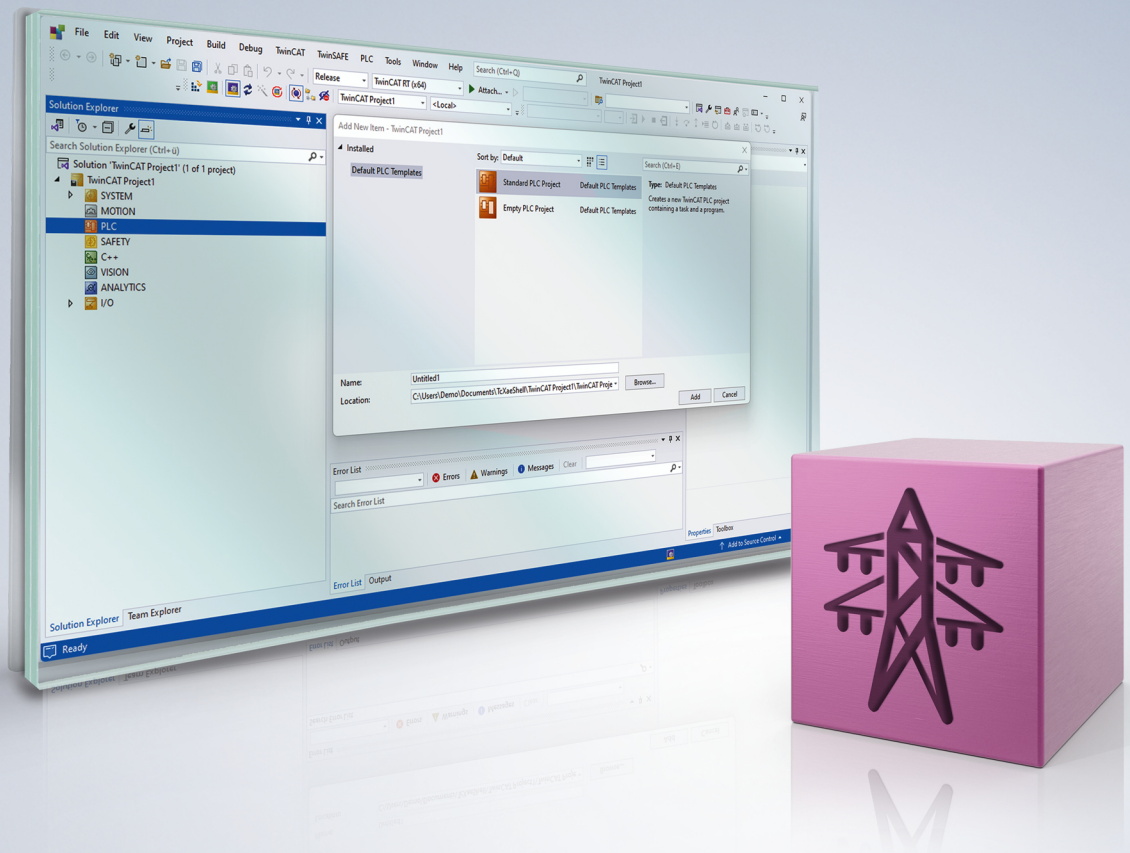
BECKHOFF New Automation Technology

Manual | EN

TF6510

TwinCAT 3 | IEC 61850



2026-07-29 | Version: 1.9.0

Table of contents

1 Foreword 13
1.1 Notes on the documentation 13
1.2 For your safety 14
1.3 Notes on information security 15
2 Overview 16
3 Installation 17
3.1 System requirements 17
3.2 Installation 17
3.3 Licensing 19
4 Technical introduction 22
4.1 Overview of standards 22
4.2 Data model 23
4.3 Services supported 25
4.4 General Client - Project structure 27
4.5 General Server - Project structure 31
4.6 RT Ethernet adapter configuration 35
4.7 TCP Keep-Alive Messages 38
4.8 Firewall settings 40
5 Configuration 42
5.1 Basics and entry 42
5.2 Menu bar 49
5.2.1 File 49
5.2.2 View 50
5.2.3 Tools 51
5.2.4 Help 51
5.3 New Project Wizard 52
5.4 Device configuration 60
5.5 Configuration of the data objects 61
5.6 DataSets 66
5.7 Reporting 71
5.8 GOOSE 73
5.9 Sampled Values 75
5.10 Automatic code generation 80
5.10.1 Code version 2.0 84
5.11 Custom Logical Node Groups 96
5.12 Custom Logical Nodes 96
5.13 Custom Common Data Classes 98
6 PLC API 101
6.1 Libraries 101
6.2 Namespaces 101
6.2.1 TwinCAT TypeSystem namespace 101
6.2.2 Tc3_Acsi namespace 101
6.2.3 Tc3_Ber namespace 101

