2.10.0 oder höher;

Produktkomponenten

- TwinCAT Telecontrol Configurator zur Konfiguration des standardisierten Datenmodells. Logical Devices, Logical Nodes, Data Object und Data Attributes können konfiguriert werden.
- TcIEC61850Server.Lib (SPS-Bibliothek mit dem Server-Baustein, diese Bibliothek muss in dem SPS-Projekt eingebunden werden, alle anderen Bibliotheken werden automatisch zum SPS-Projekt hinzugefügt);
- TcACSI.Lib (Basis ACSI-Datentypen für die Gerätemodellierung nach IEC 61850-7-2);
- TcIEC61850_8_1.Lib (SCSM: Mapping auf MMS-Dienste nach IEC 61850-8-1);
- TcMMS.Lib (MMS-Basisfunktionen, Bausteine, Datentypen);
- TcULOSI.Lib (Implementierung der Session-, Presentation- und ACSE-Schicht);
- TcBER.Lib (BER-Decoding/Encoding);
- TcRFC1006.Lib (Implementierung der Transportschicht);
- TcTPKT.Lib (Transport-Packetizer);
- TcCollections.Lib (Collection-Classes);
- TcSocketHelper.Lib (TCP/IP Hilfsfunktionen);
- TcpIp.Lib (TCP/IP Basisfunktionen);
- TwinCAT TCP/IP Connection Server;

Installation

Demoversion

Wenn Sie keinen gültigen Produkt-Schlüssel erworben haben, besteht die Möglichkeit den IEC 61850/61400-25 Server als eingeschränkte Demoversion zu nutzen. Dazu geben Sie bitte bei der Installation **Demo** als Produkt-Schlüssel ein. Ihnen stehen dann maximal drei logische Knoten zur Verfügung und nach

einer Stunde wird die Verbindung zu dem Client unterbrochen, was für einen grundlegenden Test der Kommunikation jedoch ausreichend ist. Nach dieser einen Stunde kann ein Restart des Servers durchgeführt werden.

Windows XP, XPe

Die SPS-Bibliotheken werden während der Installation in den ..\TwinCAT\Plc\Lib-Ordner hineinkopiert. Der TwinCAT TCP/IP Connection Server wird in die Liste der TwinCAT Server eingetragen. Beim TwinCAT Start wird der TCP/IP Connection Server automatisch gestartet und beim TwinCAT Stopp gestoppt.

Windows CE

Produktversion für das Laufzeitsystem unter Windows CE ist als separates Produkt verfügbar. Führen Sie folgende Schritte aus, wenn Sie eine Produktversion für Windows CE erworben haben:

- Installieren Sie das Produkt zuerst wie gewohnt auf Ihrem Programmier-PC. Die SPS-Bibliotheken werden während der Installation in den ..\TwinCAT\Plc\Lib-Ordner hineinkopiert.
- X86 CPU (CX1000, CX1020, IPC, usw.):
  - Nach der Installation finden Sie im Ordner: **...\TwinCAT\CE\TwinCAT PLC IEC 61850 Server CE** ein Cabinet-File für das CE-Laufzeitsystem.
  - Kopieren Sie die darin befindliche Datei: **TcPlcIec61850Svr_CE.I586.CAB** in einen Ordner auf dem CE-Laufzeitsystem.
- ARM CPU (CX9000, CX9010, usw.), TwinCAT PLC Library: IEC 60870-5-104 Unterstation (slave) **v2.0.4** und höher:
  - Nach der Installation finden Sie im Ordner: **...\TwinCAT\CE\TwinCAT PLC IEC 61850 Server CE** ein Cabinet-File für das CE-Laufzeitsystem.
  - Kopieren Sie die darin befindliche Datei: **TcPlcIec61850Svr_CE.ARMV4I.CAB** in einen Ordner auf dem CE-Laufzeitsystem.
- Auf dem CE-System: Installieren Sie (durch einen Doppelklick auf das Cabinet-File) die CE-Komponenten.
- Rebooten Sie das CE-Gerät. Der TCP/IP Connection Server wird mit dem CE-Betriebssystem automatisch gestartet.

Während der Installation werden Sie in einem Dialog gefragt, ob Sie "case sensitive" arbeiten möchten oder nicht. Wenn Sie diese Option anwählen, dann gilt dies für *alle* TwinCAT Projekte die sich auf dem Installationsgerät befinden. Nutzen Sie diese Option nicht, so werden die von der IEC 61850 bzw. IEC 61400-25 vorgesehenen Unterscheidungen in der Groß- und Kleinschreibung ignoriert und alles wird, wie in der SPS gewohnt, in Großbuchstaben kommuniziert.

Einführung/Tutorial

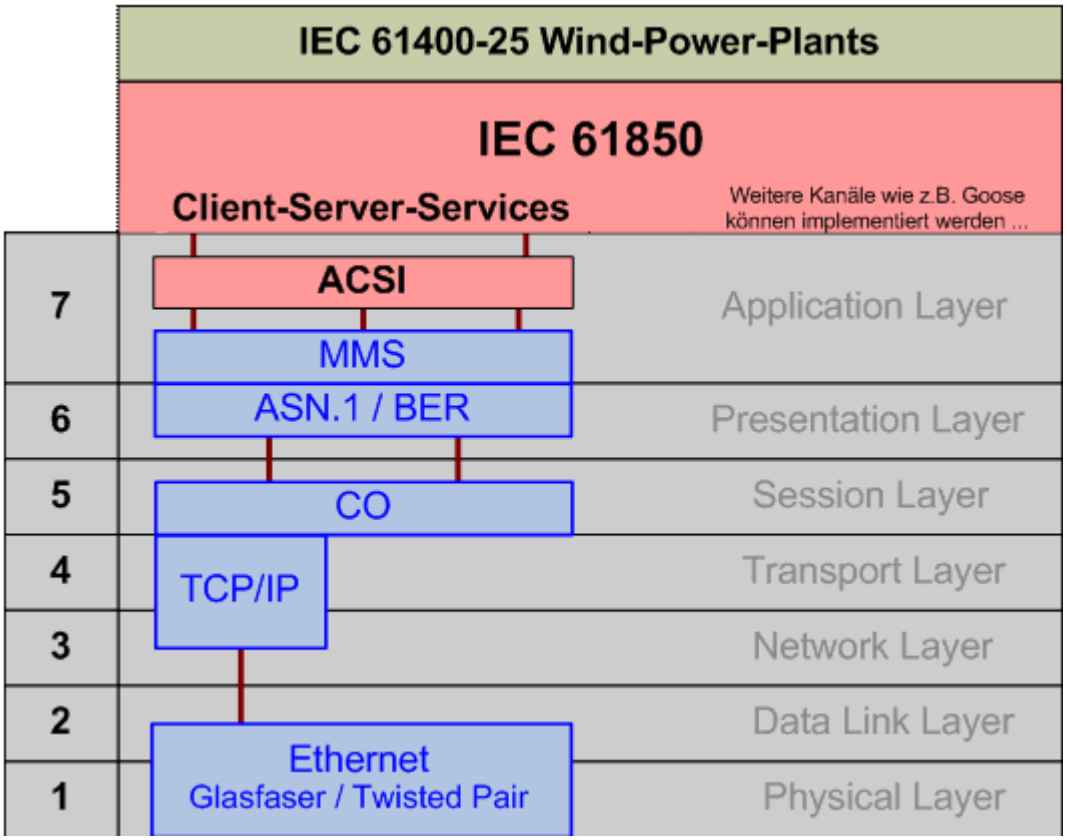
Für den TwinCAT IEC 61850 Server steht Ihnen eine ausführliche Schritt für Schritt Anleitung zur Verfügung. Programmcode für einen schnellen Kommunikationstest finden Sie unter Beispiele [▶ 62], im Anhang dieser Dokumentation.

Weiterführende Dokumentation

- Standard IEC 61850-7-1:2003
- Standard IEC 61850-7-2:2003
- Standard IEC 61850-7-3:2003
- Standard IEC 61850-7-4:2003
- Standard IEC 61850-8-1:2004
- Standard IEC 61400-25:2003
- Standard ISO/FDIS 9506-1
- Standard ISO/FDIS 9506-2
- TwinCAT TCP/IP Connection Server

3 Protokoll

Die IEC 61850 ist ein allgemeines Übertragungsprotokoll für die Schutz- und Leittechnik in elektrischen Schaltanlagen der Mittel- und Hochspannungstechnik (Stationsautomatisierung). Physikalisch basiert diese Kommunikation auf der Ethernet Technologie. Derzeit wird in TwinCAT die Server-Client-Kommunikation über MMS (Manufacturing Message Specification) und TCP/IP (Transmission Control Protocol/ Internet Protocol) unterstützt. Der Kommunikationsstack für den von Beckhoff entwickelten Server, wie er in der unteren Abbildung gezeigt wird, ist Schicht für Schicht in die TwinCAT PLC in Form von Bibliotheken eingebunden worden. Die folgende Abbildung zeigt den Aufbau der IEC 61400-25, welche speziell für Windenergieanlagen definiert worden ist und somit eine Spezialisierung der IEC 61850 ist.



Die wichtigsten Bibliotheken, welche unter anderem für das Mapping der IEC 61850 zuständig sind, werden hier kurz beschrieben.

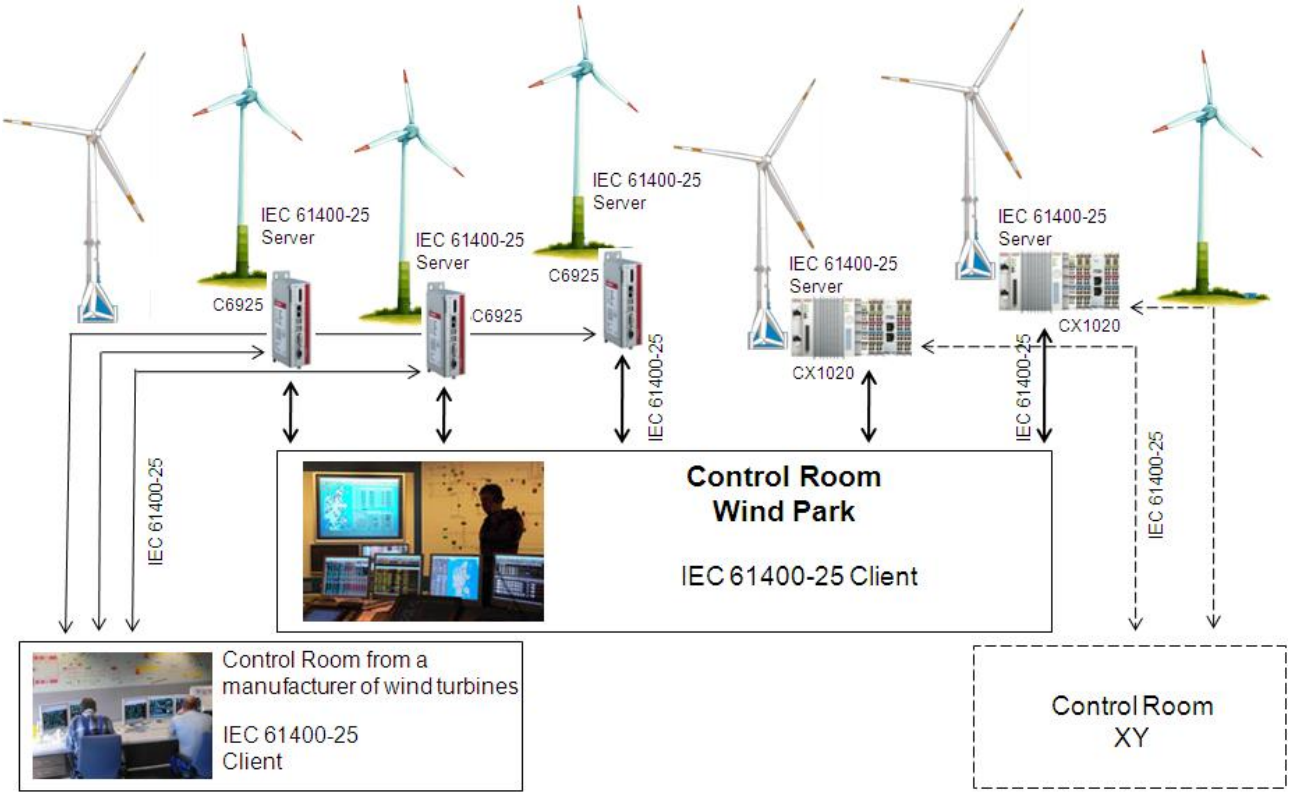
Bibliothek	Beschreibung
TcACSI.Lib	Für eine saubere Schnittstelle zu den unterlagerten Schichten ist diese Bibliothek zuständig. Das damit realisierte "Abstract Communication Service Interface" beinhaltet alle Basis-Datentypen der oben genannten Standards und die entsprechenden Bausteine, um durch die hierarchischen Strukturen der IEC 61850 bzw. IEC 61400-25 browsen zu können. Diese ACSI-Schnittstelle erlaubt eine universelle und zukunftssichere Anbindung der Norm an die unterlagerten Protokolle.
TcMMS.Lib	In der TcMMS.Lib sind die für die Kommunikation notwendigen Services implementiert worden.
TcASN1.Lib	In dieser Bibliothek stehen dem User alle Kodier- und Dekodierbausteine für die in BER (Basic Encoding Rules for ASN.1) kodierten Telegramme zur Verfügung.

Um die TCP/IP Kommunikation zu realisieren wird unterhalb der vorgestellten Bibliotheken der TwinCAT TCP/IP Connection Server benutzt. Dieses Supplementprodukt im Rahmen des Beckhoff Softwaresystems sorgt für die Realisierung eines oder mehrerer TCP/IP -Server/-Clients in der TwinCAT PLC.

Die im Jahr 2004 veröffentlichte IEC 61850 ist strikt objektorientiert und ihr Datenmodell ist hierarchisch aufgebaut.Dabei handelt es sich um 5 Hierarchiestufen welche durch die entsprechenden Bibliotheken der Standards in die TwinCAT PLC implementiert worden sind. Erklärungen und Beispiele für das Anlegen einer solchen Struktur finden Sie unter Protokoll Übersicht [▶ 11].

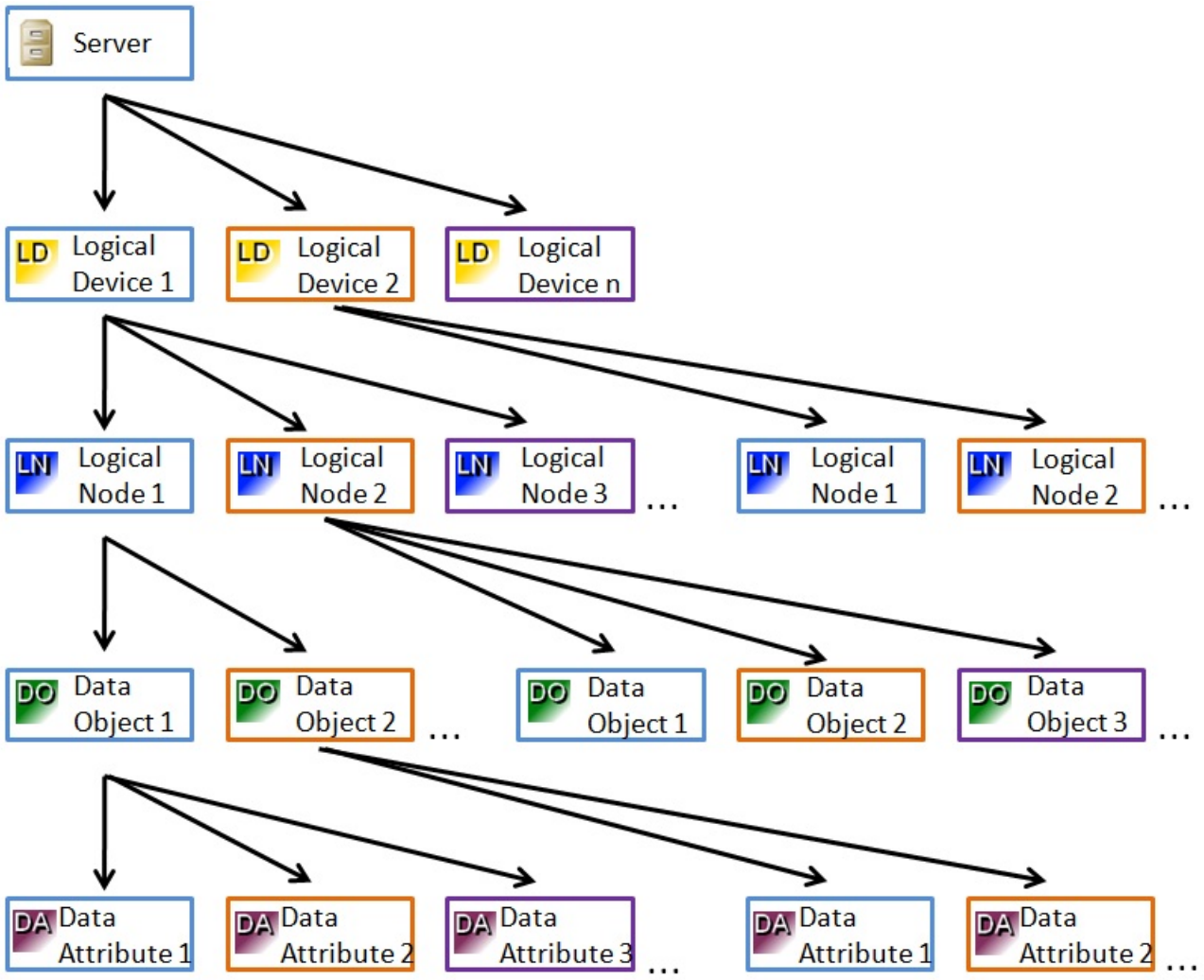
Anwendungsbeispiel

In diesem Fall ermöglich die Verwendung der IEC 61400-25 die Kommunikation aller in einem Windpark befindlichen Windkraftanlagen verschiedener Hersteller mit einer Zentralstation.
Die Standardisierung vermeidet die Verwendung herstellerspezifische Protokolle, welche zu einem erhöhten Applikationsaufwand führen. Die Firma BECKHOFF unterstützt mit der Implementierung der IEC 61850 bzw. IEC 61400-25 in der TwinCAT Automationsuite diesen Weg.



3.1 Datenmodell

Die IEC 61850 ist die Basisnorm für verschiedene Anwendungs-Spezialisierungen (u. a. für die IEC 61400-25), deshalb sind die Daten- oder Objektmodelle für alle Standards, die auf dieser Norm basieren gleich. Dieses Datenmodell gliedert sich in fünf Hierarchiestufen. Dabei geben die entsprechenden Standards alle nötigen Logical Nodes, Data Objects und Data Attributes vor. Der Anwender muss nur mit seinem Server für eine Anbindung an das Kommunikationssystem sorgen und seine eigenen Logical Devices anlegen.



Server	Die erste Hierarchieebene im Datenmodell der IEC 61850 bzw. IEC 61400-25 bildet der Server. Der Server bietet für ein Gerät einen Anschlusspunkt an, wodurch entsprechende auf Ethernet basierende Kommunikationssysteme angebunden werden können. In diesem Fall würde TwinCAT, mit den zum Protokoll zugehörigen Bibliotheken, auf einem Industrie PC oder Embedded Gerät einen solchen Server darstellen.
Logical Device	Die Logical Devices bilden die zweite Hierarchieebene im Datenmodell der IEC 61850 und IEC 61400-25. Diese Ebene unterteilt ein einzelnes physikalisches Gerät in mehrere separate Teile, so genannte Logical Device (Logisches Gerät). Der Vorteil dieser Unterteilung ist, dass zusammengehörige Funktionen, beziehungsweise Objekte, aufgrund ihrer Gemeinsamkeiten zusammenstehen und beispielsweise gemeinsam in einen anderen Betriebsmodus geschaltet werden können. Dabei können je nach Gerät verschiedene Logical Nodes aus der dritten Hierarchieebene implementiert werden.
Logical Node	Die Logical Nodes (logische Knoten) bilden die dritte Hierarchiestufe im Datenmodell der IEC 61850 bzw. IEC 61400-25. Sie repräsentieren die Informationen aller erdenklichen Teilfunktionen, die in der digitalen Stationsleittechnik und unter anderem in Windkraftanlagen vorkommen können. Dies sind sämtliche Schutzfunktionen, aber auch Automatisierungsfunktionen und Funktionen zur Verarbeitung von Mess- und Zählwerten. Für alle genormten Logical Nodes der IEC 61850 und IEC 61400-25 gibt es definierte Bezeichner, die immer aus einer Abkürzung mit vier Buchstaben bestehen. Die Abkürzung *XSWI* steht zum Beispiel für *Circuit Switch*. Ein Logical Devices kann aus mehreren solchen Knoten bestehen.

 **Data Object**	Ein Logical Node kann aus mehreren Data Objects (Datenobjekte) bestehen, diese bilden die vierte Hierarchieebene des IEC 61850 bzw. IEC 61400-25 Datenmodells. Eine Besonderheit dieser Hierarchiestufe ist, dass Datenobjekte ihrerseits hierarchisch verschachtelt sein können. So kann ein Datenobjekt aus mehreren Unterdatenobjekten bestehen, bevor letztendlich auf einen Basis- oder Komplexen-Datentyp in einer Common Data Class (CDC) zugegriffen wird.
 **Data Attribute**	Die unterste Hierarchieebene der IEC 61850 und IEC 61400-25 bilden die Datenattribute. Sie stellen die Detailinformationen oder Werte der Datenobjekte dar. Da für viele Datenobjekte immer derselbe Umfang an Datenattributen definiert ist, werden die Attribute eines Objektes in so genannten Common Data Classes (CDC) zusammengefasst. Dabei wird zu jedem Datenobjekt die zugehörige CDC angegeben, man kann auch sagen, dass die Datenobjekte Spezialisierungen der CDC sind.

3.2 Kompatibilitätsliste

Inhalt

1. Übereinstimmung mit dem Profil

2. MMS Konformität

1. Übereinstimmung mit dem Profil

N*in Entwicklung

A Profil-Support

Profil		Client	Server	Bemerkungen
A1	Client/Server	N	Y	
A2	GOOSE/GSE-Management	N	N	nur GOOSE-, nicht GSE-Management
A3	GSSE	N	N	

Profil		Client	Server	Bemerkungen
A4	Time sync	N	N*	Zeitgenauigkeit: 1ms (Leistungsklasse T1) Zeitauflösung: ca. 0,9ms

T Profil-Support

Profil		Client	Server	Bemerkungen
T1	TCP/IP-Profil	N	Y	
T2	OSI T-Profil	N	N	
T3	GOOSE/GSE T-Profil	N	N	nur GOOSE, nicht GSE
T4	GSSE T-Profil	N	N	
T5	Time sync T-Profil	N	N	

MMS InitiateRequest – allgemeine Parameter

InitiateRequest	Client-CR	Server-CR
InitiateRequest		
localDetailCalling	N	Y
proposedMaxServOutstandingCalling	N	Y
proposedMaxServOutstandingCalled	N	Y
initRequestDetail	N	Y
InitiateRequestDetail		
proposedVersionNumber	N	Y
proposedParameterCBB	N	Y
servicesSupportedCalling	N	Y

MMS InitiateResponse – allgemeine Parameter

InitiateResponse	Client-CR	Server-CR
InitiateResponse		
localDetailCalled	N	Y
negotiatedMaxServOutstandingCalling	N	Y
negotiatedMaxServOutstandingCalled	N	Y
initResponseDetail	N	Y
InitiateResponseDetail		
negotiatedVersionNumber	N	Y
negotiatedParamterCBB	N	Y
servicesSupportedCalled	N	Y

Client/Server – unterstützte Services

IEC 61850-7-2 Model	IEC 61850-7-2 Services	Implementiert (Y/N)
Server	GetServerDirectory	Y
Association	Associate	Y
	Abort	Y

IEC 61850-7-2 Model	IEC 61850-7-2 Services	Implementiert (Y/N)
	Release	Y
Logical Device	GetLogicalDeviceDirectory	Y
Logical Node	GetLogicalNodeDirectory	Y
	GetAllDataValues	Y
Data	GetDataValues	Y
	SetDataValues	Y
	GetDataDirectory	Y
	GetDataDefinition	Y
Data Set	GetDataSetValues	N*
	SetDataSetValues	N*
	CreateDataSet	Y
	DeleteDataSet	Y
	GetDataSetDirectory	Y
Substitution	GetDataValues	N
	SetDataValues	N
Setting Group Control Block	SelectActiveSG	N
	SelectEditSG	N
	SetSGValues	N
	ConfirmEditSGValues	N
	GetSGValues	N
	GetSGCBValues	N
Report Control Block	Report	N*
	GetBRCBValues	N*
	SetBRCBValues	N*
	GetURCBValues	N*
	SetURCBValues	N*
LOG Control Block	GETLCBValues	N
	SETLCBValues	N
	GetLogStatusValues	N
	QueryLogByTime	N
	QueryLogAfter	N
GOOSE	GetCoCBValues	N
	SetGoCBValues	N
GSSE	GetGsCBValues	N
	SetGsCBValues	N
Control	Select	N
	SelectWithValue	N
	Cancel	N
	Operate	N
	CommandTermination	N
	TimeActivatedOperate	N
FILE transfer	GetFile	N
	SetFile	N
	DeleteFile	N
	GetFileAttributeValues	N

2. MMS-Konformität

MMS Initiate request – allgemeine Parameter

MMS Service unterstützte CBB	Client-CR		Server-CR	
	realisiert	Wert/ Bereich	realisiert	Wert/ Bereich
status			N	
getNameList			Y	
identify			Y	
rename			N	
read			Y	
write			Y	
getVariableAccessAttributes			Y	
defineNamedVariable			N	
defineScatteredAccess			N	
getScatteredAccessAttributes			N	
deleteVariableAccess			N	
defineNamedVariableList			Y	
getNamedVariableListAttributes			Y	
deleteNamedVariableList			Y	
defineNamedType			N	
getNamedTypeAttributes			N	
deleteNamedType			N	
input			N	
output			N	
takeControl			N	
relinquishControl			N	
defineSemaphore			N	
deleteSemaphore			N	
reportPoolSemaphoreStatus			N	
reportSemaphoreStatus			N	
initiateDownloadSequence			N	
downloadSegment			N	
terminateDownloadSequence			N	
initiateUploadSequence			N	
uploadSegment			N	
terminateUploadSequence			N	
requestDomainDownload			N	
requestDomainUpload			N	
loadDomainContent			N	
storeDomainContent			N	
deleteDomain			N	
getDomainAttributes			N	
createProgramInvocation			N	
deleteProgramInvocation			N	
start			N	
stop			N	
resume			N	
reset			N	
kill			N	

getProgramInvocationAttributes			N	
obtailFile			N	
devineEventCondition			N	
deleteEventCondition			N	
getEventConditionAttributes			N	
getEventConditionStatus			N	
getEventConditionMonitoring			N	
triggerEvent			N	
defineEventAction			N	
deleteEventAction			N	
alterEventEnrollment			N	
reportEventEnrollmentStatus			N	
getEventEnrollmentAttributes			N	
acknowledgeEventNotification			N	
getAlarmSummary			N	
getAlarmEnrollmentSummary			N	
readJournal			N	
writeJournal			N	
initializeJournal			N	
reportJournalStatus			N	
createJournal			N	
deleteJournal			N	
fileOpen			N	
fileRead			N	
fileClose			N	
fileRename			N	
fileDelete			N	
fileDirectory			N	
unsolicitedStatus			N	
informationReport			N*	
eventNotification			N	
attachToEventCondition			N	
attachToSemaphore			N	
conclude			Y	
cancel			N*	
getDataExchangeAttributes			N	
exchangeData			N	
defineAccessControlList			N	
getAccessControlListAttributes			N	
reportAccessControlledObjects			N	
deleteAccessControlList			N	
alterAccessControl			N	
reconfigureProgramInvocation			N	

MMS Parameter-Konformität Building Block (CBB)

MMS Service unterstützte CBB	Client-CR		Server-CR	
	realisiert	Wert/ Bereich	realisiert	Wert/ Bereich
STR1			Y	
STR2			Y	

VNAM			Y	
VALT			Y	
VADR			N	
VSCA			N	
TPY			Y	
VLIS			Y	
REAL			N	
CEI			N	

Alternative AccessSelection Konformitätserklärung

AlternateAccessSelection	Client-CR		Server-CR	
	realisiert	Wert/ Bereich	realisiert	Wert/ Bereich
accessSelection			N	
component			N	
index			N	
indexRange			N	
allElements			N	
alternateAccess			Y	
selectAccess			N	
component			N	
index			N	
indexRange			N	
allElements			N	

Variable Access Konformitätserklärung

VariableAccessSpecification	Client-CR		Server-CR	
	realisiert	Wert/ Bereich	realisiert	Wert/ Bereich
listOfVariable			Y	
variableSpecification			N*	
alternateAccess			N*	
variableListName			N*	

Variable Konformitätserklärung

VariableSpecification	Client-CR		Server-CR	
	realisiert	Wert/ Bereich	realisiert	Wert/ Bereich
name			Y	
address			N	
variableDescription			N	
scatteredAccessDescription			N	
invalidated			N	

Read Konformitätserklärung

Read	Client-CR		Server-CR	
	realisiert	Wert/ Bereich	realisiert	Wert/ Bereich
Request				
specificationWithResult			N	
variableAccessSpecification			Y	
Response				

variableAccessSpecification			N	
listOfAccessResult			Y	

GetVariableAccessAttributes Konformitätserklärung

GetVariableAccessAttributes	Client-CR		Server-CR	
	realisiert	Wert/ Bereich	realisiert	Wert/ Bereich
Request				
name			Y	
address			N	
Response				
mmsDeletable			N*	
address			N	
typeSpecification			Y	

DefineVariableAccessAttributes Konformitätserklärung

DefineVariableAccessAttributes	Client-CR		Server-CR	
	realisiert	Wert/ Bereich	realisiert	Wert/ Bereich
Request				
variableListName			N	
listOfVariable			N	
variableSpecification			N	
alternateAccess			N	
Response			N	

GetNamedVariableList Konformitätserklärung

GetNamedVariableListAttributes	Client-CR		Server-CR	
	realisiert	Wert/ Bereich	realisiert	Wert/ Bereich
Request				
objectName			Y	
Response				
mmsDeletable			N*	
listOfVariable			Y	
variableSpecification			Y	
alternateAccess			N	

DeleteNamedVariableList Konformitätserklärung

DeleteNamedVariableList	Client-CR		Server-CR	
	realisiert	Wert/ Bereich	realisiert	Wert/ Bereich
Request				
Scope			N	
listOfVariableListName			Y	
domainName			Y	
Response				
numberMatched			Y	
numberDeleted			Y	
DeleteNamedVariableList-Error			Y	

GOOSE Konformitätserklärung

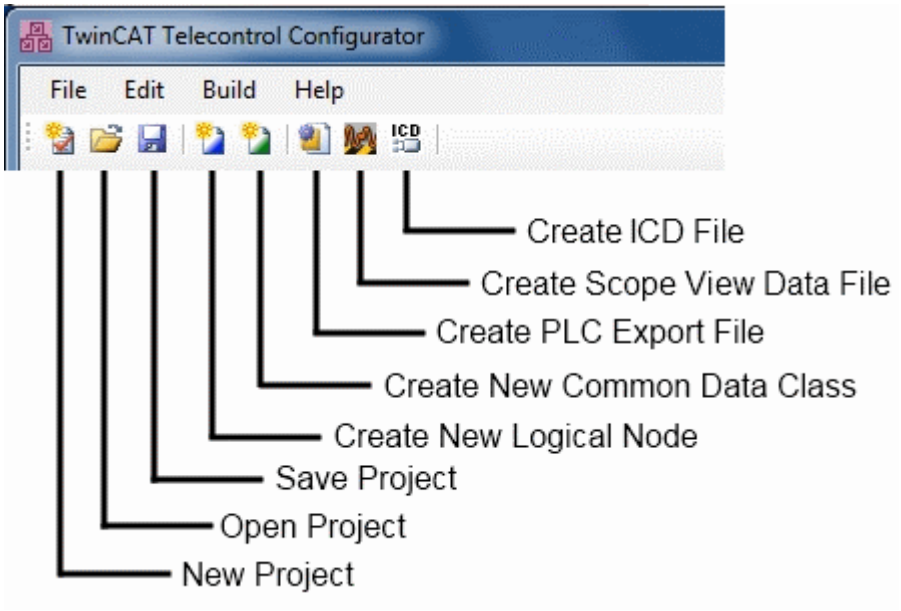
GOOSE	Subscriber	Publisher	Wert/Kommentar
GOOSE Services	N	N	
SendGOOSEMessage	N	N	
GetGoReference	N	N	
GetGOOSEElementNumber	N	N	
GetGoCBValue	N	N	
SetGoCBValue	N	N	
GSENotSupported	N	N	
GOOSE Control Block	N	N	ReadOnly

Sehen Sie dazu auch

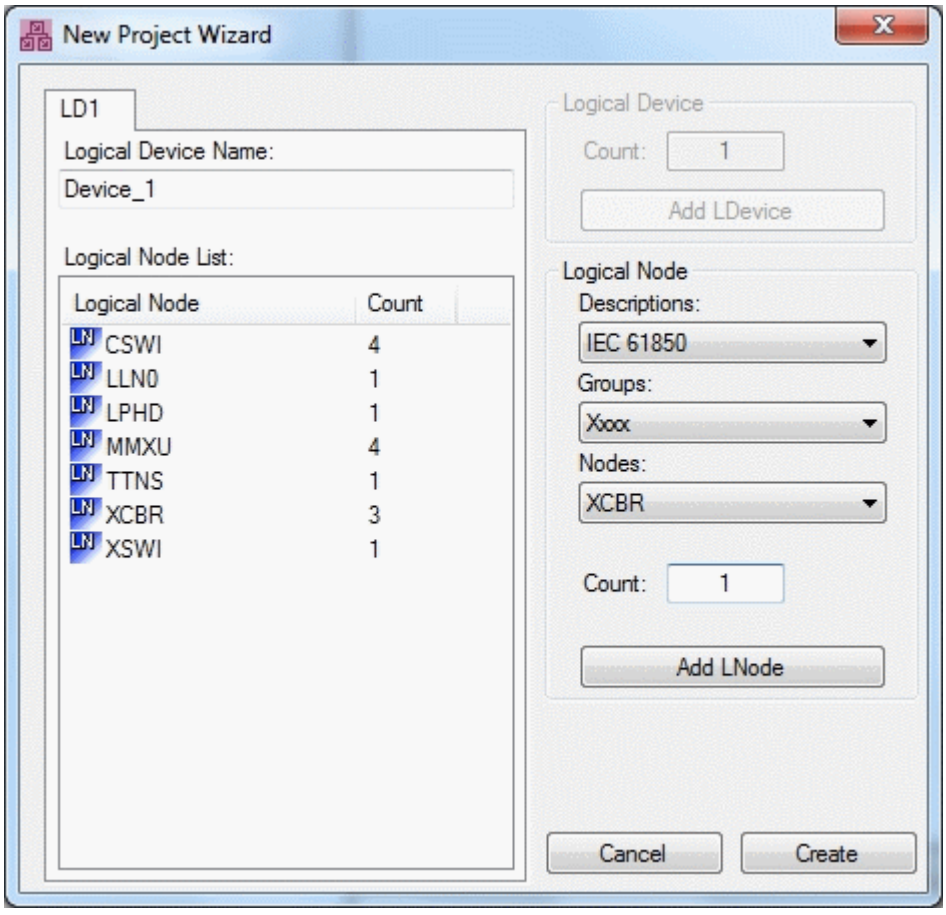
Kompatibilitätsliste [▶ 14]

4 Konfigurator

Der TwinCAT Telecontrol Configurator ermöglicht eine ideale Trennung von Konfiguration des standardisierten Datenmodells und der Programmierung in der SPS. Der nach der Konfiguration erzeugte SPS Code kann einfach als Export-File in das PLC Control eingebunden und verwendet werden.



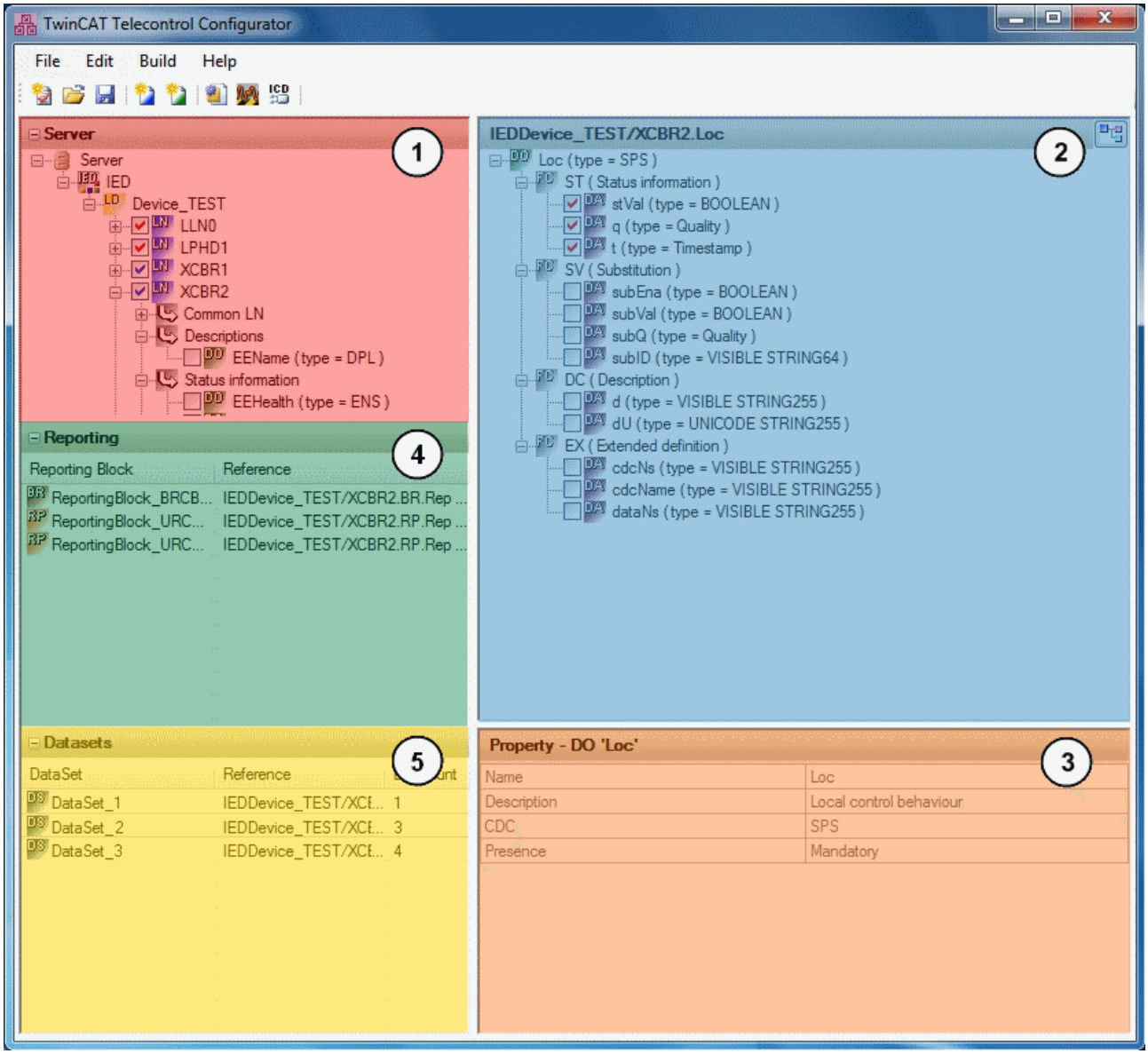
Durch die Betätigung des *New Project* Buttons in der Toolbar des Konfigurators öffnet sich ein Wizard-Fenster, wie es in der nächsten Abbildung gezeigt wird. Dieser Wizard hilft eine erste Startkonfiguration zu erstellen. Sie haben die Möglichkeit einen Logical Device Namen einzutragen und die Logical Nodes aus den verschiedenen Standards zu wählen. Die Mandatory Nodes *LLN0* und *LPHD* müssen nicht ausgewählt werden, diese werden automatisch eingefügt. Ein weiterer Vorteil liegt darin, dass Sie die Anzahl der Instanzen für einen Logical Node angeben können. Dies verringert deutlich die Konfigurationszeit.



Neben weiteren Standardfunktionen wie *Open* und *Save Project*, bietet der TwinCAT Telecontrol Configurator auch die Möglichkeit eigene, also private, Logical Nodes und Common Data Classes zu definieren (*Create New Logical Node* und *Create New Common Data Class*). Dabei müssen die vorgegebenen Datentypen der IEC 61850 verwendet werden, um die Konformität zu bewahren. Nähere Informationen zur Erstellung von anwenderspezifischen Erweiterungen finden Sie unter Private Logical Nodes [▶ 35] und Private Common Data Classes [▶ 37].

Die wichtigste Funktionalität dieses Konfigurators besteht aber darin, aus der erstellten Konfiguration eine SPS Exportdatei zu erstellen (*Create PLC Export File*). Dabei werden nur die Datentypen und Objekte exportiert, welche wirklich in der Konfiguration benötigt werden. Dies erspart Ressourcen, was dieses TwinCAT Supplementprodukt sehr skalierbar macht. Details über die Inhalte der Exportdatei und zur Einbindung in ein TwinCAT SPS Projekt finden Sie hier [▶ 38].

Alle Funktionalitäten der Toolbar sind auch in der Menüleiste des Konfigurators untergebracht. Unter *File* steht zudem die Funktion *Save As...* zur Verfügung. Des Weiteren gliedert sich die Oberfläche des Konfigurators in fünf wesentliche Teile:

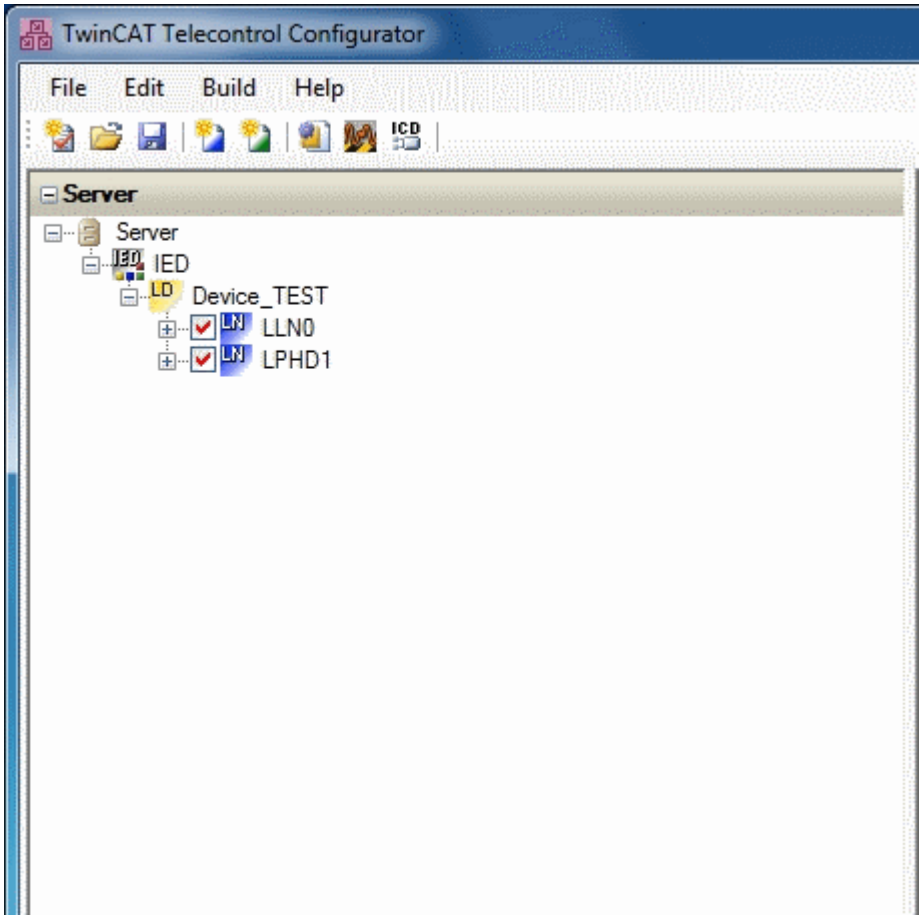


[▶ 23] 1	**Server Konfiguration:** Die Server Übersicht beinhaltet alle konfigurierten Logical Devices und liefert auch dessen Logical Nodes: Server Konfiguration [▶ 23]

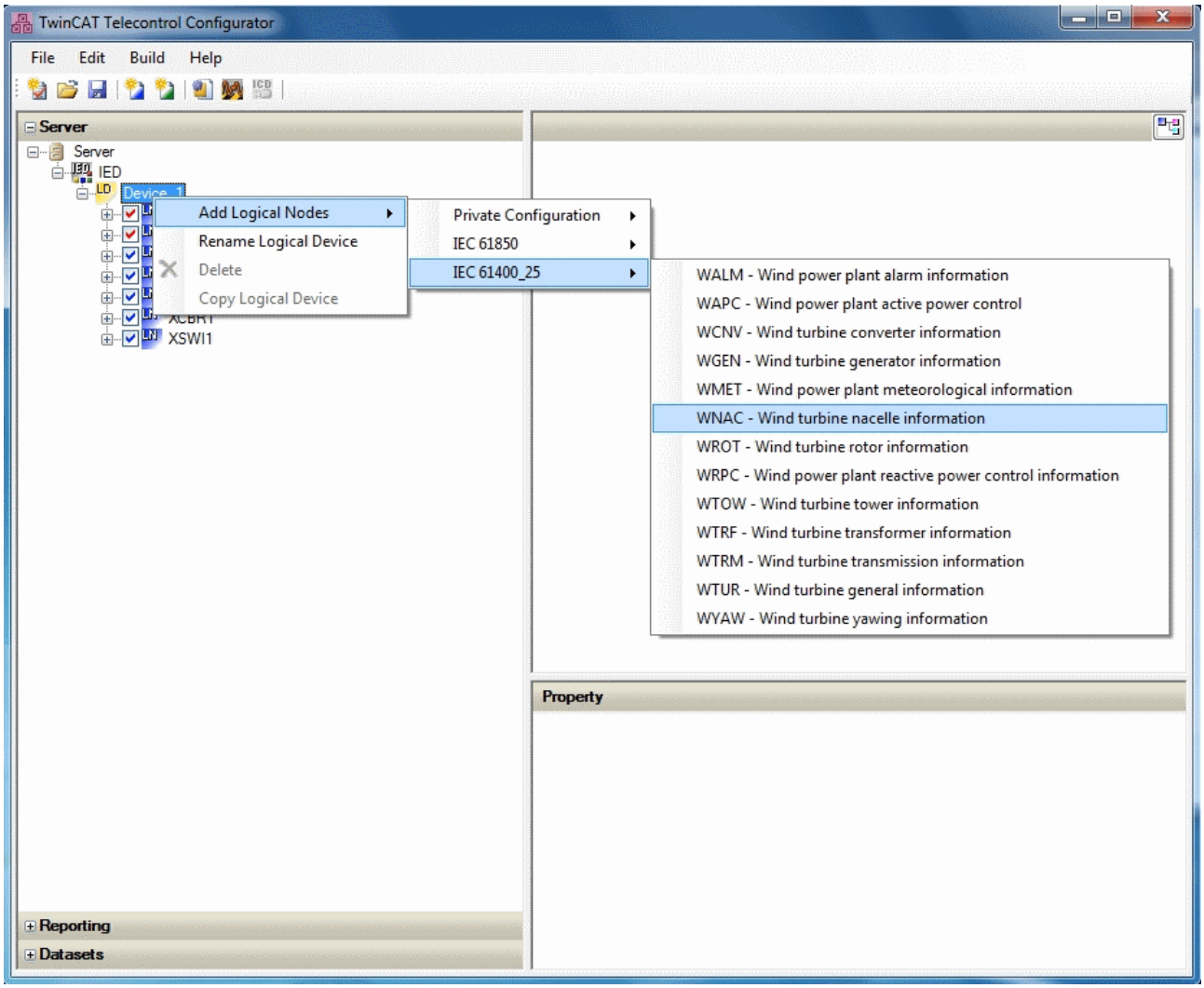
[▶ 24] 2	**Daten Objekt Konfiguration:** Diese Übersicht beinhaltet alle Datenattribute des in der Server Konfiguration angewählten Datenobjektes: Daten Objekt Konfiguration [▶ 24]
[▶ 29] 3	**Objekteigenschaften:** In diesem Fenster werden zu den angewählten Objekten die entsprechenden Eigenschaften dargestellt: Objekteigenschaften [▶ 29]
[▶ 31] 4	**Reporting:** In diesem Fenster werden alle erzeugten Reporting Blöcke aufgelistet: Reporting [▶ 31]
[▶ 33] 5	**DataSets:** In diesem Fenster werden alle erzeugten DataSets aufgelistet: DataSets [▶ 33]

4.1 Server Konfiguration

In der Server Übersicht des TwinCAT Telecontrol Konfigurators sehen Sie alle konfigurierten Logical Devices und die dazugehörigen Logical Nodes. Sie haben die Möglichkeit in diesem Fenster neue Logical Nodes zur bestehenden Konfiguration hinzuzufügen und die optionalen Daten Objekte (Data Objects) eines Knotens zu aktivieren.