6.2.4 Tc3_Collections namespace ... 101
6.2.5 Tc3_Gse namespace ... 102
6.2.6 Tc3_iec61850_8_1 namespace ... 102
6.2.7 Tc3_iec61850 namespace ... 102
6.2.8 Tc3_Mms namespace ... 102
6.2.9 Tc3_Rfc1006 namespace ... 102
6.2.10 Tc3_Sockets namespace ... 102
6.2.11 Tc3_Tpkt namespace ... 102
6.2.12 Tc3_Ulosi namespace ... 102
6.2.13 Tc2_TcpIp namespace ... 102
6.2.14 Tc2_System namespace ... 102
6.3 Function blocks ... 103
6.3.1 Data model ... 103
6.3.2 Protocol settings ... 173
6.3.3 Communication ... 247
6.3.4 FB_AsyncServiceResultClass ... 299
6.3.5 FB_ServiceErrorClass ... 300
6.4 Functions ... 300
6.4.1 Accuracy_To_UtcTimeQualityAccuracy ... 300
6.4.2 AcsiAnalogueValue ... 302
6.4.3 AcsiCalendarTime ... 303
6.4.4 AcsiCell ... 303
6.4.5 AcsiLogOptionalFields ... 304
6.4.6 AcsiOptionalFields ... 305
6.4.7 AcsiOriginator ... 305
6.4.8 AcsiPhyComAddr ... 306
6.4.9 AcsiPoint ... 307
6.4.10 AcsiPulseConfig ... 307
6.4.11 AcsiQuality ... 308
6.4.12 AcsiQuality_To_String ... 309
6.4.13 AcsiQuality_To_WORD ... 310
6.4.14 AcsiQualitySource_To_Source ... 311
6.4.15 AcsiQualityValidity_To_Validity ... 311
6.4.16 AcsiRangeConfig ... 312
6.4.17 AcsiReasonCode ... 313
6.4.18 AcsiScaledValueConfig ... 314
6.4.19 AcsiSvOptionalFields ... 314
6.4.20 AcsiTriggerConditions ... 315
6.4.21 AcsiUnit ... 316
6.4.22 AcsiValWithTrans ... 316
6.4.23 AcsiVector ... 317
6.4.24 AnyBuffer_To_OctetString ... 317
6.4.25 BinaryTime ... 319
6.4.26 BinaryTime_To_SystemTime ... 319
6.4.27 BinaryTime6_To_LWORD ... 320
6.4.28 BinaryTime6_To_String ... 321

6.4.29 BinaryTime6Day_To_Date ... 322
6.4.30 BYTE_To_UtcTimeQuality ... 323
6.4.31 CltsEveryBool ... 324
6.4.32 CltsSomeBool ... 325
6.4.33 DATE_and_TOD_To_DT ... 326
6.4.34 Date_To_BinaryTime6Day ... 327
6.4.35 FillOctetString ... 328
6.4.36 INT24 ... 329
6.4.37 INT128 ... 330
6.4.38 LTIME_To_UtcTimeFractionOfSecond ... 330
6.4.39 LWORD_To_BinaryTime6 ... 332
6.4.40 LWORD_TO_MACADDR ... 333
6.4.41 LWORD_TO_OCTET6 ... 333
6.4.42 LWORD_TO_OCTET8 ... 334
6.4.43 LWORD_To_UtcTime ... 334
6.4.44 OctetString_To_AnyBuffer ... 335
6.4.45 SizeOfOctetString ... 336
6.4.46 Source_To_AcsiQualitySource ... 337
6.4.47 String_To_AcsiQuality ... 338
6.4.48 String_To_BinaryTime6 ... 339
6.4.49 String_To_UtcTime ... 340
6.4.50 String_To_UtcTimeQuality ... 341
6.4.51 String_To_UtcTimeStamp ... 342
6.4.52 SystemTime_To_BinaryTime ... 343
6.4.53 SystemTime_To_UtcTime ... 344
6.4.54 UINT24 ... 346
6.4.55 UINT128 ... 346
6.4.56 UtcTime ... 347
6.4.57 UtcTime_To_LWORD ... 348
6.4.58 UtcTime_To_String ... 348
6.4.59 UtcTime_To_SystemTime ... 349
6.4.60 UtcTimeFractionOfSecond_To_LTIME ... 351
6.4.61 UtcTimeIsNull ... 352
6.4.62 UtcTimeQuality ... 353
6.4.63 UtcTimeQuality_To_BYTE ... 353
6.4.64 UtcTimeQuality_To_String ... 354
6.4.65 UtcTimeQualityAccuracy_To_Accuracy ... 356
6.4.66 UtcTimeStamp_To_String ... 357
6.4.67 Validity_To_AcsiQualityValidity ... 358
6.4.68 WORD_To_AcsiQuality ... 359
6.4.69 ZeroOctetString ... 360
6.5 Interfaces ... 361
6.5.1 Data model ... 361
6.5.2 Protocol settings ... 375
6.5.3 Communication ... 397
6.5.4 I_AsyncServiceResultClass ... 463