Wird eine neue Konfiguration geöffnet, so werden automatisch die beiden Mandatory Logical Nodes LLN0 und LPHD angelegt. Dass diese "Mandatory" sind, erkennen Sie auch an den roten Haken in den Checkboxen der Knoten. Dieses Unterscheidungsmerkmal gilt für alle Objekte im Konfigurator. Werden optionale Objekte aktiviert so erhalten diese einen blauen Haken. Sie haben auch die Möglichkeit den Logical Device Namen durch Anklicken zu ändern.

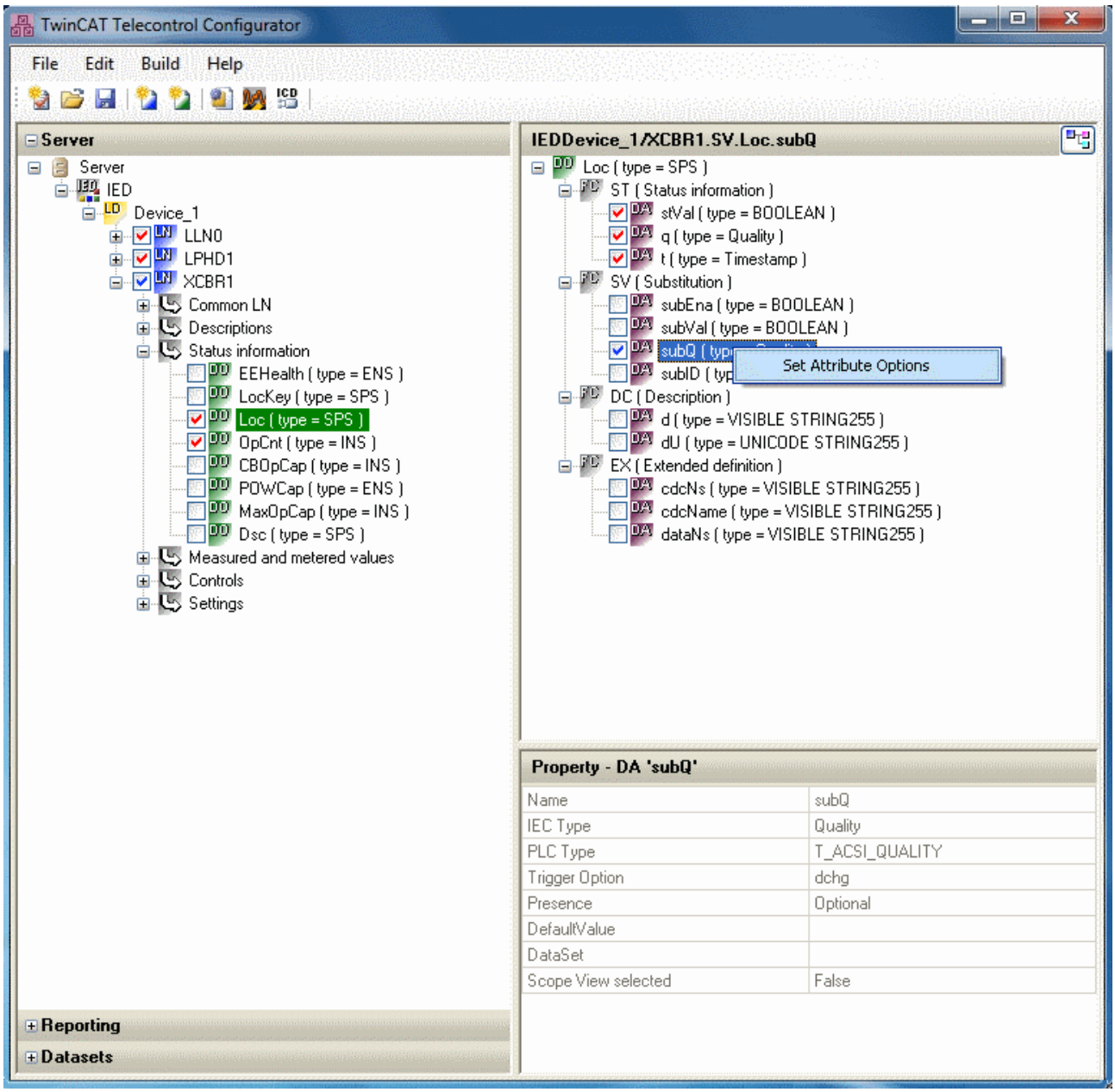


Mit einem Rechtsklick auf das jeweilige Logical Device öffnet sich das Kontextmenu mit der Option *Add Logical Node*. In einem weiteren Fenster können Sie einen Logical Node aus verschiedenen Datenbanken wählen. Es gibt immer eine Private und eine IEC 61850 Datenbank. Abhängig von der Produktlizenz, welche Sie erworben haben, werden weitere Datenbanken frei geschaltet. In diesem Beispiel steht auch die Datenbank des Windkraft-Standards IEC 61400-25 zur Verfügung.

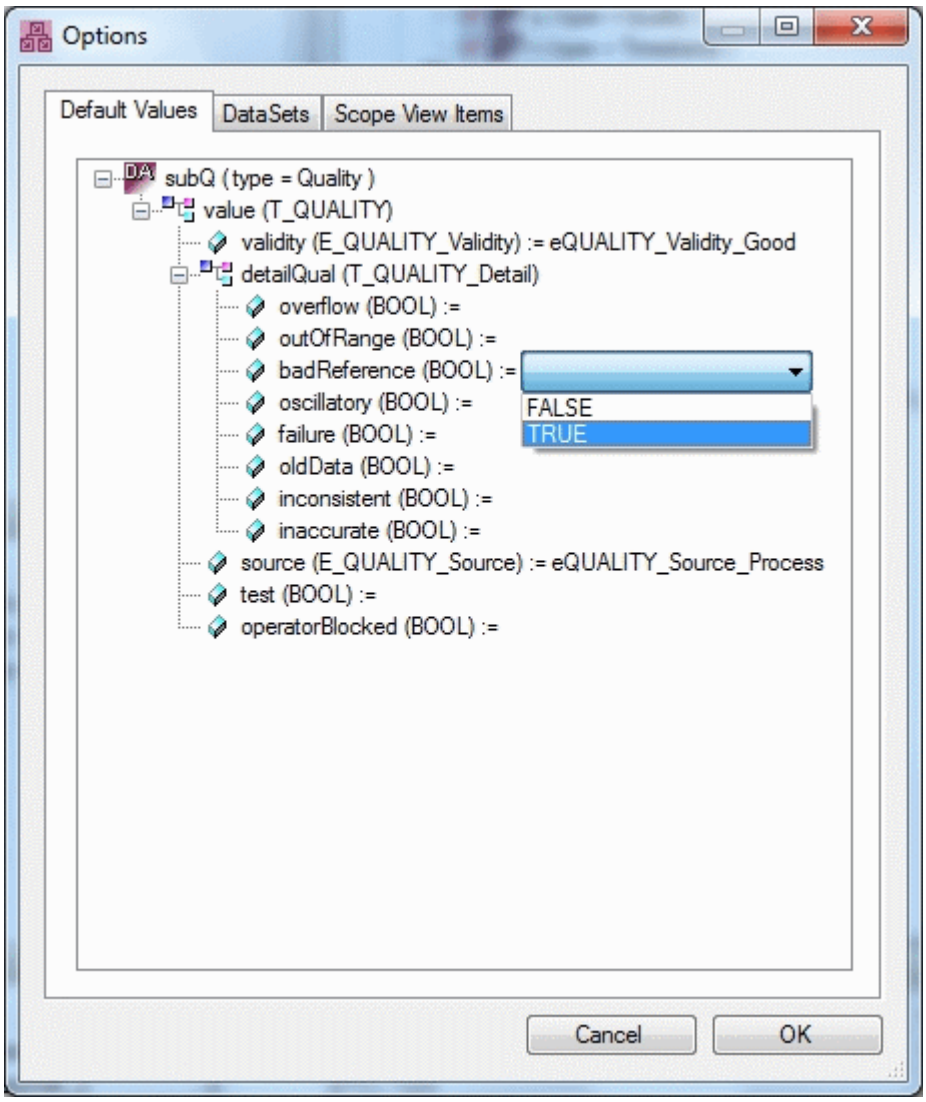
Die Konfiguration der einzelnen Daten Objekte können Sie rechts im Konfigurator in der Daten Objekt Konfiguration [▶ 24] vornehmen.

4.2 Daten Objekt Konfiguration

In der Datenobjekt Übersicht rechts im Telecontrol Configurator können die Datenattribute der verschiedenen Datenobjekte konfiguriert werden.



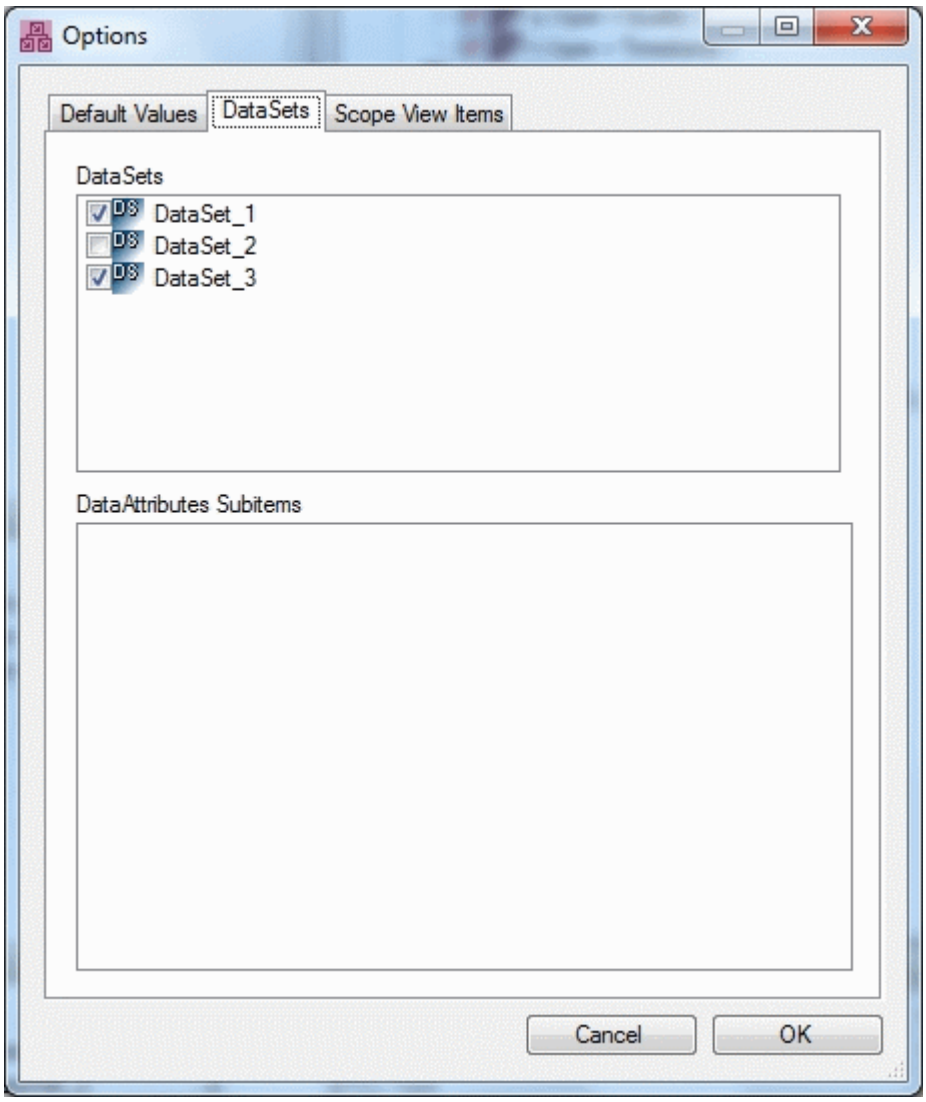
Wie in der Server Konfiguration [▶ 23] können Sie hier die optionalen Datenattribute durch die entsprechenden Checkboxen aktivieren. Darüber hinaus ist es möglich durch einen Doppelklick oder mit Hilfe des Kontextmenus eines Datenattributs in den *Set Attribute Options* Dialog zu gelangen.

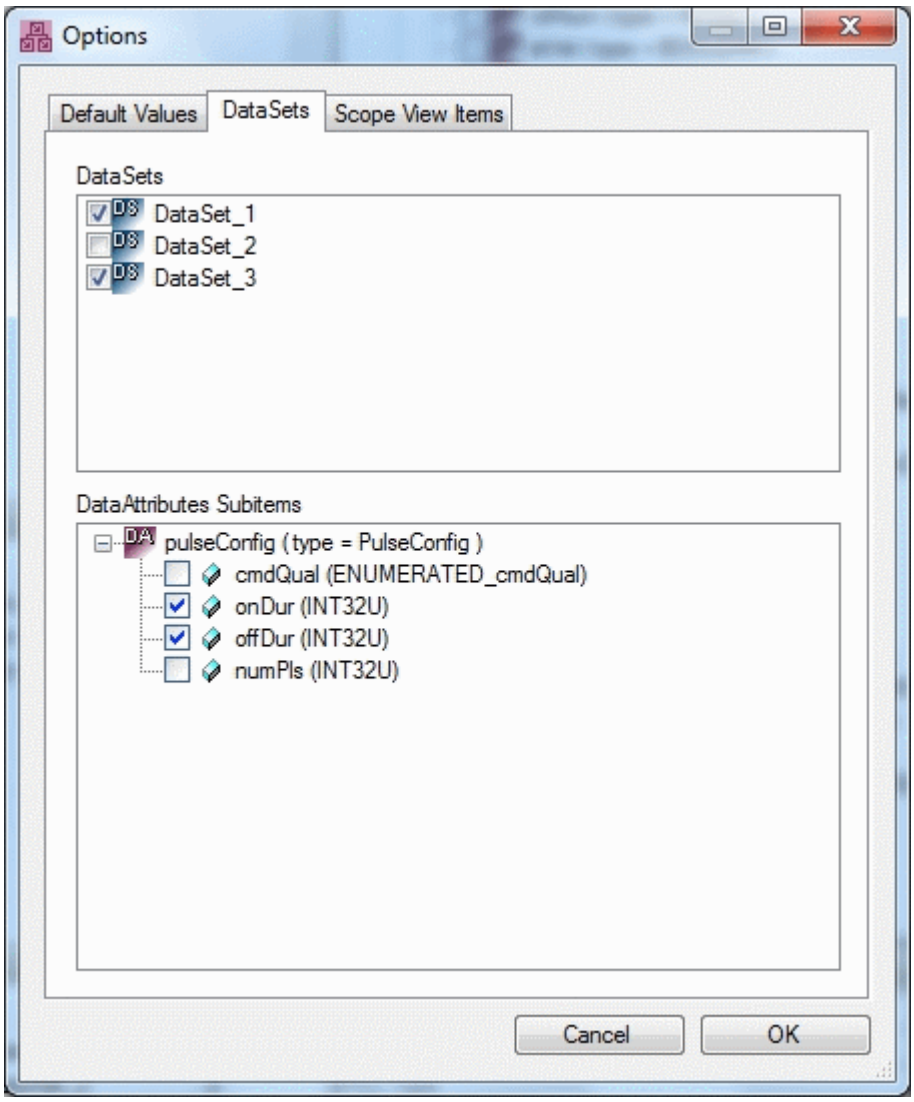


Im *Set Attribute Options* Dialog ist es unter anderem möglich Initialwerte für die Datenattribute zu setzen. Dabei werden dem Anwender mit Ausnahme von Zahlenwerten alle Auswahlmöglichkeiten für das entsprechende Attribut angeboten. Diese Werte werden später in den SPS Code exportiert.

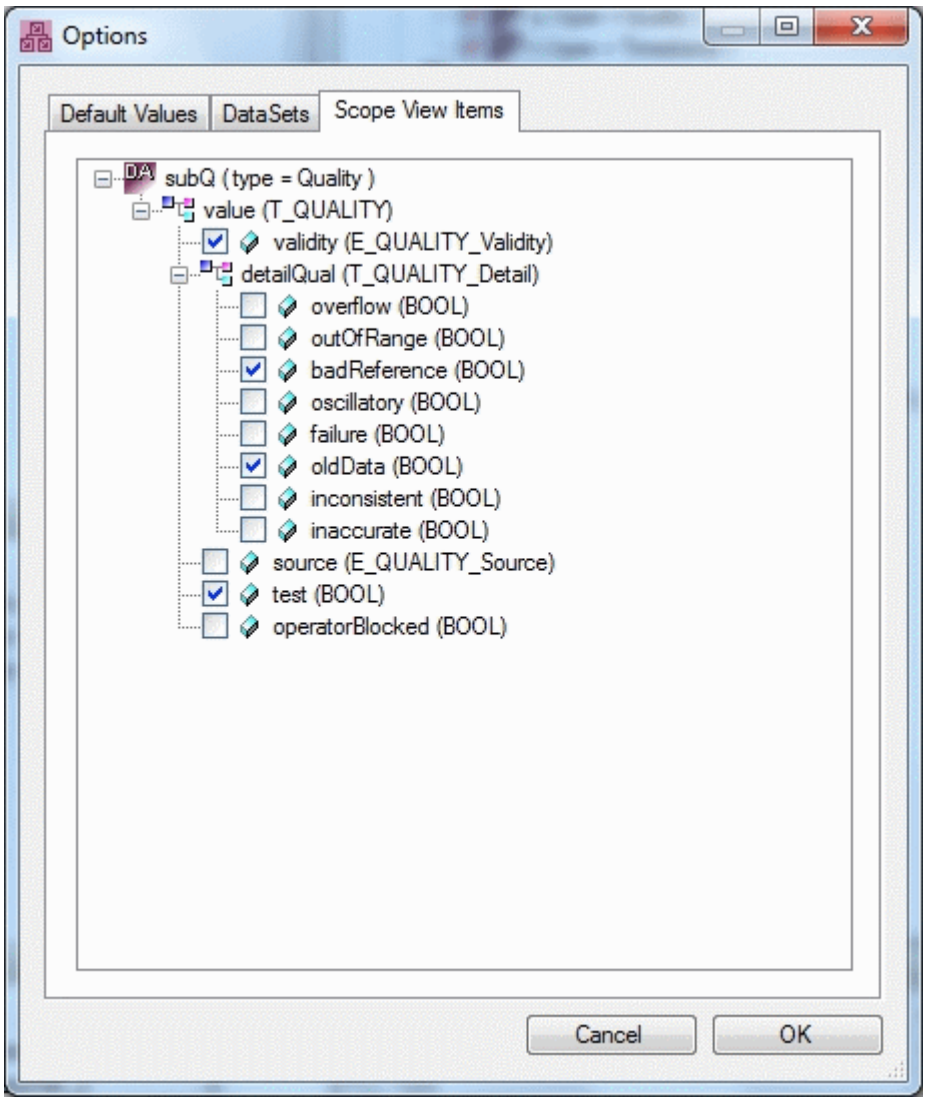
Wählen Sie ein Attribut an, so wie es im ersten Bild dieser Seite gezeigt wird, so erhalten Sie in der Übersicht für die Objekteigenschaften [▶ 29] die Information über den eingestellten Initialwert.

Des Weiteren besteht die Möglichkeit unter dem Karteireiter "DataSets" das ausgewählte Attribut einem DataSet zuzuordnen. Enthält das Attribut Subelemente so werden diese in einem Baum angezeigt wie Sie in den folgen Bildern sehen.


Options
Default Values
DataSets
Scope View Items
DataSets
DataSet_1
DataSet_2
DataSet_3
DataAttributes Subitems
Cancel
OK



Unter dem Karteireiter "Scope View Items" besteht die Möglichkeit alle Elemente des ausgewählten Attributs für eine Scope View Konfiguration auszuwählen.



4.3 Objekteigenschaften

In den Objekteigenschaften werden Informationen zu dem aktuell selektierten Objekt angezeigt.

Logical Node:

Property - LN 'XCBR'	
Name	XCBR
Description	Circuit breaker
Group	Switchgear

Name: Name des Logical Nodes, wie er im Datenmodell-Baum erscheint.

Description: Liefert eine Kurzbeschreibung, was die aus 4 Zeichen bestehende Abkürzung bedeutet.

Group: Gibt an zu welcher übergeordneten Gruppe der entsprechende Logical Node (IEC 61850-7-4) gehört.

Data Object:

Property - DO 'BlkOpn'	
Name	BlkOpn
Description	Block opening
CDC	SPC
Presence	Mandatory

Name: Name des Data Objects, wie er im Datenmodell-Baum erscheint.

Description: Liefert eine Kurzbeschreibung darüber, welche Aufgabe dieses Datenobjekt hat.

CDC: Gibt an zu welcher Common Data Class (IEC 61850-7-3) das ausgewählte Datenobjekt gehört.

Presence: Gibt an, ob es sich bei dem gewählten Datenobjekt um ein "Mandatory" oder "Optional" Objekt handelt.

Data Attribute:

Property - DA 'subEna'	
Name	subEna
IEC Type	BOOLEAN
PLC Type	T_ACSI_BOOLEAN
Trigger Option	dchg
Presence	Optional
DefaultValue	(value:=TRUE)
DataSet	DataSet_1
Scope View selected	True

Name: Name des Data Attribute, wie er im Datenmodell-Baum erscheint.

IEC Type: Gibt den Datentyp des Attributs an, wie er in der IEC 61850 definiert ist.

PLC Type: Gibt den Datentyp des Attributs an, wie er in der SPS gemappt wird.

Trigger Option: Gibt die Trigger Option für das jeweilige Attribut an. Dies ist entscheidend für die Data Sets bzw. für das Reporting.

Presence: Gibt an, ob es sich bei dem gewählten Attribut um ein "Mandatory" oder "Optional" Attribut handelt.

Default Value: Zeigt den gesetzten Defaultwert an, wie er dann auch in der SPS bei einem Export gesetzt wird.

DataSet: Listet die DataSets auf in denen das Attribut enthalten ist.

Scope View selected: Zeigt an ob das Attribut für die Scope View Konfiguration ausgewählt wurde.

Functional Constrain:

Name: Normierte Abkürzung des ausgewählten Functional Constrain.

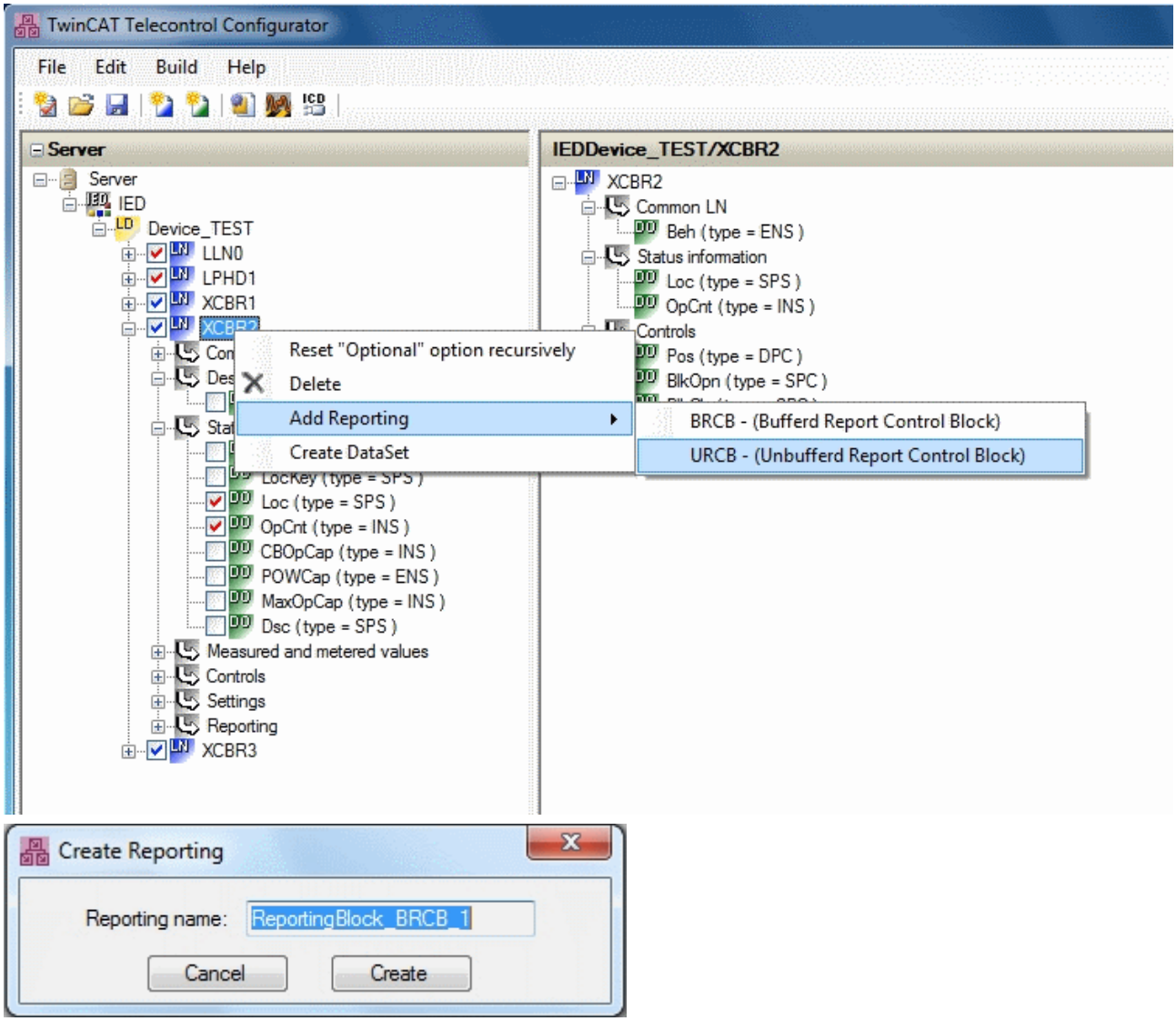
Description: Bedeutung des abgekürzten Functional Constrain.

Bit Mask: Bit Mask welche in der SPS Export-Datei gesetzt wird.

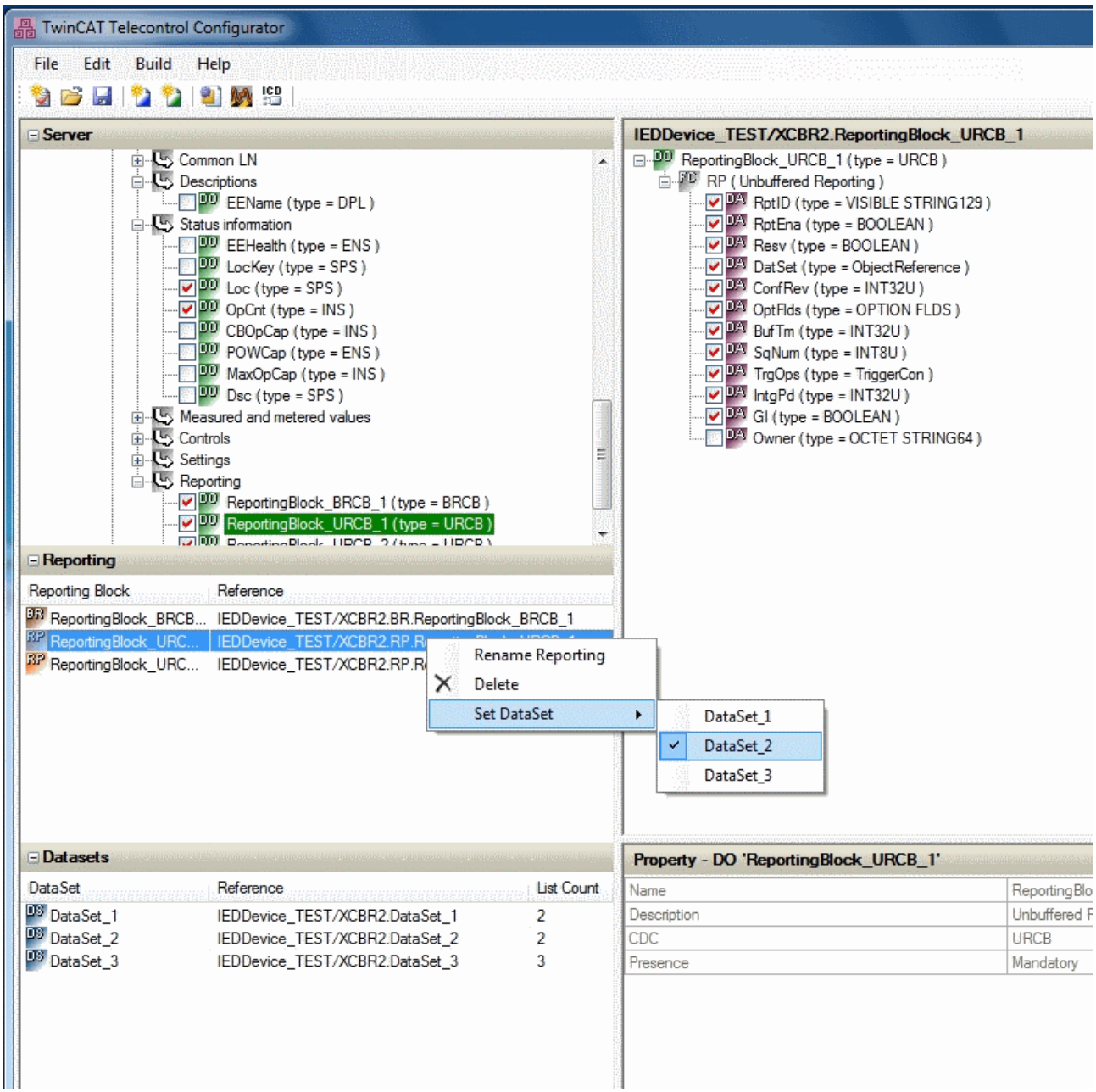
DataSet: Listet die DataSets auf in denen das Function Constrain enthalten ist.

4.4 Reporting

Reportingblöcke können aus dem Kontextmenü heraus erzeugt werden wie im unteren Bild dargestellt. Es besteht die Möglichkeit "Bufferd" oder "Unbufferd" Reportingblöcke zu erzeugen. Da die Reportingblöcke den Logical Nodes zugeordnet werden, ist der entsprechende Kontextmenueintrag nur bei ausgewählten Logical Node verfügbar. Danach öffnet sich ein Dialog in dem Sie den Namen des Reportingblocks angeben können. Nach bestätigen des Dialogs wird der Reportingblock an das Logical Node angehangen und in die Reportingliste gelistet siehe zweites Bild.



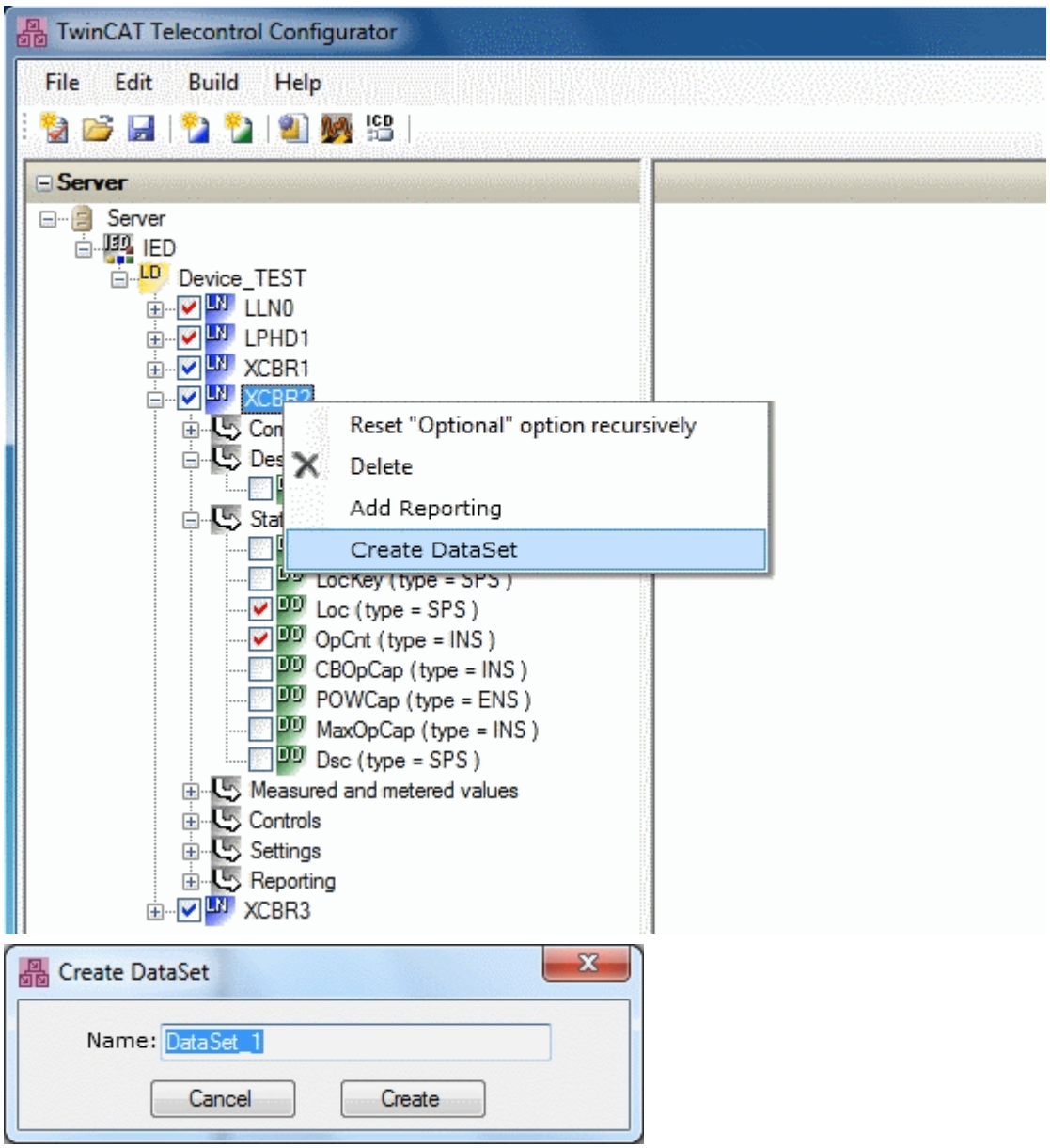
In der Reportingliste besteht die Möglichkeit die Reportingblöcke mit einem DataSet zu verknüpfen. Die Referenz des DataSets wird dann in das Attribut "DatSet" des Reportingblocks geschrieben. Des Weiteren besteht die Möglichkeit den Reportingblock umzubenennen bzw. zu löschen.



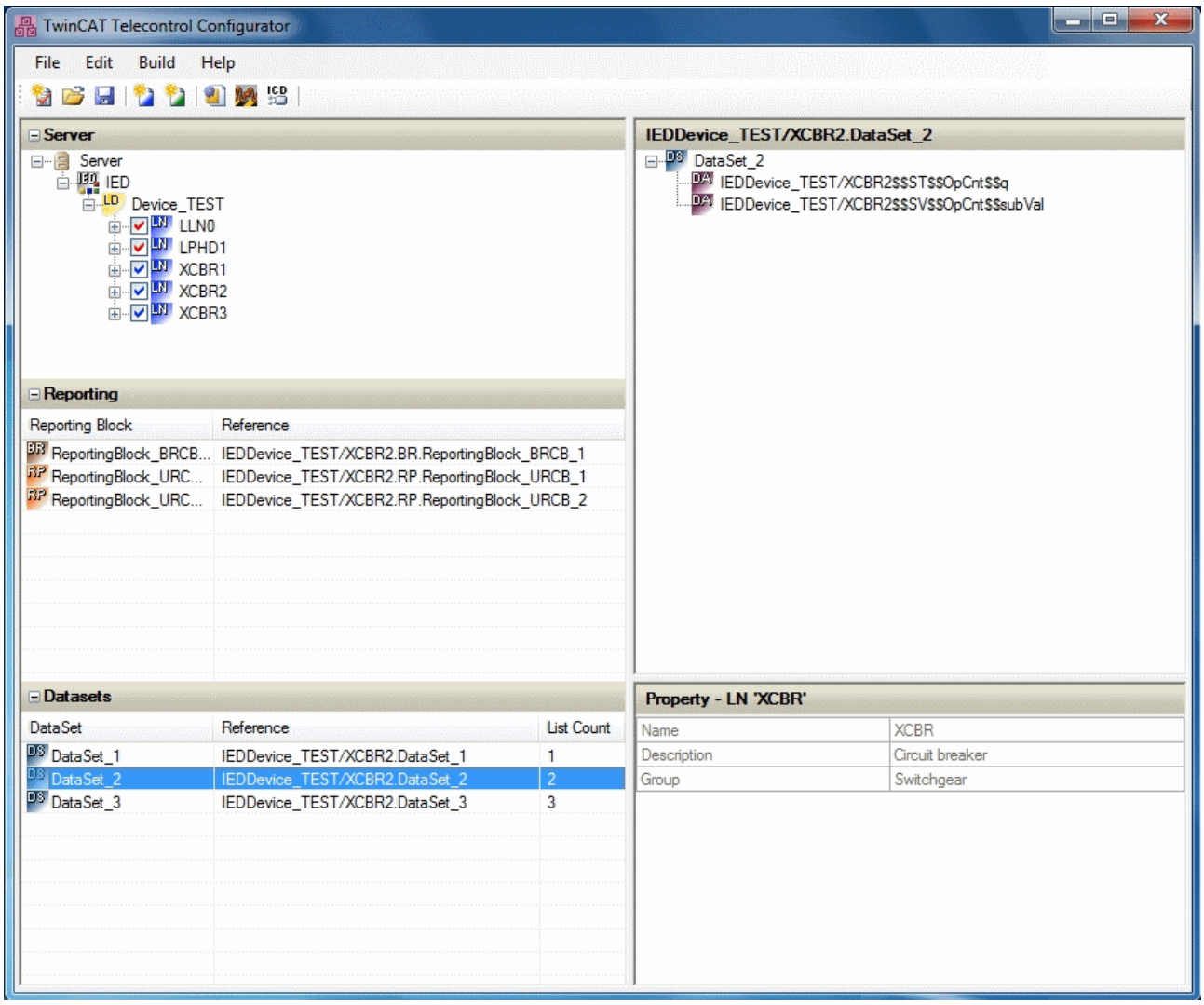
Wird ein Reportingblock aus der Reportingliste ausgewählt werden die DataAttribute des Reportingblocks im rechten Fenster angezeigt.

4.5 DataSets

DataSets können aus dem Kontextmenü heraus erzeugt werden wie im unteren Bild dargestellt. Da die DataSets den Logical Nodes zugeordnet werden, ist der entsprechende Kontextmenüeintrag nur bei ausgewählten Logical Node verfügbar. Danach öffnet sich ein Dialog in dem Sie den Namen des DataSets angeben können. Nach Bestätigen des Dialogs wird das DataSet in der DataSetlist gelistet siehe zweites Bild.



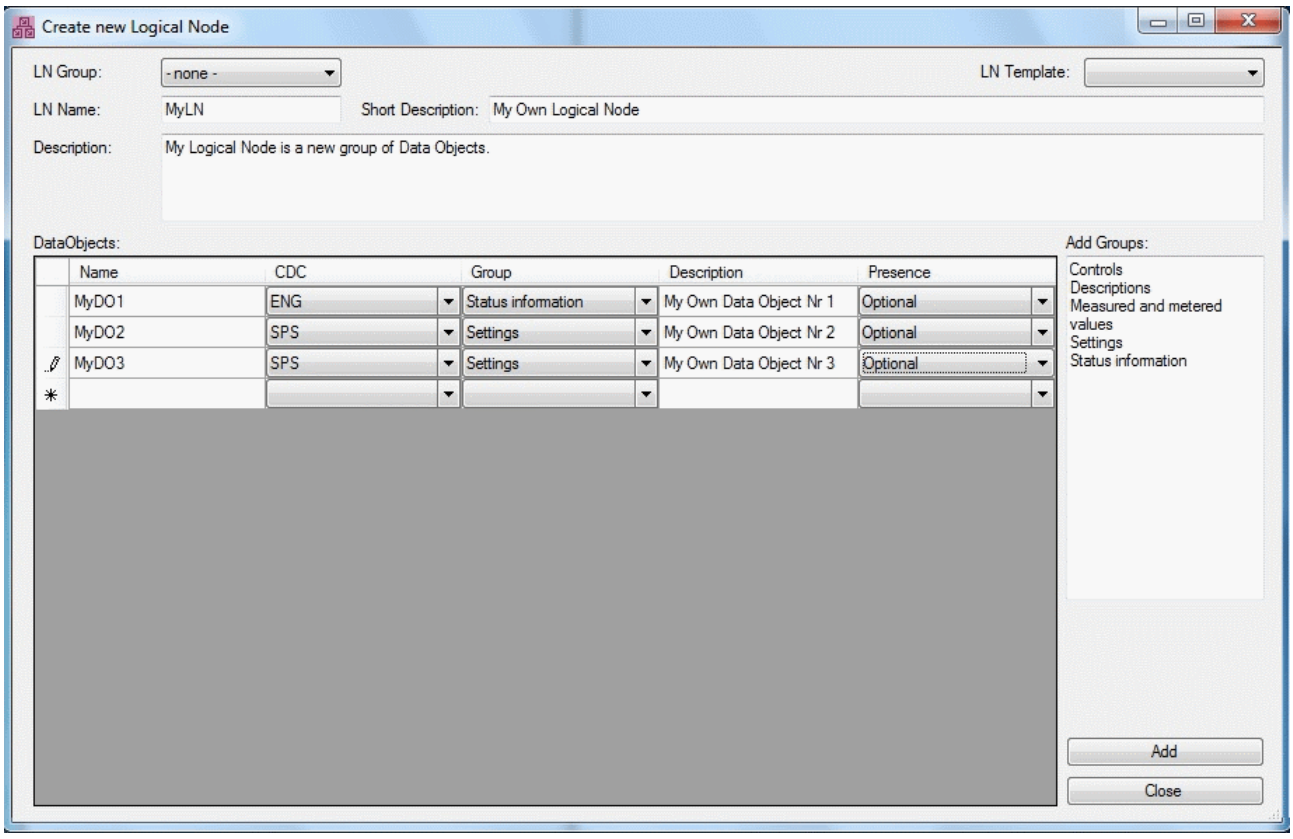
Wählen Sie ein DataSet aus der DataSetlist aus, so werden alle dem DataSet zugehörigen DataAttributes und Functional Constrains im rechten Fenster dargestellt.