6.5.5 I_ServiceErrorClass ... 468
6.5.6 I_BaseStreamClass ... 471
6.6 Data types ... 471
6.6.1 E_ACSE_AareDiagnostic ... 471
6.6.2 E_ACSE_AareResult ... 472
6.6.3 E_ACSE_AareSource ... 473
6.6.4 E_ACSE_AbrtDiagnostic ... 473
6.6.5 E_ACSE_AbrtSource ... 473
6.6.6 E_AcseAuthentication ... 474
6.6.7 E_AcsiAccessPermission ... 474
6.6.8 E_AcsiAdjSt ... 475
6.6.9 E_AcsiAlmLev ... 475
6.6.10 E_AcsiAnalogueValuePresCond ... 475
6.6.11 E_AcsiAngRef ... 476
6.6.12 E_AcsiApplAddCause ... 476
6.6.13 E_AcsiApplError ... 478
6.6.14 E_AcsiAutoRecSt ... 478
6.6.15 E_AcsiBeh ... 479
6.6.16 E_AcsiCBOpCap ... 479
6.6.17 E_AcsiCheck ... 479
6.6.18 E_AcsiClcIntvTyp ... 480
6.6.19 E_AcsiClcMod ... 480
6.6.20 E_AcsiClcMth ... 481
6.6.21 E_AcsiClcRfTyp ... 481
6.6.22 E_AcsiClcTotVA ... 482
6.6.23 E_AcsiCmdQual ... 482
6.6.24 E_AcsiCtlModels ... 482
6.6.25 E_AcsiCtlReport ... 483
6.6.26 E_AcsiCtlService ... 483
6.6.27 E_AcsiCycTrMod ... 484
6.6.28 E_AcsiDbpos ... 484
6.6.29 E_AcsiDir ... 484
6.6.30 E_AcsiDirMod ... 485
6.6.31 E_AcsiEEHealth ... 485
6.6.32 E_AcsiFailMod ... 486
6.6.33 E_AcsiFanCtl ... 486
6.6.34 E_AcsiFanCtlGen ... 486
6.6.35 E_AcsiFc ... 487
6.6.36 E_AcsiFilTyp ... 488
6.6.37 E_AcsiFltLoop ... 488
6.6.38 E_AcsiGnSt ... 489
6.6.39 E_AcsiHealth ... 489
6.6.40 E_AcsiHvRef ... 490
6.6.41 E_AcsiIntrDetMth ... 490
6.6.42 E_AcsiLevMod ... 491
6.6.43 E_AcsiLivDeaMod ... 491

6.6.44 E_AcsiLogicalNodeClass ... 491
6.6.45 E_AcsiMechHealth ... 492
6.6.46 E_AcsiMod ... 492
6.6.47 E_AcsiMonth ... 493
6.6.48 E_AcsiMultiplier ... 493
6.6.49 E_AcsiOccPer ... 494
6.6.50 E_AcsiOccType ... 495
6.6.51 E_AcsiOpModRect ... 495
6.6.52 E_AcsiOpModSyn ... 495
6.6.53 E_AcsiOrCategory ... 496
6.6.54 E_AcsiParColMod ... 496
6.6.55 E_AcsiParMod ... 497
6.6.56 E_AcsiParTraMod ... 497
6.6.57 E_AcsiPFSign ... 498
6.6.58 E_AcsiPhsRef ... 498
6.6.59 E_AcsiPhyHealth ... 498
6.6.60 E_AcsiPIDAlg ... 499
6.6.61 E_AcsiPmpCtl ... 499
6.6.62 E_AcsiPmpCtlGen ... 499
6.6.63 E_AcsiPolQty ... 500
6.6.64 E_AcsiPOWCap ... 500
6.6.65 E_AcsiQualitySource ... 501
6.6.66 E_AcsiQualityValidity ... 501
6.6.67 E_AcsiRange ... 501
6.6.68 E_AcsiRcdMod ... 502
6.6.69 E_AcsiReTrMod ... 502
6.6.70 E_AcsiRotDir ... 503
6.6.71 E_AcsiRstMod ... 503
6.6.72 E_AcsiSamplingMode ... 503
6.6.73 E_AcsiSboClasses ... 504
6.6.74 E_AcsiSeqT ... 504
6.6.75 E_AcsiServerDirectoryClass ... 504
6.6.76 E_AcsiServiceError ... 505
6.6.77 E_AcsiServiceType ... 505
6.6.78 E_AcsiSetCharact ... 506
6.6.79 E_AcsiSev ... 507
6.6.80 E_AcsiShOpCap ... 508
6.6.81 E_AcsiSIUnit ... 508
6.6.82 E_AcsiSptEndSt ... 510
6.6.83 E_AcsiStClcTun ... 511
6.6.84 E_AcsiStrWeekDay ... 512
6.6.85 E_AcsiSwOpCap ... 512
6.6.86 E_AcsiSwTyp ... 513
6.6.87 E_AcsiTcmd ... 513
6.6.88 E_AcsiTmSyn ... 513
6.6.89 E_AcsiTnkTyp ... 514

6.6.90 E_AcsiTpcRxMod ... 514
6.6.91 E_AcsiTpcTxMod ... 515
6.6.92 E_AcsiTrBeh ... 515
6.6.93 E_AcsiTrgMod ... 516
6.6.94 E_AcsiTrMod ... 516
6.6.95 E_AcsiTypRsCrv ... 516
6.6.96 E_AcsiUnbDetMth ... 517
6.6.97 E_AcsiUnBlkMod ... 517
6.6.98 E_AcsiVLanPriority ... 517
6.6.99 E_AcsiWeekDay ... 518
6.6.100 E_AcsiWeiMod ... 519
6.6.101 E_AsyncEnvironmentState ... 519
6.6.102 E_AsyncServiceState ... 519
6.6.103 E_EventCompletion ... 520
6.6.104 E_GseDispatchMode ... 520
6.6.105 E_GseDispatchMode ... 521
6.6.106 E_GseGoCBPublisherState ... 521
6.6.107 E_GseGoCBSubscriberState ... 521
6.6.108 E_GseLinkStatus ... 522
6.6.109 E_GseRetransmissionStrategy ... 522
6.6.110 E_GseSvCBState ... 523
6.6.111 E_GseSvGeneratorMode ... 523
6.6.112 E_GseSvSmpSynch ... 523
6.6.113 E_MmsVmdLogicalStatus ... 524
6.6.114 E_MmsVmdPhysicalStatus ... 524
6.6.115 E_Rfc1006TpduSize ... 524
6.6.116 E_ScsmEdition ... 525
6.6.117 E_ScsmLinkStatus ... 525
6.6.118 E_ScsmRCBState ... 525
6.6.119 E_TcpLinkStatus ... 526
6.6.120 E_TcpSocketType ... 526
6.6.121 E_TraceLevel ... 526
6.6.122 E_UtcTimeAccuracy ... 527
6.6.123 External data types ... 528
6.6.124 ST_AcsiAnalogueValue ... 531
6.6.125 ST_AcsiCalendarTime ... 532
6.6.126 ST_AcsiCell ... 532
6.6.127 ST_AcsiDataSetClassInfo ... 533
6.6.128 ST_AcsiFunctionalConstraints ... 533
6.6.129 ST_AcsiIEDGroupClassInfo ... 534
6.6.130 ST_AcsiIntelligentElectronicDeviceClassInfo ... 534
6.6.131 ST_AcsiLogicalDeviceClassInfo ... 535
6.6.132 ST_AcsiLogicalNodeClassInfo ... 535
6.6.133 ST_AcsiLogOptionalFields ... 536
6.6.134 ST_AcsiOptionalFields ... 536
6.6.135 ST_AcsiOptionalSvFields ... 537