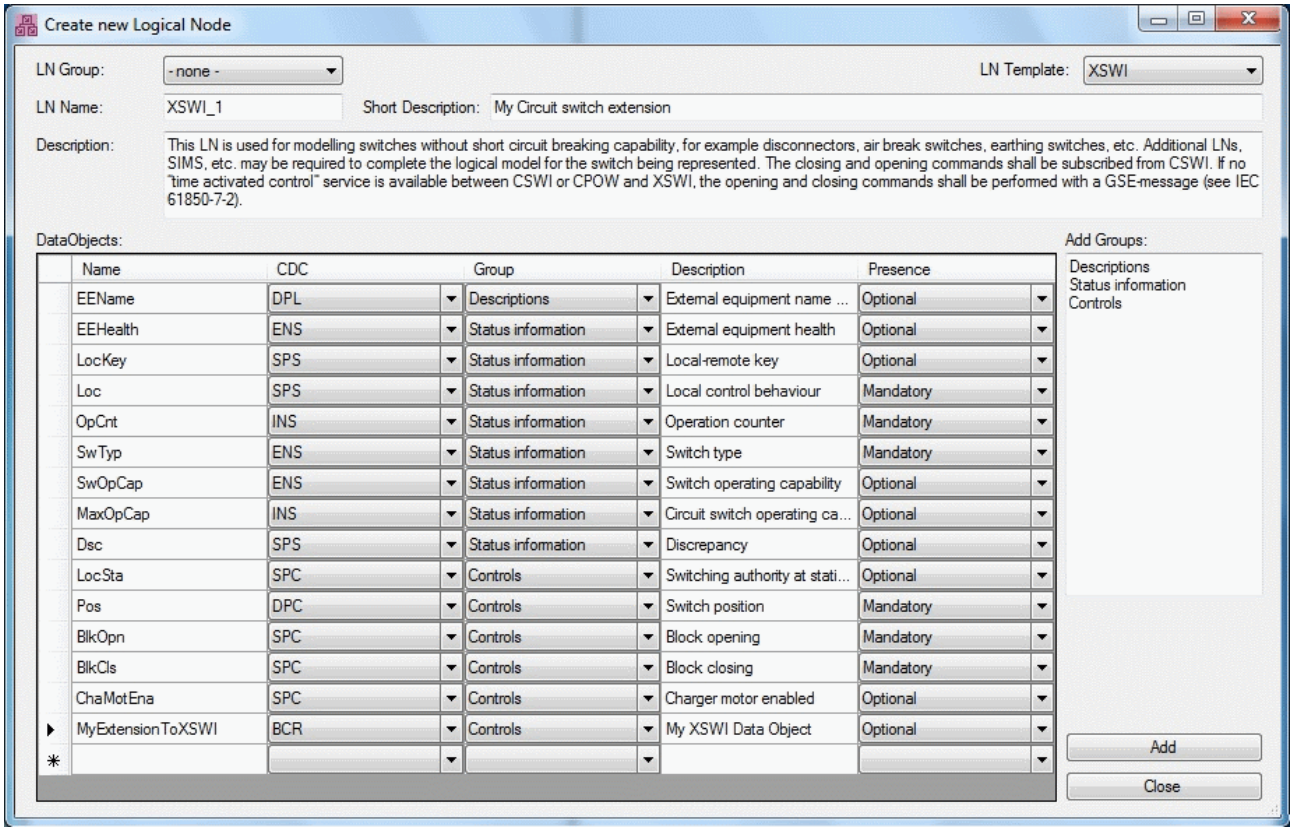
In der DataSetliste werden alle Datasets mit der dazugehörigen Referenz und der Anzahl der angefügten Elemente dargestellt. Aus dem Kontextmenü heraus besteht die Möglichkeit die DataSetname zu ändern bzw. die DataSets zu löschen.

4.6 Private Logical Nodes

Mit der Funktion *New Logical Node* ist es möglich eigene Logical Nodes zu definieren oder bestehende Nodes zu erweitern. Die neuen Elemente werden dabei in eine private Datenbank geschrieben. So stehen diese auch für jede neue Konfiguration wieder zur Verfügung.



In dem gezeigten Fenster können eigene Logical Nodes definiert werden. Dafür können Sie für jedes Data Object eine Common Data Class auswählen. Des Weiteren kann das Objekt einer Gruppe zugeordnet werden, damit eine Kategorisierung im Konfigurator zu einer Steigerung der Übersichtlichkeit führt. In der Spalte Presence ist einstellbar, ob das Element per Default eingeschaltet oder optional sein soll.

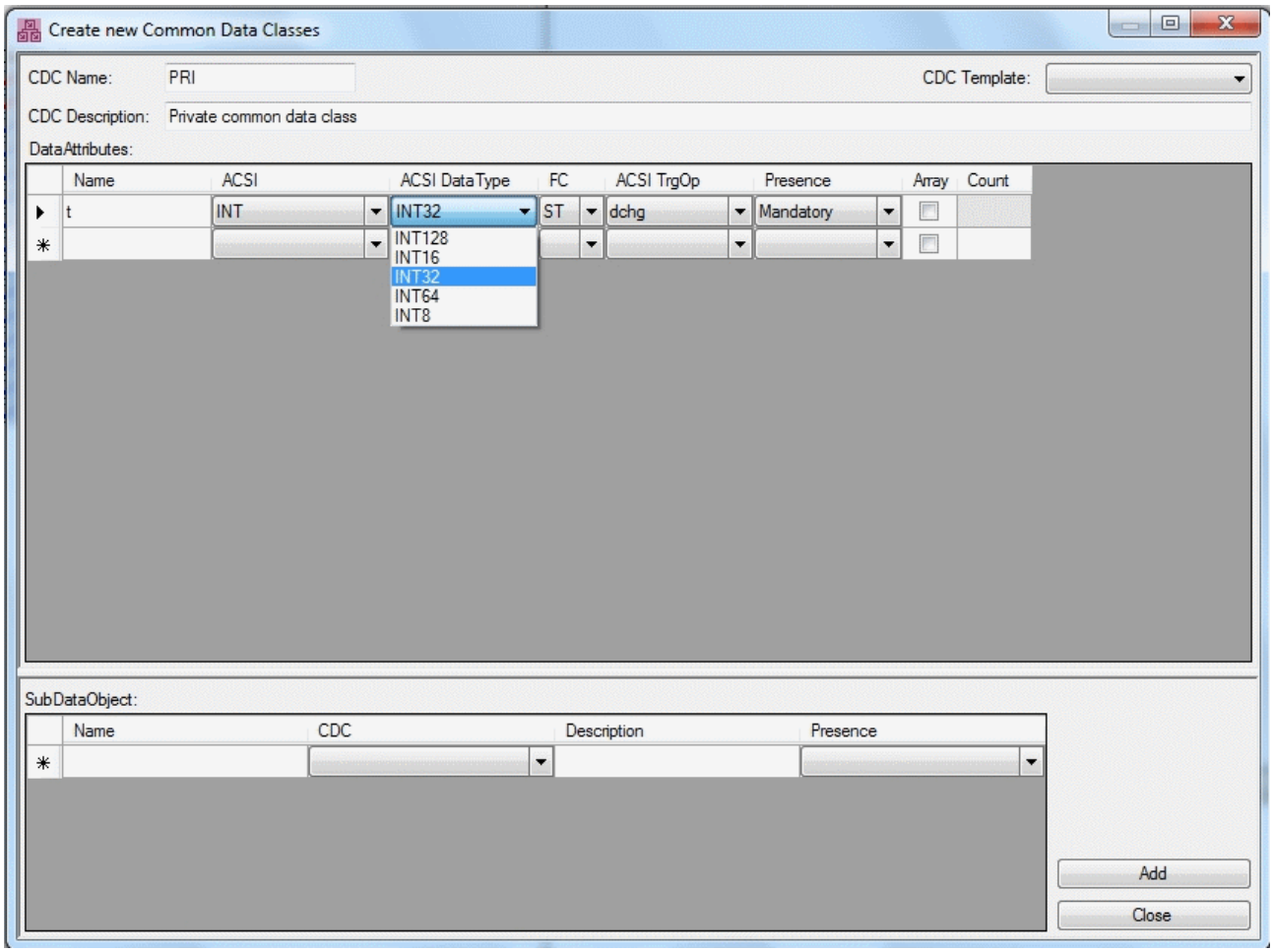


In der hier gezeigten Abbildung wird der im Standard definierte Logical Node *XSWI* (*Circuit Switch*) anwenderspezifisch erweitert. Wählen Sie dazu rechts oben im Fenster ein *Template* für einen Logical Node. Damit es nicht zu Konflikten kommt muss der Name des Logical Nodes anders lauten als der Template-Name.

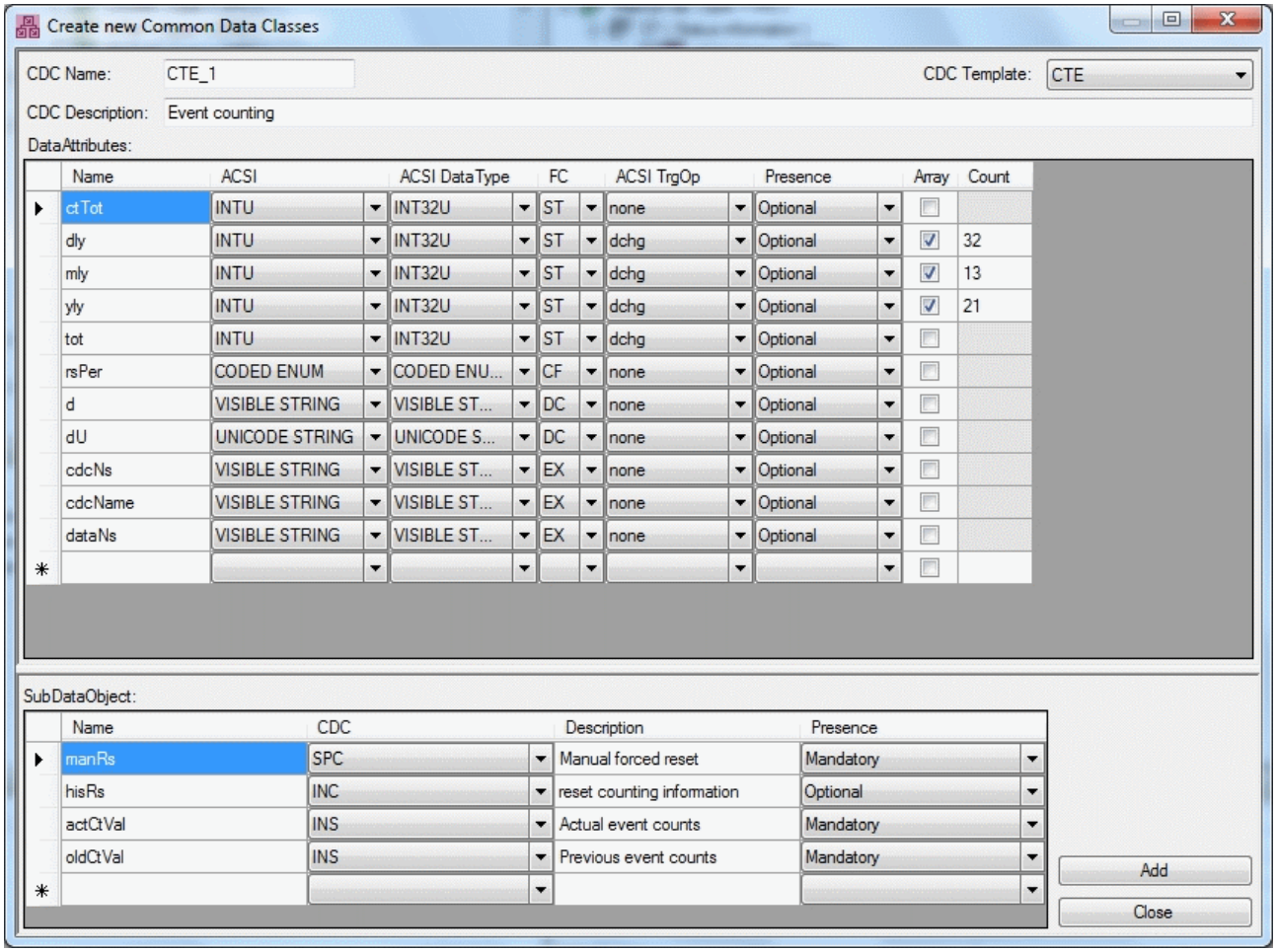
Durch die Bestätigung mit dem *Add* Button wird der neue Logical Node in die private Datenbank geschrieben und steht ab sofort für die Konfiguration zur Verfügung.

4.7 Private Common Data Classes

Mit der Funktion *New Common Data Class* ist es möglich eigene Common Data Classes zu definieren oder bestehende CDC zu erweitern. Die neuen Elemente werden dabei in eine private Datenbank geschrieben. So stehen diese auch für jede neue Konfiguration wieder zur Verfügung.



Wenn eine neue Common Data Class anlegt, wird, können neben den Attributen auch weitere *Sub Data Objects* eingebunden werden. Dazu gibt es unten im Fenster die Möglichkeit eine entsprechende CDC auszuwählen. Für die Attribute können die Datentypen der IEC 61850 ausgewählt werden. Dies könnte zum Beispiel ein Integer sein. Um die Größe des Integers genauer beschreiben zu können, gibt es unter *ACSI Data Types* eine Auswahl für den jeweiligen Datentyp. Auch die Functional Constraints und die Trigger Option können gesetzt werden. In der Spalte Presence kann eingestellt werden, ob das Element per Default eingeschaltet oder optional sein soll. Wenn das Attribut ein Array von x Elementen sein soll, setzen Sie den Haken in der Spalte Array und tragen Sie die Anzahl der Elemente in die Spalte Count ein.

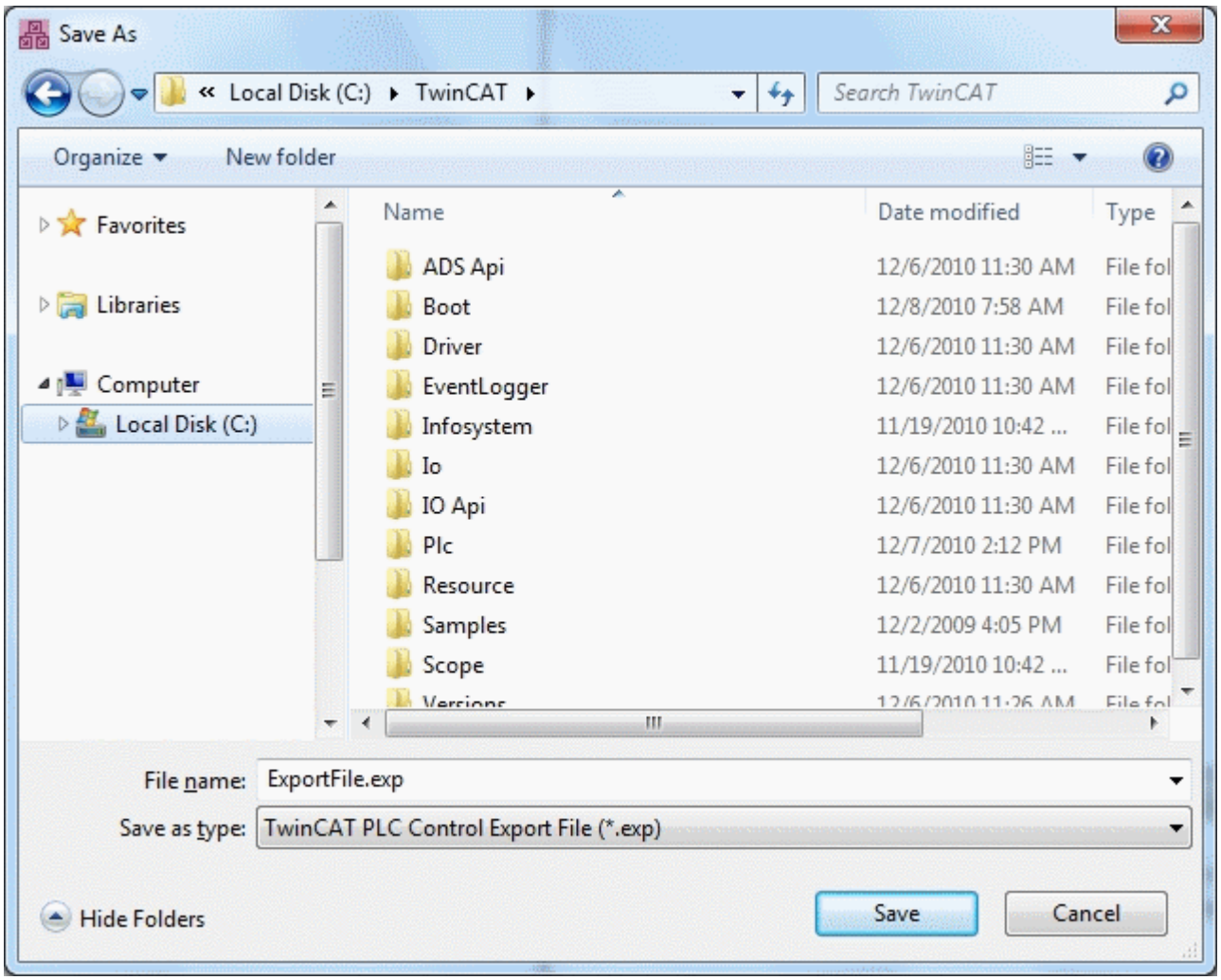


In der hier gezeigten Abbildung wird die im Standard definierte CDC *INC* (*Controllable Integer*) anwenderspezifisch erweitert. Wählen Sie dazu rechts oben im Fenster ein *Template* für eine CDC aus. Damit es nicht zu Konflikten kommt muss der Name der CDC anders lauten als der Template-Name.

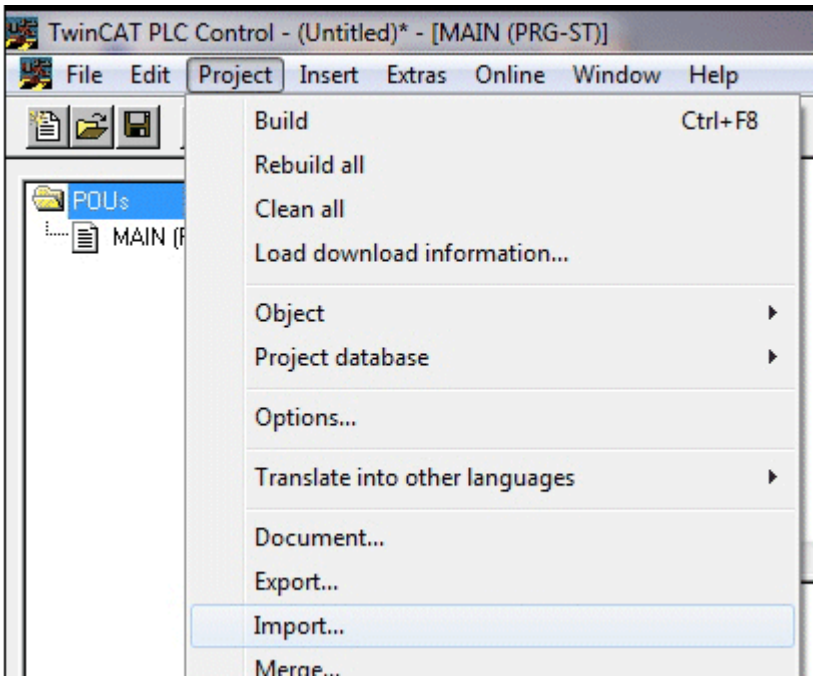
Durch die Bestätigung mit dem *Add* Button wird die neue Common Data Class in die private Datenbank geschrieben und steht ab sofort für die Konfiguration zur Verfügung.

4.8 SPS Exportdatei

Nach Erstellung der Konfiguration für den IEC 61850 Server kann *Create PLC Export File* in der Toolbar betätigt werden, um eine entsprechende Expotdatei zu erzeugen. Der spätere Import in das TwinCAT PLC Control macht das Datenmodell der IEC 61850 / IEC 61400-25 in der SPS nutzbar.

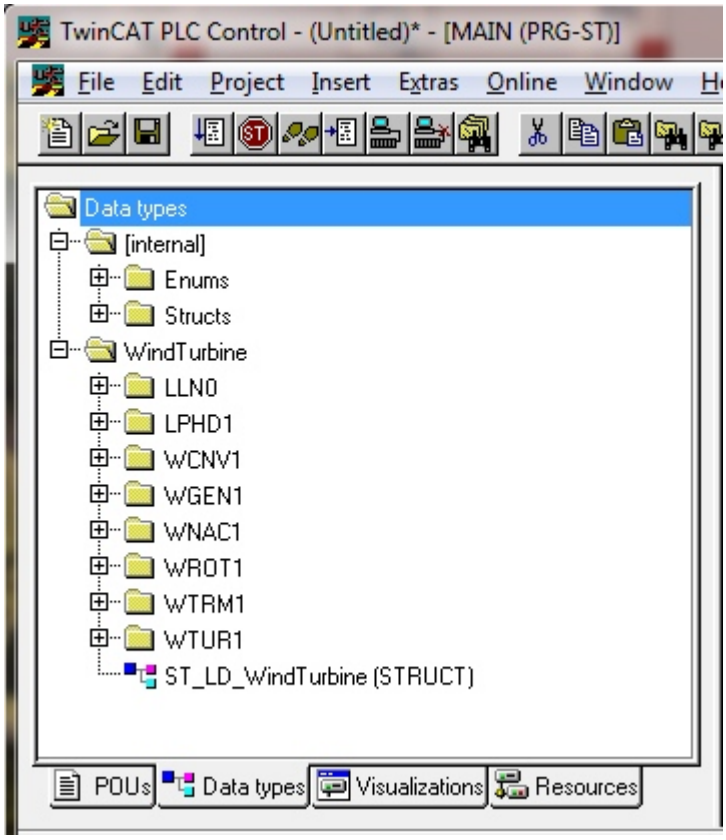


Im *Speichern unter* Dialog kann der Speicherort der Datei ausgewählt werden. Hier haben Sie die Möglichkeit im TwinCAT PLC Control mit der Import-Funktion die Exportdatei einzulesen. Wählen Sie dazu in der Menüleiste *Project* und dann *Import...*, so wie es in der folgenden Abbildung gezeigt wird:

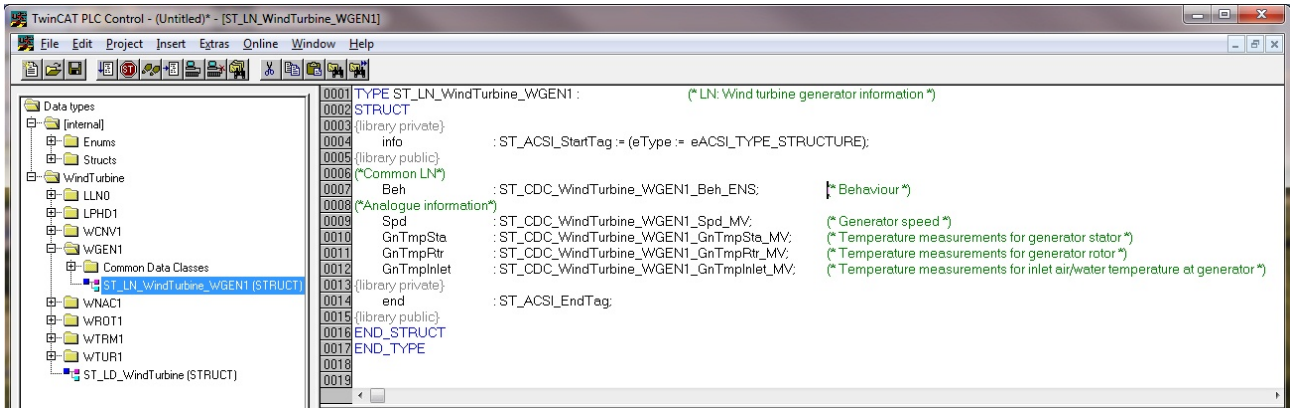


Durch den Import werden im Karteireiter *Data Types* zwei Ordner angelegt. Zum einen *[internal]* und zum anderen ein Ordner, welcher den Namen des im Konfigurator angelegten Logical Devices trägt. In der nächsten Abbildung ist zu sehen, dass dieser Ordner *WindTurbine* heißt. Dieser beinhaltet die tatsächliche

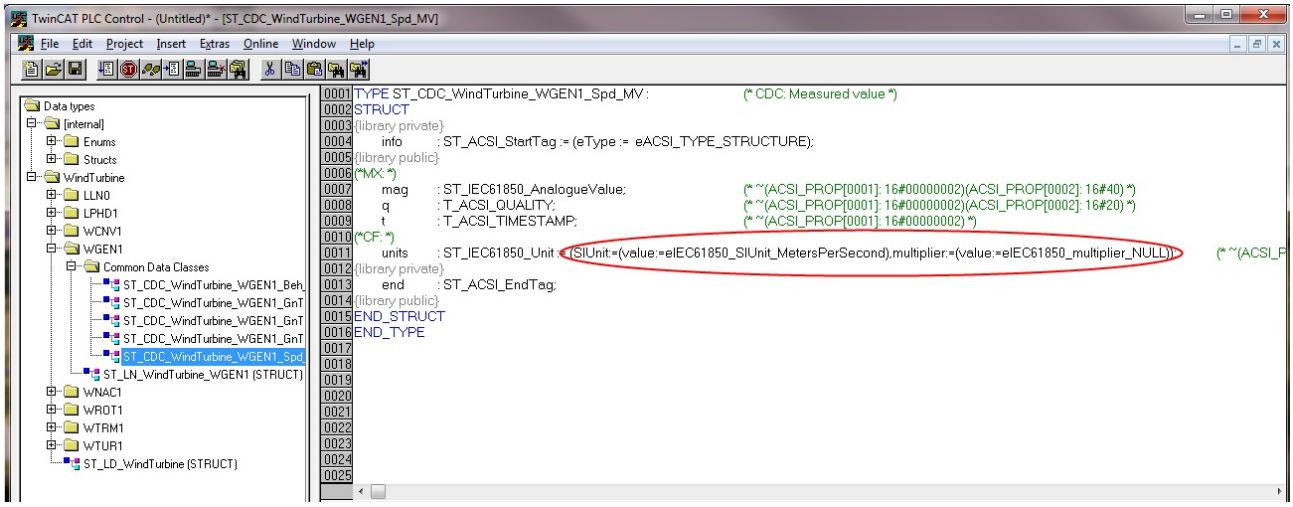
Konfiguration mit allen Logical Nodes und den weiteren unterlagerten Hierarchiestufen. Im Ordner *[internal]* befinden sich lediglich einige strukturierte Datentypen, welche in der Konfiguration verwendet werden. Dieser Ordner ist für den Anwender in der Regel weniger von Bedeutung.



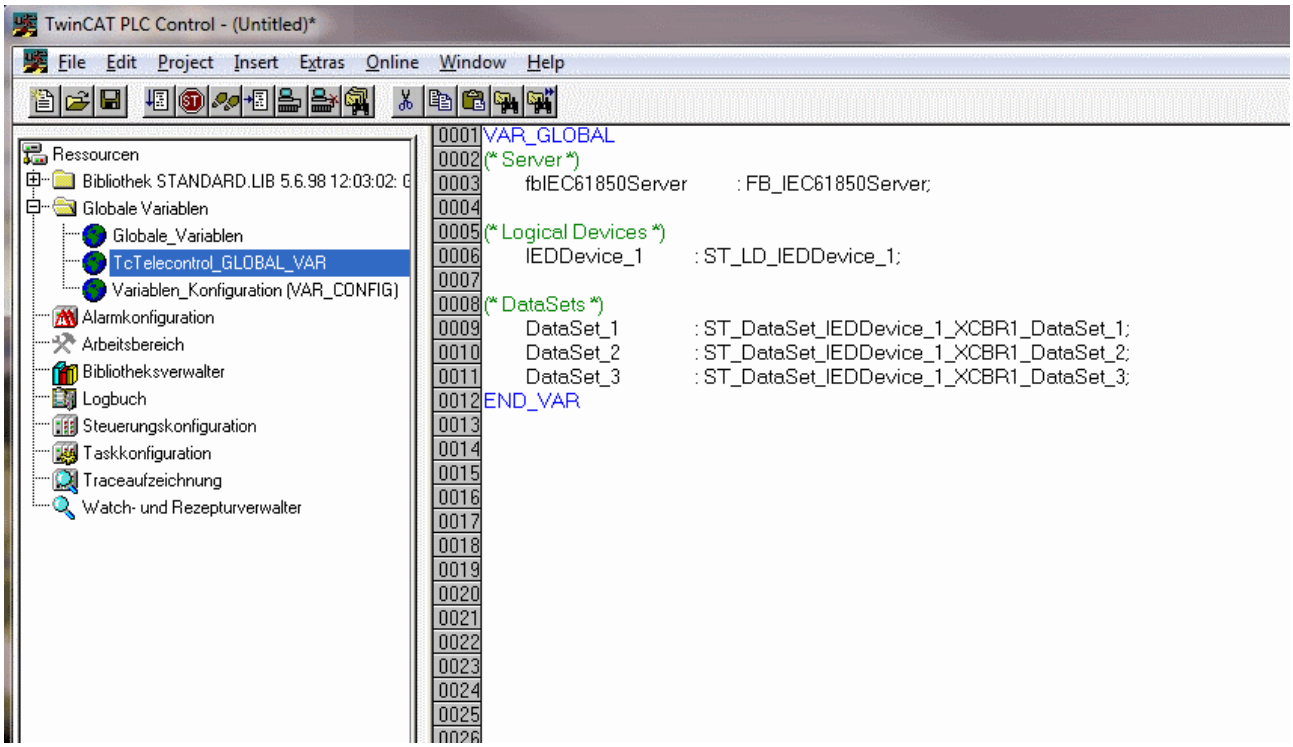
Im Ordner WindTurbine ist eine Struktur *ST_LD_WindTurbine* zu sehen. Diese Struktur muss im SPS Projekt instanziiert und dem FB_IEC6850Server Baustein übergeben werden. Genauere Informationen hierzu finden Sie in den Beispielen [▶ 62] dieser Dokumentation.



Wenn Sie die einzelnen Ordner der Logical Nodes aufklappen, werden die enthaltenen Data Objects in einer weiteren Struktur sichtbar. Die Variablen *info* und *end* dienen der internen Kommunikation. Diese haben für den Anwender nur eine Bedeutung, wenn die vom Konfigurator erstellten Strukturen manuell nachgebaut werden sollen. Sie müssen in jeder Struktur vorhanden sein. Der Ordner *Common Data Classes* enthält die Attribute der unterschiedlichen Objekte.



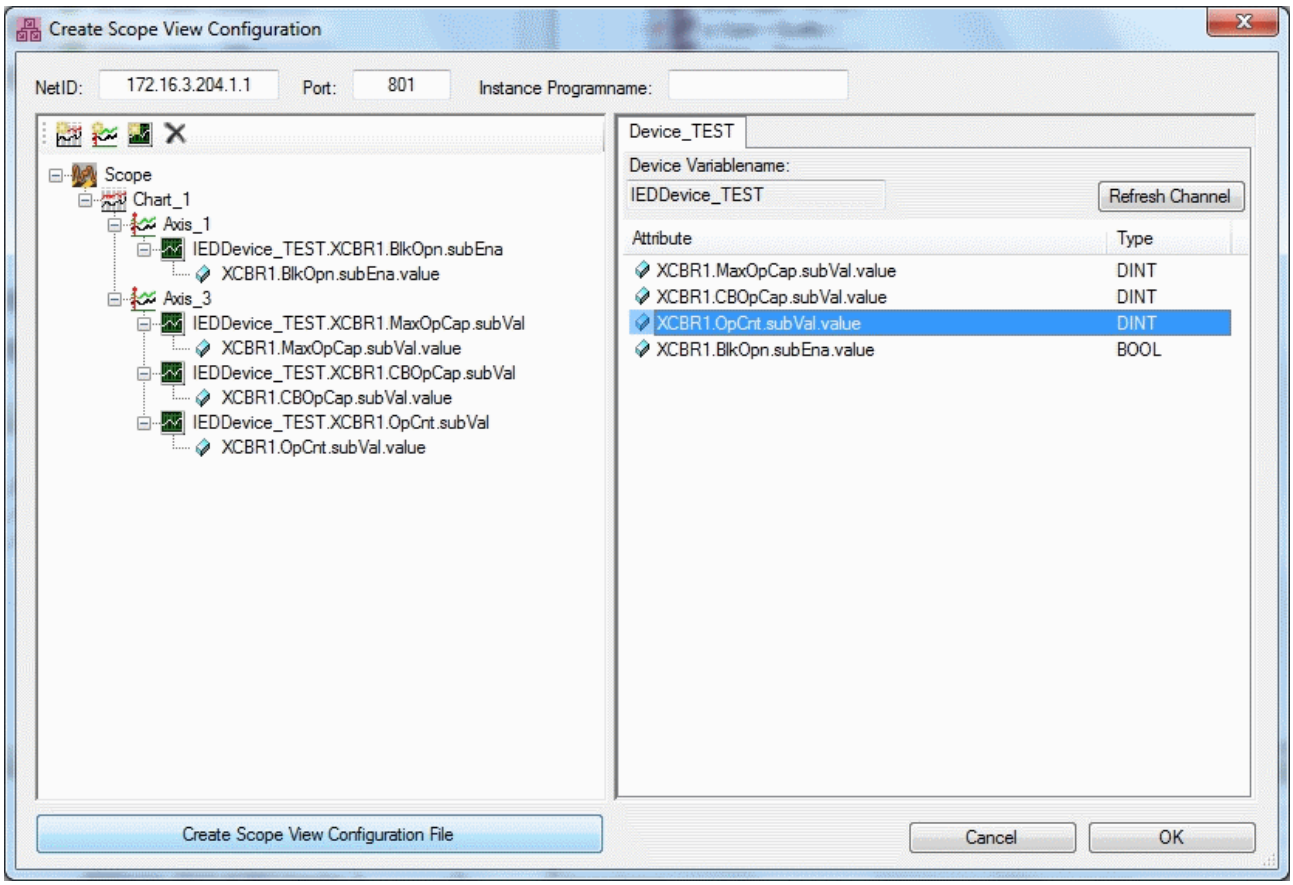
Beispielhaft werden hier die Attribute für das Data Object *Spd* (*Wind Turbine Generator Speed*) gezeigt. Durch die Kommentare werden der Steuerung Informationen wie zum Beispiel zugehöriger Functional Constrain oder Trigger Option weitergegeben. Aber auch die im Konfigurator gesetzten *Default Values* werden hier sichtbar. Hier ist erkennbar, dass für das Attribut *units* die Einheit *Meter pro Sekunde* gesetzt worden ist.



Beim Importieren der Export Datei werden unter anderem auch Globale Variablen erzeugt. Das Objekt "TcTelecontrol_GLOBAL_VAR" wird erzeugt und beinhaltet eine Instanz des FB_IEC61850Server Bausteins, jeweils eine Instanz pro Logical Device und jeweils eine Instanz der erzeugten DataSets.

4.9 Scope View Konfigurationsdatei

Nach Erstellung der Konfiguration für den IEC 61850 Server kann *Create Scope View Data File* in der Toolbar betätigt werden, um eine Scope View Konfigurationsdatei zu erzeugen mit allen ausgewählten Attributen. Die Zuordnung der einzelnen Elemente zu den verschiedenen Kanälen erfolgt über den folgenden Dialog.



Auf der linke Seite ist der Scope View Konfigurationsbaum mit den bekannten Scope View Elemenen "Scope", "Chart", "Axis" und "Channel".

Auf der rechten Seite finden sich alle ausgewählten Attribute der aus der Telecontrol configuration wieder.

Um die Attribute mit einem Kanal zu verbinden, muss nur das Attribute per Drag and Drop zum entsprechenden Kanal gezogen werden. Das Attribut wird automatisch an den Kanal angehangen und der Kanalname wird dem entsprechenden Attribut angepasst. Der Kanalname kann allerdings auch von Hand nach eigenen Vorstellungen geändert werden.

Um die Scope View Konfigurationsdatei zu erzeugen, muss der Button "Create Scope View Configuration File" betätigt werden. Im *Speichern unter* Dialog kann nun der Speicherort der Datei ausgewählt werden.

5 SPS-API

Die wichtigsten Funktionsbausteine, Funktionen, Datentypen und Konstanten des TwinCAT IEC 61850 Server sind in diesem Teil der Dokumentation beschrieben. Um diese Elemente nutzen zu können, muss die TcIEC61850Server.lib in das gewünschte TwinCAT PLC Control Projekt eingebunden werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcIEC61850Server.Lib

5.1 TcIEC61850Server

5.1.1 Funktionsbausteine

5.1.1.1 FB_IEC61850Server

Mit diesem Funktionsbaustein kann in der TwinCAT SPS ein Server nach der IEC 61850 Norm realisiert werden. Über die Eingangsvariablen appPara und srvPara werden die Verbindungsparameter wie IP-Adresse und die applikationsspezifischen Parameter eingestellt.

Der Server-Baustein besitzt zwei Aktionen:

- A_INIT; Beim Aufruf dieser Aktion wechselt der Server in den Zustand INITIALIZING. In dieser Phase wird die interne Datenbank initialisiert (die Instanzen der gemeinsamen Datenklassen, Attribute, logische Knoten usw. werden erzeugt). Nach der Initialisierung der internen Datenbank befindet sich der Server im PREOP-Zustand. In diesem Zustand können weitere applikationsabhängige Datenklassen in der Datenbank generiert werden z.B. Datasets.
- A_OPERATE; Durch den Aufruf dieser Aktion wechselt der Server zum Zustand: OPERATIONAL. In diesem Zustand kann der Server eine Verbindung zum Client aufbauen und aktiv Daten austauschen.

VAR_INPUT

```
VAR_INPUT
    appPara     : ST_MMS_ApplicationPara;    (* Application parameter *)
    srvPara     : ST_MMS_ProtocolPara;  (* Communication parameter *)
    pLD         : DWORD := 0;          (* Pointer to logical device node *)
    cbLD        : UDINT := 0;          (* Byte size of logical device node *)
END_VAR
```

appPara: Applikationsspezifische Konfigurationsparameter [▶ 47] (MMS- und IEC 61850-Parameter).

srvPara: Parameter [▶ 47] die hauptsächlich die Konfiguration des Kommunikationsstacks unterhalb von MMS betreffen.

pLD: Pointer (Adresse) auf die Instanz der Struktur mit der Modellbeschreibung des logischen Gerätes (LD). Die Adresse kann mit dem ADR-Operator ermittelt werden.

cbLD: Bytegröße der Struktur mit der Modellbeschreibung des logischen Gerätes (LD). Die Bytegröße kann mit dem SIZEOF-Operator ermittelt werden.

VAR_OUTPUT

```
VAR_OUTPUT
    system : ST_IED_System;
    eState : E_IED_STATE;
END_VAR
```

system: System [▶ 46]-Variable.

eState : Gerätestatus [▶ 45].

Beispiel:

Weitere Beispiele inklusive aller Sourcen finden Sie unter: Beispiele [▶ 62].

```
PROGRAM MAIN
VAR
    bInit   : BOOL := TRUE;
    bOperate    : BOOL := TRUE;
    bServer     : FB_IEC61850Server := (    srvPara := (    sHost := '127.0.0.1',(* change the IP-
Address to meet your needs *)
                            debug := ( bError := TRUE, bLog := TRUE ) ) );(* Server instance *)

    fbChange : FB_ACSI_ChangePresence;(* This function block changes the mandatory/
optional presence parameter *)
 Relay   : ST_LD_Relay;(* Our logical device *)
END_VAR
```

```
IF bInit THEN
    bInit := FALSE;
    fbServer.appPara.bAccessCtrl := FALSE; (* TRUE = Use access control/
authentication, FALSE = Don't use access control/authentication *)
 (* first user *)
    fbServer.appPara.accessCtrl[1].bAuthentication := TRUE;(* TRUE = access control/
authentication for this user enabled, FALSE = Disabled *)
 fbServer.appPara.accessCtrl[1].authentMechanism := '2.2.3.1';
    fbServer.appPara.accessCtrl[1].authentPassword := 'PASSWORD';
    (* second user *)
    fbServer.appPara.accessCtrl[2].bAuthentication := TRUE;
    fbServer.appPara.accessCtrl[2].authentMechanism := '2.2.3.1';
    fbServer.appPara.accessCtrl[2].authentPassword := 'OTHER';

  (* at this point you can change the presence of some attributes... *)

    (* change presence of some attributes *)
(*      fbChange( pDO := ADR( Relay.LLN0.Loc.subEna ), cbDO := SIZEOF(  Relay.LLN0.Loc.subEna ), bPr
esence := TRUE );*)(* change to mandatory *)
(*      fbChange( pDO := ADR( Relay.LLN0.Loc.t ), cbDO := SIZEOF(  Relay.LLN0.Loc.t ), bPresence :=
FALSE );*)(* change to optional *)

    fbServer.A_INIT( pLD := ADR( Relay ), cbLD := SIZEOF( Relay ) );
END_IF

IF bOperate THEN(* switch to oparating state *)
    IF fbServer.eState = eIED_STATE_PREOP THEN
        bOperate := FALSE;

        (* at this point you can configure your data sets...*)

        fbServer.A_OPERATE( pLD := ADR( Relay ), cbLD := SIZEOF( Relay ) );
    END_IF
END_IF

fbServer( pLD   := ADR( Relay ), cbLD := SIZEOF( Relay ) );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC oder CX (X86, ARM)	TcIEC61850Server.Lib

5.1.2 Funktionen

5.1.2.1 F_GetVersionTcIEC61850Server

Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcIEC61850Server: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC oder CX (x86, ARM)	TcIEC61850Server.Lib

5.1.3 Datentypen

5.1.3.1 E_IED_STATE

Betriebszustand des Servers.

```
TYPE E_IED_STATE :
(
    eIED_STATE_IDLE := 0,
    eIED_STATE_INITIALIZING,
    eIED_STATE_PREOP,
    eIED_STATE_OPERATIONAL,
    eIED_STATE_STOPPING
);
END_TYPE
```

Der Server befindet sich im Zustand IDLE, nach dem der Server-Code auf das Zielgerät herunterladen wurde und keine der Server-Aktionen (A_INIT oder A_OPERATE) aufgerufen wurde.
Beim Aufruf der Aktion A_INIT wechselt der Server in den Zustand INITIALIZING. In dieser Phase wird die interne Datenbank initialisiert (die Instanzen der gemeinsamen Datenklassen, Attribute, logische Knoten usw. werden erzeugt). Danach wechselt der Server zu dem PREOP-Zustand. In diesem Zustand können weitere applikationsabhängige Datenklassen in der Datenbank generiert werden z.B. Datasets. Wenn alle Datasets konfiguriert wurden, kann durch den Aufruf der Aktion A_OPERATE zum weiteren Zustand OPERATIONAL gewechselt werden. In diesem Zustand kann der Server eine Verbindung zum Client aufbauen und aktiv Daten austauschen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcIEC61850Server.Lib

5.1.3.2 ST_IED_System

```
TYPE ST_IED_System :
STRUCT
    vmd : ST_MMS_VMD;
END_STRUCT
END_TYPE
```

vmd: Diese Struktur [▶ 50] repräsentiert das VMD-Object (Virtual Manufacturing Device).

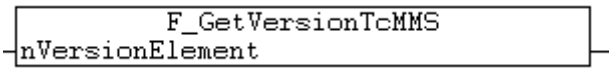
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcIEC61850Server.Lib

5.2 TcMMS

5.2.1 Funktionen

5.2.1.1 F_GetVersionTcMMS



Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcMMS : UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC oder CX (x86, ARM)	TcMMS.Lib

5.2.2 Datentypen

5.2.2.1 E_MMS_Environment

```
TYPE E_MMS_Environment:
(
    eMMS_Environment_None := 0,
    eMMS_Environment_Establishing,
    eMMS_Environment_Running,
    eMMS_Environment_Relinquishing
);
END_TYPE
```

Diese Variable liefert den Status der MMS-Umgebung.

eMMS_Environment_None: MMS-Umgebung befindet sich im inaktiven Status.

eMMS_Environment_Establishing: MMS-Umgebung wird gerade hergestellt.

eMMS_Environment_Running: MMS-Umgebung ist aktiv (MMS-Dienste können ausgeführt werden).

eMMS_Environment_Relinquishing: MMS-Umgebung wird abgebaut.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcMMS.Lib

5.2.2.2 ST_MMS_ApplicationPara

```
TYPE ST_MMS_ApplicationPara :
STRUCT
  (* Default mms settings *)
    maxPDUsize          : UDINT := 65000;
    localDetail         : DINT  := 65000;
    maxServOutCalling   : INT := 10;
    maxServOutCalled    : INT := 10;

    services            : ST_MMS_ServiceSupportOptions;(* Default supported services *)
    parameters          : ST_MMS_ParameterSupportOptions;(* Default parameter support CBB *)
    nestingLevel        : SINT := 5;(* Default nesting level *)
  (* authentication / access control *)
    bAccessCtrl         : BOOL := FALSE;(* Access control: TRUE = enabled, FALSE = disabled *)
    accessCtrl          : ARRAY[1..MAX_MMS_ACCESS_CONTROL_USERS] OF ST_MMS_AccessControl;
END_STRUCT
END_TYPE
```

maxPDUsize: Maximale Bytelänge der PDU (protocol data unit)

localDetail: Local detail called.

maxServOutCalling: Maximal zugelassene Anzahl unbestätigter MMS-Dienste beim aufrufenden System.

maxServOutCalled: Maximal zugelassene Anzahl unbestätigter MMS-Dienste beim aufgerufenen System.

services: Unterstützte MMS-Dienste [▶ 49].

parameters: Unterstützte CBB-Parameter [▶ 50].

nestingLevel: Maximale Verschachtelungstiefe der MMS-Objektdaten.

bAccessCtrl: Aktiviert/deaktiviert die Authentifizierung über ein Passwort. Ist diese Variable gesetzt, dann werden beim Verbindungsaufbau die Konfigurationseinstellungen in der Arrayvariablen *accessCtrl* überprüft.

accessCtrl: Benutzerzugriff [▶ 49] und -Konfigurationseinstellungen. Jedes Arrayelement entspricht einem Benutzer, somit können bis zu MAX_MMS_ACCESS_CONTROL_USERS [▶ 53] konfiguriert werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcMMS.Lib

5.2.2.3 ST_MMS_ProtocolPara

MMS-Kommunikationsparameter.

```
TYPE ST_MMS_ProtocolPara:
STRUCT
    sSrvNetID           : T_AmsNetID := ''; (* TwinCAT TCP/
IP Connection Server network address (default = empty string) *)
    bClient             : BOOL := FALSE;
(* TRUE => act as client, FALSE => act as server (reserved for future use) *)
```

```
    sHost              : T_IPv4Addr := '127.0.0.1'; (* If client => remote address, if server => loc
al address. String containing an (Ipv4) internet protocol dotted address. *)
    nPort              : UDINT := 102; (* If client => remote internet protocol (IP) port, if server
 => local listener port (default 102) *)
    nMode              : DWORD := 16#80000001;
(* Additional (optional) parameters e.g. bit 31 = CONNECT_MODE_ENABLEDBG, bit 0 = LISTEN_MODE_CLOSEA
LL *)
    eAccept            : E_SocketAcceptMode := eACCEPT_ALL; (* Connection accept flags (server only)
 *)
    local_AP_title     : STRING := '1.1.1.999.1';
    local_AE_qualifier : UDINT := 12;
    local_T_selector   : STRING := '00 01';(* local-transport-selector *)
    local_S_selector   : STRING := '00 01';(* local-session-selector *)
    local_P_selector   : STRING := '00 00 00 01';(* local-presentation-selector *)

    remote_AP_title    : STRING := '1.1.1.999';
    remote_AE_qualifier: UDINT := 12;
    remote_T_selector  : STRING := '00 01';(* remote-transport-selector *)
    remote_S_selector  : STRING := '00 01';(* remote-session-selector *)
    remote_P_selector  : STRING := '00 00 00 01';(* remote-presentation-selector *)


    eTPDUsize          : E_COTP_DUsize := eCOTP_DUsize_1024;

    debug              : ST_MMS_DebugPara;
END_STRUCT
END_TYPE
```

sSrvNetID: Die Netzwerkadresse des TwinCAT TCP/IP Connection Servers (Default = Leerstring);

bClient: Wenn diese Variable gesetzt ist, agiert der Server als Client. Reserviert und zurzeit nicht benutzt (Default = FALSE);

sHost: Lokale IPv4 Server-Hostadresse als String (Default = '127.0.0.1' = Loopback address);

nPort: Lokale Server-Portadresse (Default = 102).

nMode: Zusätzliche (optionale) Parameter (Default = CONNECT_MODE_ENABLEDBG OR LISTEN_MODE_CLOSEALL):

- Bit 31: Aktiviert/deaktiviert Debugmeldungen beim Aufbauen/Abbauen der TCP/IP-Verbindungen (das Bit kann über die Konstante CONNECT_MODE_ENABLEDBG gesetzt/maskiert werden);
- Bit 0: Aktiviert/deaktiviert das automatische Schließen aller alten Verbindungen beim Programmstart (das Bit kann über die Konstante LISTEN_MODE_CLOSEALL gesetzt/maskiert werden);
- Alle anderen Bits sind reserviert;

eAccept: Reserviert und zurzeit nicht benutzt. Dieser Parameter bestimmt welche eingehenden Verbindungen vom Server akzeptiert oder abgelehnt werden. (Default = eACCEPT_ALL = alle Verbindungen werden akzeptiert);

local_AP_title: Lokaler ACSE AP-Title als String (Default = '1,1,1,999,1');

local_AE_qualifier: Lokaler ACSE AE-Qualifier (Default = 12);

local_T_selector: Lokale TSAP-Adresse als String (T-SAP = Transport Service Access Point, Default = '00 01');

local_S_selector: Lokale SSAP-Adresse als String (S-SAP = Session Service Access Point, Default = '00 01');

local_P_selector: Lokale PSAP-Adresse als String (P-SAP = Presentation Service Access Point, Default = '00 00 00 01');

remote_AP_title: Remote ACSE AP-Title als String (default = '1,1,1,999');

remote_AE_qualifier: Remote ACSE AE-Qualifier (Default = 12);

remote_T_selector: Remote TSAP-Adresse als String (T-SAP = Transport Service Access Point, Default = '00 01');

remote_S_selector: Remote SSAP-Adresse als String (S-SAP = Session Service Access Point, Default = '00 01');

remote_P_selector: Lokale PSAP-Adresse als String (P-SAP = Presentation Service Access Point, Default = '00 00 00 01');

eTPDUSize: Maximale Länge [▶ 52] der segmentierten Transport Protocol Dateneinheit in Byte (Default = eCOTP_DUsize_1024 = 1024 Byte);

debug: Diese Struktur [▶ 52] konfiguriert die Debugausgaben ins TwinCAT System Manager Logview. Die Membervariablen können während der Inbetriebnahme/Fehlersuche benutzt werden, um Debugausgaben zu aktivieren bzw. zu deaktivieren.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcMMS.Lib

5.2.2.4 ST_MMS_AccessControl

```
TYPE ST_MMS_AccessControl :
STRUCT
  (* authentication *)
    bAuthentication     : BOOL := TRUE;(* Authentication: TRUE = enabled, FALSE = disabled *)
    authentMechanism    : STRING := '2.2.3.1';(* Authentication mechanism: (joint-iso-
ccitt.2.3.1) = 52 03 01 hex *)
    authentPassword     : STRING(MAX_MMS_PASSWORD_LENGTH) := 'PASSWORD';
(* Authentication password *)

    services            : ST_MMS_ServiceSupportOptions;(* Supported services *)
    parameters          : ST_MMS_ParameterSupportOptions;(* Parameter support CBB *)
    nestingLevel        : SINT := 5;(* Nesting level *)
END_STRUCT
END_TYPE
```

bAuthentication: Ist dieser Parameter auf TRUE gesetzt, dann sind die Authentifizierungseinstellungen für diesen Benutzer aktiviert. Über diesen Parameter können Sie den Zugriff für bestimmte Benutzer kurzzeitig deaktivieren.

authentMechanism: Authentifizierungs-Mechanismus (default = "2.2.3.1").

authentPassword: Authentifizierungs-Passwort [▶ 53] (default = "PASSWORD").

services: Unterstützte MMS-Dienste [▶ 49].

parameters: Unterstützte CBB-Parameter [▶ 50].

nestingLevel: Max. zulässige Verschachtelungstiefe der MMS-Variablen (default = 5).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcMMS.Lib

5.2.2.5 ST_MMS_ServiceSupportOptions

```
TYPE ST_MMS_ServiceSupportOptions:
STRUCT
    status                          : BOOL := TRUE;
    getNameList                     : BOOL := TRUE;
    identify                        : BOOL := TRUE;
    read                            : BOOL := TRUE;
    write                           : BOOL := TRUE;
    getVariableAccessAttributes     : BOOL := TRUE;
    defineNamedVariableList         : BOOL := TRUE;
    getNamedVariableListAttributes  : BOOL := TRUE;
    deleteNamedVariableList         : BOOL := TRUE;
    getDomainAttributes             : BOOL := TRUE;
    kill                            : BOOL := FALSE;
    readJournal                     : BOOL := TRUE;
```

```
    writeJournal                    : BOOL := TRUE;
    initializeJournal               : BOOL := TRUE;
    reportJournalStatus             : BOOL := TRUE;
    getCapabilityList               : BOOL := TRUE;
    FILEOPEN                        : BOOL := TRUE;
    FILEREAD                        : BOOL := TRUE;
    FILECLOSE                       : BOOL := TRUE;
    fileRename                      : BOOL := FALSE;
    fileDelete                      : BOOL := TRUE;
    fileDirectory                   : BOOL := TRUE;
    unsolicitedStatus               : BOOL := TRUE;
    informationReport               : BOOL := TRUE;
    conclude                        : BOOL := TRUE;
    cancel                          : BOOL := TRUE;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcMMS.Lib

5.2.2.6 ST_MMS_ParameterSupportOptions

Parameter die beim Verbindungsaufbau (Association) zwischen den zwei Kommunikationspartnern ausgehandelt werden.

```
TYPE ST_MMS_ParameterSupportOptions:
STRUCT
    str1 : BOOL := TRUE;
    str2 : BOOL := TRUE;
    vnam : BOOL := TRUE;
    valt : BOOL := TRUE;
    vlis : BOOL := TRUE;
END_STRUCT
END_TYPE
```

str1: Wenn gesetzt, dann unterstützt das Gerät den Array-Type-Bezeichner für die Typbeschreibung der Array-Variablen.

str2: Wenn gesetzt, dann unterstützt das Gerät den Structure-Type-Bezeichner für die Typbeschreibung der Strukturierten Variablen.

vnam: Wenn gesetzt, dann unterstützt das Gerät den named-Zugriff in variableSpecification.

valt: Wenn gesetzt, dann unterstützt das Gerät den sogenannten AlternateAccess. Indizierter Zugriff auf Array-Elemente (INDEX oder INDEX-RANGE).

vlis: Wenn gesetzt, dann unterstützt das Gerät de variableListName in variableAccessSpecification.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcMMS.Lib

5.2.2.7 ST_MMS_VMD

```
TYPE ST_MMS_VMD :
STRUCT
    vendorName      : T_MaxString := 'BECKHOFF';
    modelName       : T_MaxString := 'TwinCAT IEC 61850 Server';
    revision        : T_MaxString := '1.0.0';

    Associations    : ARRAY[1..MAX_MMS_APPLICATION_ASSOCIATIONS] OF ST_MMS_ApplicationAssociation;

    errors          : FB_MMS_ErrorFifo;
```

```
    nAllocLoad      : UDINT := 0;
END_STRUCT
END_TYPE
```

vendorName: VMD-Herstellername.

modelName: VMD-Modelbezeichnung.

revision: VMD-Versionsnummer als String.

Associations: Eine Liste der Application-Associations [▶ 53]. Eine Application-Association [▶ 51] stellt eine Client-Server-Verbindung (Zuordnung) dar. Beim Aufbau einer neuen Verbindung wird ein unbenutztes Array-Element der neuen Verbindung zugeordnet. Die Application-Association-Strukturvariable liefert Statusinformationen der Association.

errors: Error-Fifo.

logs: Log-Fifo.

nAllocLoad: Prozentualer Füllstatus der MMS-Objektdatenbank.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcMMS.Lib

5.2.2.8 ST_MMS_ApplicationAssociation

```
TYPE ST_MMS_ApplicationAssociation :
STRUCT
    aaIdentifier    : UDINT := 0;
    authentPassword : STRING(MAX_MMS_PASSWORD_LENGTH) := '';

    services        : ST_MMS_ServiceSupportOptions;
    parameters      : ST_MMS_ParameterSupportOptions;
    nestingLevel    : SINT := 0;


    bAuthentUser    : BOOL := FALSE;
(* TRUE = User with authentication, FALSE = User without authentication *)
    eEnvironment    : E_MMS_Environment := eMMS_Environment_None;
END_STRUCT
END_TYPE
```

aaIdentifier: Application-Association-Identifier.

authentPassword: Authentifizierungs-Passwort [▶ 53].

services: Unterstützte MMS-Dienste [▶ 49].

parameters: Unterstützte MMS-CBB-Parameter [▶ 50].

nestingLevel: Maximale Verschachtelungstiefe der MMS-Datenstruktur.

bAuthentUser: Diese Variable liefert TRUE wenn beim Aufbau der Application-Association ein Authentifizierungs-Passwort benutzt wurde.

eEnvironment: Status der MMS-Laufzeitumgebung [▶ 46].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcMMS.Lib

5.2.2.9 ST_MMS_DebugPara

```
TYPE ST_MMS_DebugPara :
STRUCT
    bError  : BOOL := FALSE; (* Enable/disable error messages. *)
    bLog    : BOOL := FALSE; (* Enable/disable log messages. *)

    mms     : ST_ProtocolDebugPara;
    ulosi   : ST_ProtocolDebugPara;
    cotp    : ST_ProtocolDebugPara;
END_STRUCT
END_TYPE
```

bError: Aktiviert/deaktiviert die Ausgabe von Fehlermeldungen im TwinCAT System Manager Logger-View-Fenster.

bLog: Aktiviert/deaktiviert die Ausgabe von Logmeldungen (Informationen, Warnungen) im TwinCAT System Manager Logger-View-Fenster.

mms: Diese Struktur [▶ 52] beinhaltet die Konfigurationsparameter für die Debugausgaben der der MMS-Schicht;

ulosi: Diese Struktur beinhaltet die Konfigurationsparameter für die Debugausgaben der Upper-Layer-OSI-Schichten (ACSI, Presentation, Session).

cotp: Diese Struktur beinhaltet die Konfigurationsparameter für die Debugausgaben der Transport-Schicht.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcMMS.Lib

5.2.2.10 ST_ProtocolDebugPara

```
TYPE ST_ProtocolDebugPara:
STRUCT
    eTx   : E_DbgDirection := eDbgDirection_OFF;(* Enable/disable debug output of data-requests. *)
    eRx   : E_DbgDirection := eDbgDirection_OFF;(* Enable/disable debug output of data-
indications. *)
    eCtrl : E_DbgDirection := eDbgDirection_OFF;(* Enable/
disable debug output of connection control events/actions. *)
END_STRUCT
END_TYPE
```

eTx : Aktiviert die Ausgabe der ausgehenden Telegramme.

eRx : Aktiviert die Ausgabe der eingehenden Telegramme.

eCtrl : Aktiviert die Ausgabe der Kontrollnachrichten.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcMMS.Lib

5.2.2.11 E_COTP_DUsize

```
TYPE E_COTP_DUsize:
(
    eCOTP_DUsize_Unused := 0, (* *)
    eCOTP_DUsize_128 := 7, (* 128 byte (default) *)
    eCOTP_DUsize_256 := 8, (* 256 byte *)
    eCOTP_DUsize_512 := 9, (* 512 byte *)
    eCOTP_DUsize_1024 := 10, (* 1024 byte *)
    eCOTP_DUsize_2048 := 11, (* 2048 byte *)
    eCOTP_DUsize_4096 := 12 , (* 4096 byte *)
    eCOTP_DUsize_8192 := 13 , (* 8192 byte *)
    eCOTP_DUsize_16384 := 14, (* 16384 byte *)
```

```
    eCOTP_DUsize_32768 := 15 (* 32768 byte *)
);
END_TYPE
```

Größere Daten werden von der Transportschicht segmentiert. Diese Variable konfiguriert die maximale Länge eines Datensegments.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcRFC1006.Lib

5.2.3 Konstanten

5.2.3.1 MMS-Konstanten

Wert	Wert	Beschreibung
MAX_MMS_APPLICATION_ASSOCIATIONS	1	Maximale Anzahl gleichzeitig aktiver Application-Associations.
MAX_MMS_PASSWORD_LENGTH	10	Maximale Länge des Authentifizierungs-Passworts.
MAX_MMS_ACCESS_CONTROL_USERS	4	Maximale Anzahl der Benutzerkonten.

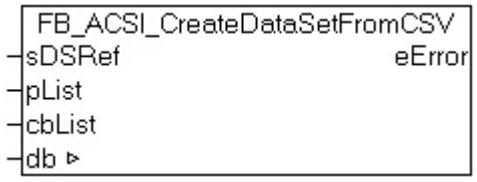
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcMMS.Lib

5.3 TcACSI

5.3.1 Funktionsbausteine

5.3.1.1 FB_ACSI_CreateDataSetFromCSV



Mit diesem Funktionsbaustein können DataSets für den IEC 61850/61400-25 Server erzeugt werden. Das Erzeugen der DataSets kann erst nach Abschluss der Initialisierung aller Logical Devices erfolgen.

VAR_IN_OUT

```
VAR_IN_OUT
    db : T_HACSIDB; (* ACSI configuration database *)
END_VAR
```

db: Ist die Datenbank des IEC-Servers

VAR_INPUT

```
VAR_INPUT
    sDSRef  : T_MaxString := ''; (* data-set reference string (data-set name)*)
 pList   : DWORD := 0; (* Adress of data-set member list (CSV-String) *)
 cbList  : UDINT := 0; (* Byte size of data-set member list (CSV-String) *)
END_VAR
```

sDSRef: Gibt alle DataSet Referenzen der entsprechenden DataSet-Instanz an.

pList: Pointer (Adresse) auf die Liste der Struktur der DataSet-Instanz. Die Adresse kann mit dem ADR-Operator ermittelt werden.

cbList: Länge der Liste der DataSet-Instanz.

VAR_OUTPUT

```
VAR_OUTPUT
    eError : E_ACSI_ServiceError := eACSI_ServiceError_Success;
END_VAR
```

eError: Rückgabeparameter. Siehe E_ACSI_ServiceError [▶ 59]

Beispiel:

Im folgenden Beispiel wird das im TcTelecontrol angelegte DataSet für den IEC-Server erzeugt.

```
PROGRAM MAIN
VAR
    fbCreateDataSet : FB_ACSI_CreateDataSetFromCSV; DataSet_1 : ST_DataSet_IEDDevice_XCBR1_DataSet_1
; (* Your configured DataSet from TcTelecontrol *)
END_VAR
```

```
...
(* create DataSet *)
fbCreateDataSet(sDSRef := DataSet_1.sDSRef, pList := ADR(DataSet_1.sList), cbList := LEN(DataSet_1.s
List) +1, db := fbIEC61850Server.system.db, eError =>errID);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcACSI.Lib

5.3.1.2 Event Detection FUNCTION_BLOCKS

Diese Funktionsbausteine werden genutzt, um Wertänderungen an den Variablen eines IEC-Servers zu erkennen. Es können verschiedene Events die an einer Variablen auftreten detektiert werden. Der Baustein kann auch verwendet werden, um ein Reporting Frame an einen Client zu schicken. Dafür müssen die entsprechenden Variablen in einem Report Control Block als Referenz vorhanden sein.

Diese Art von Funktionsbaustein existiert in der TcACSI.lib für jeden IEC-Datentyp. Da sich die Bausteine auch nur an diesem Datentyp unterscheiden gibt es auf dieser Seite eine Übersicht der Bausteine, von denen stellvertretend einige beschrieben werden.

Alle Funktionsbausteine besitzen drei Aktionen:

- **A_SetValue:** Setzt einen Wert in der IEC Datenstruktur und teilt dem Server-Baustein mit, das der Wert eines Datenattributes mit einem neuen oder dem selben Wert beschrieben wurde (UPDATE oder CHANGE). Nur so kann der Server-Baustein erkennen das er für dieses Datenattribut ein Reporting generieren soll. Ein Reporting wird aber auch nur dann generiert, wenn dieses Attribut in dem zugehörigen Dataset des Report-Control-Blocks konfiguriert wurde.

- **A_GetEvent:** Erkennt, ob ein Client ein Datenattribut in der IEC-Datenstruktur geändert oder gelesen hat. Aktuell können folgende Events detektiert werden: ACSI_EVENT_WRITE, ACSI_EVENT_READ, ACSI_EVENT_UPDATE, ACSI_EVENT_CHANGE. A_GetEvent wird gemeinsam mit A_ClearEvent verwendet.
- **A_ClearEvent:** Setzt ein detektiertes Event zurück, damit die nächste Änderung erkannt werden kann.

	A_SetValue	A_GetEvent	A_ClearEvent
value, bits or octets	Der neue Attributwert der geschrieben werden soll	Wird nicht verwendet	Wird nicht verwendet
event_ctrl	Reserviert. Sollte := 0 sein	Wird nicht verwendet	Legt fest, welches Event zurückgesetzt werden soll. Mehrere Events können bei einem Aufruf zurückgesetzt werden. Folgende Werte können mit "OR" verknüpft werden: ACSI_EVENT_WRITE, ACSI_EVENT_READ, ACSI_EVENT_UPDATE, ACSI_EVENT_CHANGE. Um alle Events auf einmal zurückzusetzen, kann der Wert 0xFFFFFFFF verwendet werden
event_ind	Diese Variable ist eine Bitmaske. Jedes Bit entspricht einem Event. Ein Event steht an, wenn das entsprechende Bit gesetzt ist	Diese Variable ist eine Bitmaske. Jedes Bit entspricht einem Event. Ein Event steht an, wenn das entsprechende Bit gesetzt ist	Diese Variable ist eine Bitmaske. Jedes Bit entspricht einem Event. Ein Event steht an, wenn das entsprechende Bit gesetzt ist
event_cnf	Wird zurzeit nicht verwendet	Wird zurzeit nicht verwendet	Wird zurzeit nicht verwendet

Funktionsbausteine mit Interface "value":

```
FB_ACSI_BOOLEAN           ⇒ BOOLEAN

FB_ACSI_INT8              ⇒ INT8
FB_ACSI_INT16             ⇒ INT16
FB_ACSI_INT32             ⇒ INT32
FB_ACSI_INT64             ⇒ INT64
FB_ACSI_INT128            ⇒ INT128

FB_ACSI_UINT8             ⇒ UINT8
FB_ACSI_UINT16            ⇒ UINT16
FB_ACSI_UINT32            ⇒ UINT32
FB_ACSI_UINT64            ⇒ UINT64
FB_ACSI_UINT128           ⇒ UINT128

FB_ACSI_FLOAT32           ⇒ FLOAT32
FB_ACSI_FLOAT64           ⇒ FLOAT64

FB_ACSI_ENUMERATED        ⇒ ENUMERATED

FB_ACSI_VISIBLESTRING255  ⇒ VISIBLESTRING255
FB_ACSI_UNICODESTRING255  ⇒ UNICODESTRING64

FB_ACSI_ENTRYTIME         ⇒ ENTRYTIME
FB_ACSI_TIMESTAMP         ⇒ TIMESTAMP
FB_ACSI_ENTRYID           ⇒ ENTRYID
FB_ACSI_QUALITY           ⇒ QUALITY
FB_ACSI_TRIGGERCON        ⇒ TRIGGERCON
FB_ACSI_OPTIONFLDS        ⇒ OPTIONFLDS
```

Funktionsbaustein FB_ACSI_BOOLEAN:

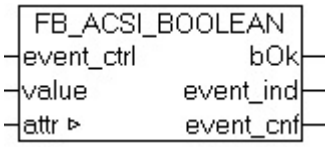


Abb. 1: TcPlcLibACSI_FB_ACSI_BOOLEAN

VAR_IN_OUT

```
VAR_IN_OUT
    attr        : T_ACSI_BOOLEAN; (* ACSI attribute; It depends on the data type *)END_VAR
```

attr: Ist das Attribut des IEC-Standards des entsprechenden Datentyps.

VAR_INPUT

```
VAR_INPUT
    event_ctrl  : DWORD := 16#FFFFFFFF;
(* Event control mask (used by A_ClearEvent), default: clear all events *)
    value       : BOOL;(* New value to set; The data type depends on the Function Block *)
END_VAR
```

event_ctrl: Siehe Tabelle oben.

value: Siehe Tabelle oben.

VAR_OUTPUT

```
VAR_OUTPUT
    bOk             : BOOL;(* TRUE = Success, FALSE = Access failed *)
    event_ind       : DWORD := 16#00000000;
(* Event indication    bits, bit set => event indication, bit reset => no event indication *)
    event_cnf       : DWORD := 16#FFFFFFFF;
(* Event confirmation bits, bit set => event confirmed, bit reset => waiting for event confirmation
*)
END_VAR
```

bOk: Ist dieser Wert TRUE sind keine Fehler aufgetreten die Bearbeitung kann fortgesetzt werden. Bei FALSE liegt ein Fehler vor.

event_ind: Siehe Tabelle oben.

event_cnf: Siehe Tabelle oben.

Beispiel:

Im folgenden Beispiel wird die Anwendung von A_SetValue, A_GetEvent und A_ClearEvent gezeigt.

```
PROGRAM MAIN
VAR
    dataCtrl : ST_ACSI_DATACTRL; bValue  : T_ACSI_BOOLEAN; (* Value from the data type BOOLEAN *)
END_VAR
---------------------(* Read Value *)
ctrl.BOOLEAN.A_GetEvent(attr := IEDRelay.XCBR1.BlkCls.stVal, event_ctrl := 0);
IF ctrl.BOOLEAN.bOk THEN  (* Detect when value is written (check the event_ind bits )*)IF (ctrl.BOOL
EAN.event_ind AND ACSI_EVENT_WRITE) = ACSI_EVENT_WRITE THEN
        write_bool_counter := write_bool_counter + 1;
    END_IF(* Detect when value is read (check the event_ind bits) *)IF (ctrl.BOOLEAN.event_ind AND A
CSI_EVENT_READ) = ACSI_EVENT_READ THEN
        read_bool_counter := read_bool_counter + 1;
    END_IF(* clear events (is necessary to detect the next event)*)
    ctrl.BOOLEAN.A_ClearEvent(attr := IEDRelay.XCBR1.BlkCls.stVal, event_ctrl := ACSI_EVENT_WRITE OR
 ACSI_EVENT_READ);(* clear read and write events *)END_IF
---------------------(* Write value *)
dataCtrl.BOOLEAN.A_SetValue(attr :=IEDDevice.XCBR1.SumSwARs.actVal, value := bValue.value, event_ctr
l := 0);
```

Funktionsbaustein mit Interface "bits":

```
FB_ACSI_CODEDENUM8      ⇒ CODEDENUM8
FB_ACSI_CODEDENUM16     ⇒ CODEDENUM16
FB_ACSI_CODEDENUM32     ⇒ CODEDENUM32
FB_ACSI_CODEDENUM64     ⇒ CODEDENUM64
FB_ACSI_CODEDENUM128    ⇒ CODEDENUM128
```

Funktionsbaustein FB_ACSI_CODEDENUM8:

VAR_IN_OUT

```
VAR_IN_OUT
    attr        : T_ACSI_CODEDENUM8; (* ACSI attribute; It depends on the data type *)
END_VAR
```

attr: Ist das Attribut des IEC-Standards vom entsprechenden Datentyp.

VAR_INPUT

```
VAR_INPUT
    event_ctrl  : DWORD := 16#FFFFFFFF;
(* Event control mask (used by A_ClearEvent), default: clear all events *)
    bits        : ARRAY[0..0] OF BYTE := 1(0); (* New value to set; The data type depends on the Fun
ction Block *)
END_VAR
```

event_ctrl: Siehe Tabelle oben.

bits: Siehe Tabelle oben.

VAR_OUTPUT

```
VAR_OUTPUT
    bOk             : BOOL;(* TRUE = Success, FALSE = Access failed *)
    event_ind       : DWORD := 16#00000000;
(* Event indication    bits, bit set => event indication, bit reset => no event indication *)
    event_cnf       : DWORD := 16#FFFFFFFF;
(* Event confirmation bits, bit set => event confirmed, bit reset => waiting for event confirmation
*)
END_VAR
```

bOk: Ist dieser Wert TRUE sind keine Fehler aufgetreten die Bearbeitung kann fortgesetzt werden. Bei FALSE liegt ein Fehler vor.

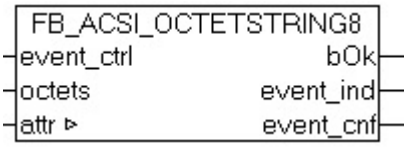
event_ind: Siehe Tabelle oben.

event_cnf: Siehe Tabelle oben.

Funktionsbaustein mit Interface "octets":

```
FB_ACSI_OCTETSTRING8    ⇒ OCTETSTRING8
FB_ACSI_OCTETSTRING16   ⇒ OCTETSTRING16
FB_ACSI_OCTETSTRING32   ⇒ OCTETSTRING32
FB_ACSI_OCTETSTRING64   ⇒ OCTETSTRING64
```

Funktionsbaustein FB_ACSI_OCTETSTRING8:



VAR_IN_OUT

```
VAR_IN_OUT
    attr        : T_ACSI_OCTETSTRING8; (* ACSI attribute; It depends on the data type *)
END_VAR
```

attr: Ist das Attribut des IEC-Standards vom entsprechenden Datentyp.

VAR_INPUT

```
VAR_INPUT
    event_ctrl      : DWORD := 16#FFFFFFFF;
(* Event control mask (used by A_ClearEvent), default: clear all events *)
 octets      : ARRAY[0..7] OF BYTE := 8(0);
(* OCTETS: 8 byte; New value to set; The data type depends on the Function Block *)
END_VAR
```

event_ctrl: Siehe Tabelle oben.

bits: Siehe Tabelle oben.

VAR_OUTPUT

```
VAR_OUTPUT
    bOk         : BOOL;(* TRUE = Success, FALSE = Access failed *)
    event_ind           : DWORD := 16#00000000;
(* Event indication bits, bit set => event indication, bit reset => no event indication *)
    event_cnf       : DWORD := 16#FFFFFFFF;
(* Event confirmation bits, bit set => event confirmed, bit reset => waiting for event confirmation
*)
END_VAR
```

bOk: Ist dieser Wert TRUE sind keine Fehler aufgetreten die Bearbeitung kann fortgesetzt werden. Bei FALSE liegt ein Fehler vor.

event_ind: Siehe Tabelle oben.

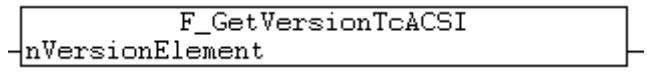
event_cnf: Siehe Tabelle oben.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcACSI.Lib

5.3.2 Funktionen

5.3.2.1 F_GetVersionTcACSI



Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcACSI : UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcACSI.Lib

5.3.3 Datentypen

5.3.3.1 E_ACSI_ServiceError

ACSI service errors

Konstante	Wert	Beschreibung
eACSI_ServiceError_Success	0x0000	Kein Fehler
eACSI_ServiceError_InstanceNotAvailable	0x8500	Instanz nicht verfügbar
eACSI_ServiceError_InstanceInUse	0x8501	Instanz in Benutzung
eACSI_ServiceError_AccessViolation	0x8502	Zugriffsverletzung
eACSI_ServiceError_AccessNotAllowedInCurrentState	0x8503	Zugriff im aktuellen Zustand nicht erlaubt
eACSI_ServiceError_ParameterValueInappropriate	0x8504	Parameterwert ist ungeeignet
eACSI_ServiceError_ParameterValueInconsistent	0x8505	Parameterwert ist folgewidrig
eACSI_ServiceError_ClassNotSupported	0x8506	Klasse wird nicht unterstützt
eACSI_ServiceError_InstanceLockedByOtherClient	0x8507	Instanz verriegelt durch einen anderen client
eACSI_ServiceError_ControlMustBeSelected	0x8508	Bedienung muss angewählt werden
eACSI_ServiceError_TypeConflict	0x8509	Typkonflikt
eACSI_ServiceError_FailedDueToCommunicationsConstraint	0x850A	Fehlgeschlagen wegen Einschränkung in der Kommunikation
eACSI_ServiceError_FailedDueToServerConstraint	0x850B	Fehlgeschlagen wegen Einschränkung im Server

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcACSI.Lib

5.3.3.2 ST_ACSI_DATACTRL

Die Struktur DATACTRL beinhaltet Funktionsbausteine für alle IEC-Datentypen, um eine Wertänderungskontrolle durchzuführen. Diese können für die Realisierung der Reportingfunktionalität genutzt werden.

```
TYPE ST_ACSI_StartTag:(* ACSI-Type start delimiter *)
STRUCT
    BOOLEAN          : FB_ACSI_BOOLEAN;

    INT8             : FB_ACSI_INT8;
    INT16            : FB_ACSI_INT16;
    INT32            : FB_ACSI_INT32;
    INT64            : FB_ACSI_INT64;
    INT128           : FB_ACSI_INT128;

    UINT8            : FB_ACSI_UINT8;
    UINT16           : FB_ACSI_UINT16;
    UINT32           : FB_ACSI_UINT32;
    UINT64           : FB_ACSI_UINT64;
    UINT128          : FB_ACSI_UINT128;
```

```
    FLOAT32          : FB_ACSI_FLOAT32;
    FLOAT64          : FB_ACSI_FLOAT64;

    ENUMERATED       : FB_ACSI_ENUMERATED;

    CODEDENUM8       : FB_ACSI_CODEDENUM8;
    CODEDENUM16      : FB_ACSI_CODEDENUM16;
    CODEDENUM32      : FB_ACSI_CODEDENUM32;
    CODEDENUM64      : FB_ACSI_CODEDENUM64;
    CODEDENUM128     : FB_ACSI_CODEDENUM128;

    OCTETSTRING8     : FB_ACSI_OCTETSTRING8;
    OCTETSTRING16    : FB_ACSI_OCTETSTRING16;
    OCTETSTRING32    : FB_ACSI_OCTETSTRING32;
    OCTETSTRING64    : FB_ACSI_OCTETSTRING64;

    VISIBLESTRING255 : FB_ACSI_VISIBLESTRING255;
    UNICODESTRING64  : FB_ACSI_UNICODESTRING255;

    ENTRYTIME        : FB_ACSI_ENTRYTIME;
    TIMESTAMP        : FB_ACSI_TIMESTAMP;
    ENTRYID          : FB_ACSI_ENTRYID;
    QUALITY          : FB_ACSI_QUALITY;
    TRIGGERCON       : FB_ACSI_TRIGGERCON;
    OPTIONFLDS       : FB_ACSI_OPTIONFLDS;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcACSI.Lib

5.3.4 Konstanten

5.3.4.1 ACSI-Konstanten

ACSI-Objektdatenbank-Parameter

Konstante	Wert	Beschreibung
MAX_ACSI_OBJECTNODE_ALLOC	30000	Max. Anzahl der verfügbaren Objekte
MAX_ACSI_OBJECTNAME_LENGTH	31	Max. Objektname-Stringlänge (Zeichen)

Functional Constraints bit mask

Alle Functional Constraints der Basisnorm IEC 61850 wurden in der TwinCAT PLC als Konstante der ACSI Library implementiert.

```
ACSI_FC_ST :DWORD := 16#00000001; (* Status information (use of FC in the definition of DATA) *)
ACSI_FC_MX :DWORD := 16#00000002; (* Measurands (analogue values) (use of FC in the definition of DA
TA) *)
ACSI_FC_CO :DWORD := 16#00000004; (* Control (use of FC in the definition of DATA) *)
ACSI_FC_SP :DWORD := 16#00000008; (* Setpoint (use of FC in the definition of DATA and control block
s) *)
ACSI_FC_SV :DWORD := 16#00000010; (* Substitution (use of FC in the definition of DATA) *)
ACSI_FC_CF :DWORD := 16#00000020; (* Configuration (use of FC in the definition of DATA) *)
ACSI_FC_DC :DWORD := 16#00000040; (* Description (use of FC in the definition of DATA) *)
ACSI_FC_SG :DWORD := 16#00000080; (* Setting group (use of FC in the definition of DATA) *)
ACSI_FC_SE :DWORD := 16#00000100; (* Setting group editable (use of FC in the definition of DATA) *)
ACSI_FC_EX :DWORD := 16#00000200; (* Extended definition (use of FC in the definition of DATA) *)
ACSI_FC_BR :DWORD := 16#00000400; (* Buffered report (-
> Reserved for control classes and use of the FC in the definition of control blocks) *)
ACSI_FC_RP :DWORD := 16#00000800; (* Unbuffered report (-
> Reserved for control classes and use of the FC in the definition of control blocks) *)
ACSI_FC_LG :DWORD := 16#00001000; (* Logging (-
> Reserved for control classes and use of the FC in the definition of control blocks) *)
ACSI_FC_GO :DWORD := 16#00002000; (* Goose control (-
> Reserved for control classes and use of the FC in the definition of control blocks) *)
```

```
ACSI_FC_GS :DWORD := 16#00004000; (* Gsse control (-
> Reserved FOR control classes AND use OF the FC in the definition OF control blocks) *)
ACSI_FC_MS :DWORD := 16#00008000; (* Multicast sampled value control (-
> Reserved for control classes and use of the FC in the definition of control blocks) *)
ACSI_FC_US :DWORD := 16#00010000; (* Unicast sampled value control (-
> Reserved for control classes and use of the FC in the definition of control blocks) *)
ACSI_FC_XX :DWORD := 16#00020000; (* Shall represent all DataAttributes of a Data of any FC. "XX" sh
all not used as value in DataAttributes *)
```

Trigger Options bit mask

```
(* ACSI_TrgOp_rsv : DWORD := 16#80; reserved *)
ACSI_TrgOp_dchg : DWORD := 16#40;(* data-change *)
ACSI_TrgOp_qchg : DWORD := 16#20;(* quality-change *)
ACSI_TrgOp_dupd : DWORD := 16#10;(* data-update *)
ACSI_TrgOp_intg : DWORD := 16#08;(* integrity *)
ACSI_TrgOp_gi : DWORD := 16#04;(* general-interrogation *)
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10.0 Build >= 1340	PC or CX (x86, ARM)	TcACSI.Lib

6 Beispiele

Die Beispiele verwenden folgende Host-IP-Adresse: "127.0.0.1" (Loopback-Adapter). Die Host-Adresse im Beispielprojekt muss passend zu der Adresse Ihrer Zielplattform eingestellt werden, auf welcher der Server laufen soll. Soll z.B. ein CX10xx mit der IP-Adresse "172.16.11.83" als Server dienen, dann müssen Sie in dem Beispielprojekt die IP-Adresse des CX10xx konfigurieren und das Projekt auf dieses Gerät runterladen. Im Client (Leitstation) wird ebenfalls die Adresse des CX10xx als Host-Adresse eingetragen.

Die max. Anzahl der Knoten bei einer Evaluierungsversion des Servers wird auf drei eingeschränkt.

Beispiel	Beschreibung
https://infosys.beckhoff.com/content/1031/tcplclibiec61850server/Resources/11681715851.zip	Ein kleiner IEC 61850 Demo-Server mit folgenden Knoten: LLN0, LPHD, MMXU, XCBR

7 Anhang

7.1 TCP Keep-Alive Messages

Ein Keep-Alive Telegramm ist eine Bestätigungsnachricht bzw. ein "acknowledge". Damit kann im Hintergrund überprüft werden, ob ein Kommunikationspartner, welcher eine Verbindung hergestellt hat, noch aktiv ist und somit noch an der Kommunikation teilnimmt. Sollte ein Kommunikationspartner nicht mehr aktiv sein, wird der Kommunikationskanal sauber und regulär geschlossen, um für einen neuen Teilnehmer frei zu sein.

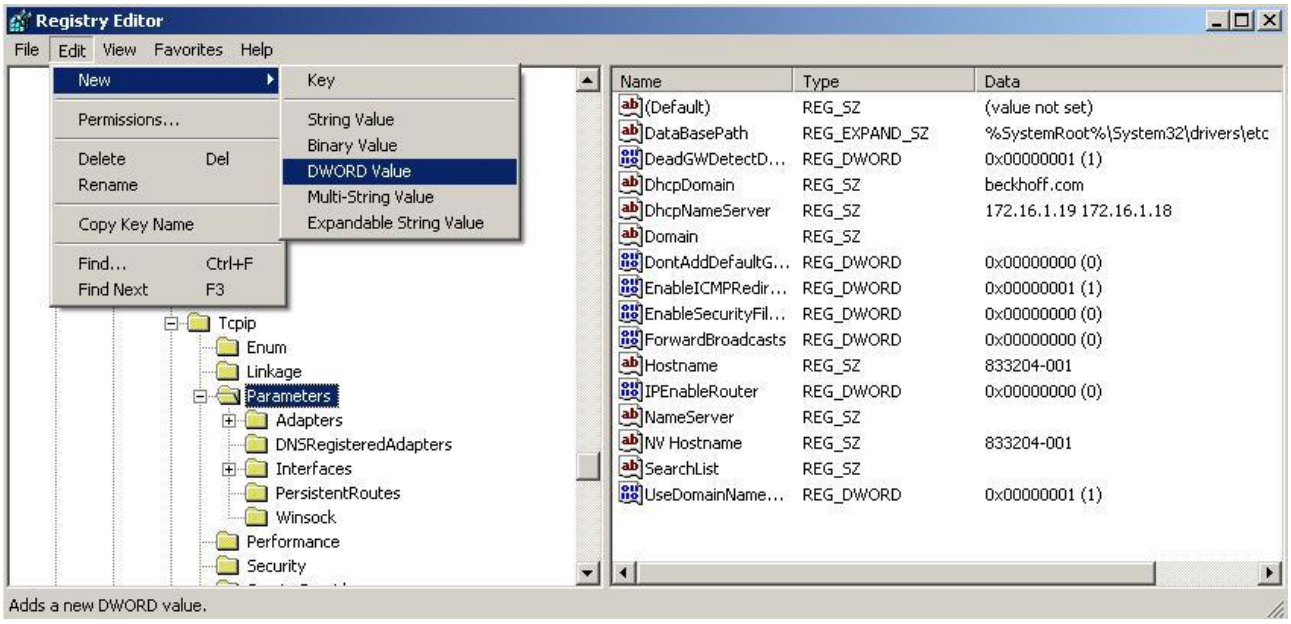
Keep-Alive kann durch die Schlüssel Keep-Alive-Time und Keep-Alive-Interval in der Registrierung konfiguriert werden. Der Defaultwert für die Keep-Alive-Time ist auf zwei Stunden eingestellt, die Interval-Time, also die Zeit bis zur Wiederholung einer nicht beantworteten Alive-Anfrage, liegt Default bei einer Sekunde. **Alle Werte werden in Millisekunden angegeben**. Wenn die entsprechenden Schlüssel noch nicht in der Registrierung vorhanden sind, müssen sie wie folgt in angelegt werden. Es handelt sich nicht um TwinCAT Einstellungen sondern um spezifische Einstellungen des Betriebssystems. Weiterführende Informationen finden Sie daher in der Betriebssystemdokumentation z.B. auf der Microsoft Internetseite.

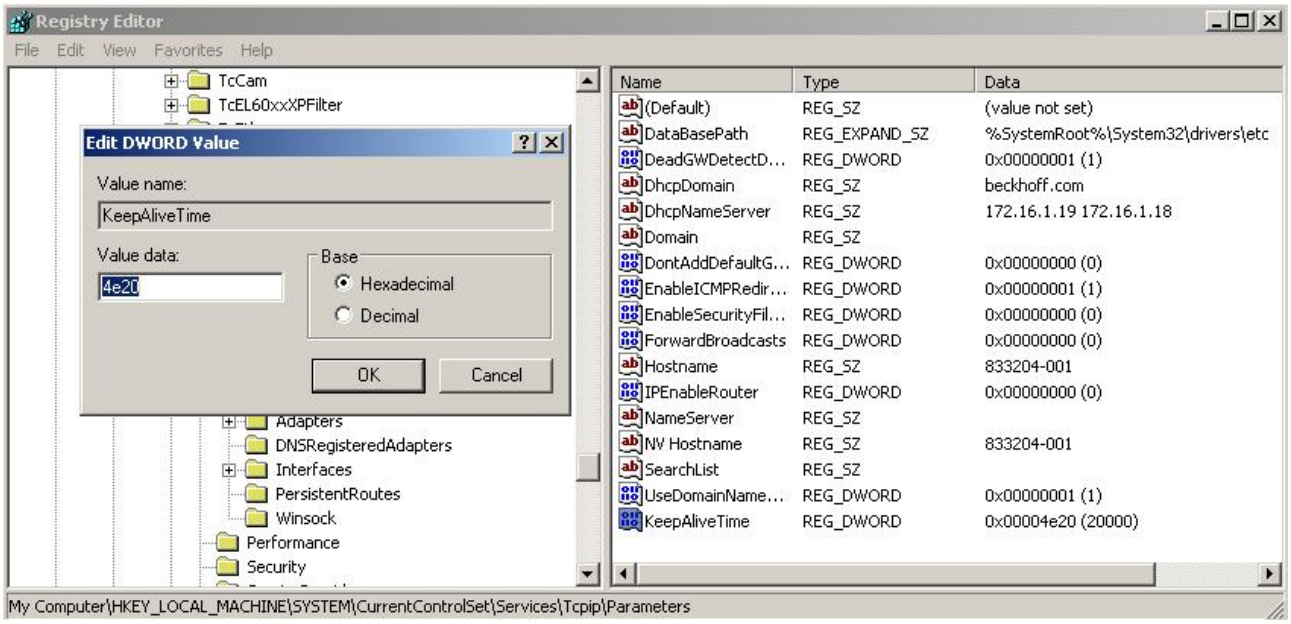
Unter Windows W2K, XP, Windows Embedded Standard:

- Im Ordner HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\ muss ein Objekt mit dem Namen KeepAliveTime als DWORD angelegt werden. Fügen Sie dazu unter Edit -> New -> DWORD Value ein. Stellen Sie hier als Wert 0x4E20 ein, entspricht dies 20 Sekunden.
- Im Ordner HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\ muss ein Objekt mit dem Namen KeepAliveInterval als DWORD angelegt werden. Fügen Sie dazu unter Edit -> New -> DWORD Value ein. Stellen Sie hier als Wert 0x1388 ein, entspricht dies 5 Sekunden.

Unter Windows CE:

- Im Ordner HKEY_LOCAL_MACHINE\Comm\TcpIp\Parms\ muss ein Objekt mit dem Namen KeepAliveTime als DWORD angelegt werden. Fügen Sie dazu unter Edit -> New -> DWORD Value ein. Stellen Sie hier als Wert 0x4E20 ein, entspricht dies 20 Sekunden.
- Im Ordner HKEY_LOCAL_MACHINE\Comm\TcpIp\Parms\ muss ein Objekt mit dem Namen KeepAliveInterval als DWORD angelegt werden. Fügen Sie dazu unter Edit -> New -> DWORD Value ein. Stellen Sie hier als Wert 0x1388 ein, entspricht dies 5 Sekunden.





7.2 Firewall Einstellungen

Der TwinCAT IEC 61850/61400-25 Server verwendet während der Client/Server-Kommunikation (MMS-Mapping) das TCP/IP als Transportprotokoll (T-Profile). Es ist daher darauf zu achten, dass der entsprechende TCP-Port bei Benutzung einer Firewall frei geschaltet wird. Die untere Tabelle listet Ports auf, die bei der Benutzung einer Firewall zu berücksichtigen sind.

Beschreibung	Typ	Protokoll	Port
Connection Oriented Transport Protocol (COTP, ISO 8073)	Protocol	TCP	102

Die Konfiguration der Windows Firewall wird über den entsprechenden Dialog in der Systemsteuerung vorgenommen. Weitere Informationen zur Konfiguration finden Sie in der Windows bzw. Firewall Dokumentation.

Achten Sie bitte bei einem Embedded Controller ohne Monitoranschluss und USB darauf, dass Sie den Port für Remote Display (Windows CE) oder Remote Desktop (Windows XP / Windows Vista) in der Firewall frei schalten. Ansonsten haben Sie keine Möglichkeit mehr, den Rechner über das Netzwerk zu administrieren.

7.3 FAQ - Häufig gestellte Fragen und Antworten

In diesem Bereich werden häufig gestellte Fragen beantwortet, um Ihnen die Arbeit mit dem TwinCAT IEC 61850 Server zu erleichtern.
Wenn Sie noch weitere Fragen haben, kontaktieren Sie bitte unseren Support (-157)

1. Können eigene Logical Nodes und Common Data Classes anlegt werden? [▶ 64]
2. Können bereits im Standard existierende Logical Nodes und Common Data Classes erweitert werden? [▶ 65]
3. Welche SPS Bibliothek muss für den Kommunikationsstack in TwinCAT PLC Control einbunden werden? [▶ 65]

? Können eigene Logical Nodes und Common Data Classes anlegt werden?
!Ja, dies ist im Telecontrol Konfigurator möglich. Durch die Verwendung des Konfigurators werden die eigenen Definitionen in eine "private" XML Datenbank geschrieben. Die entsprechenden Objekte stehen für jede neue Konfiguration im Konfigurator zur Verfügung. Weitere Informationen unter Private Logical Nodes [▶ 35] und Private Common Data Classes [▶ 37].

? Können bereits im Standard existierende Logical Nodes und Common Data Classes erweitert werden?
!Ja, Erweiterungen sind möglich. Allerdings kann nicht der Name des Originals verwendet und die Erweiterung nur in der "privaten" XML Datenbank gespeichert werden. Wie dies genau funktioniert, erfahren Sie unter Private Logical Nodes [▶ 35] und Private Common Data Classes [▶ 37].

? Welche SPS Bibliothek muss für den Kommunikationsstack in TwinCAT PLC Control einbunden werden?
!Es muss nach einer erfolgreichen Installation die TcIEC61850Server.lib aus dem TwinCAT Bibliotheksverzeichnis in das TwinCAT PLC Control Projekt eingebunden werden.

Mehr Informationen:
www.beckhoff.de/ts6511

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.de
www.beckhoff.de