6.6.136 ST_AcsiOriginator ... 538
6.6.137 ST_AcsiPhyComAddr ... 538
6.6.138 ST_AcsiPoint ... 538
6.6.139 ST_AcsiPresCond ... 539
6.6.140 ST_AcsiPulseConfig ... 540
6.6.141 ST_AcsiQuality ... 541
6.6.142 ST_AcsiRangeConfig ... 542
6.6.143 ST_AcsiReasonCode ... 542
6.6.144 ST_AcsiScaledValueConfig ... 543
6.6.145 ST_AcsiSvOptionalFields ... 543
6.6.146 ST_AcsiTriggerConditions ... 544
6.6.147 ST_AcsiUnit ... 544
6.6.148 ST_AcsiValWithTrans ... 545
6.6.149 ST_AcsiVector ... 545
6.6.150 ST_GseAdapterInfo ... 545
6.6.151 ST_GseGoCBPublisherTx ... 546
6.6.152 ST_GseGoCBSubscriberRx ... 547
6.6.153 ST_GseMsvCBPublisherTx ... 548
6.6.154 ST_GseMsvCBSubscriberRx ... 549
6.6.155 ST_ScsmAdditionalCauseDiagnosticEventInfo ... 549
6.6.156 ST_ScsmBrCBJobInfo ... 550
6.6.157 ST_ScsmBrCBValuesSet ... 550
6.6.158 ST_ScsmCommandTerminationEventInfo ... 551
6.6.159 ST_ScsmConfirmedEventInfo ... 551
6.6.160 ST_ScsmGoCBJobInfo ... 552
6.6.161 ST_ScsmGoCBValuesSet ... 552
6.6.162 ST_ScsmMsvCBValuesSet ... 553
6.6.163 ST_ScsmReportEventInfo ... 553
6.6.164 ST_ScsmUrCBJobInfo ... 554
6.6.165 ST_ScsmUrCBValuesSet ... 555
6.6.166 ST_ServiceErrorClass ... 555
6.6.167 T_AcsiCurrency ... 555
6.6.168 T_AcsiObjectName ... 555
6.6.169 T_AcsiObjectReference ... 555
6.6.170 T_AcsiTag ... 556
6.6.171 T_AcsiUnicode255 ... 556
6.6.172 T_AcsiVisString64 ... 556
6.6.173 T_AcsiVisString129 ... 556
6.6.174 T_AcsiVisString255 ... 556
6.6.175 T_BinaryTime ... 556
6.6.176 T_INT24 ... 557
6.6.177 T_INT128 ... 557
6.6.178 T_MmsInteger8 ... 557
6.6.179 T_MmsInteger16 ... 557
6.6.180 T_MmsInteger32 ... 557
6.6.181 T_MmsString ... 558

6.6.182 T_MmsVisibleString ... 558
6.6.183 T_OCTET3 ... 558
6.6.184 T_OCTET4 ... 558
6.6.185 T_OCTET6 ... 558
6.6.186 T_OCTET8 ... 558
6.6.187 T_OCTET16 ... 558
6.6.188 T_OCTET20 ... 559
6.6.189 T_OCTET32 ... 559
6.6.190 T_OCTET35 ... 559
6.6.191 T_OCTET64 ... 559
6.6.192 T_UINT24 ... 559
6.6.193 T_UINT128 ... 559
6.6.194 T_UtcTime ... 560
6.6.195 T_UtcTimeQuality ... 560
6.6.196 U_AcsiAnalogueCtlVal ... 561
6.6.197 U_AcsiCtlVal ... 561
6.6.198 U_AcsiEnumeratedCtlVal ... 562
6.6.199 U_AcsiQuality ... 562
6.7 Parameter lists ... 563
6.7.1 Param_Acsi ... 563
6.7.2 Param_Gse ... 569
6.7.3 Param_Sockets ... 569
6.8 Global variables ... 570
6.8.1 TC_EVENTS constants ... 570
6.8.2 TC_EVENT_CLASSES constants ... 656
6.8.3 GVL_Acsi constants ... 659

7 Examples ... 664
7.1 General Client - Project structure ... 664
7.2 Client - Base Sample Project ... 669
7.3 Client - Read Data (GetAllDataValues, GetDataValues) ... 670
7.4 Client - Write Data (SetAllDataValues, SetDataValues) ... 671
7.5 Client - Report Control Blocks (Unbuffered, Buffered) ... 672
7.6 Client - Direct Control with normal security ... 684
7.7 Client - SBO Control with normal security ... 685
7.8 Client - Direct Control with enhanced security ... 686
7.9 Client - SBO Control with enhanced security ... 687
7.10 Client - Read/Write DataSet Values (GetDataSetValues, SetDataSetValues) ... 689
7.11 Client - GOOSE Subscriber (GetGoCBDataValues, SetGoCBDataValues) ... 693
7.12 Client - GOOSE Subscriber (without Client-Server communication) ... 693
7.13 Client - User defined timestamp clock source ... 695
7.14 Client - Dynamic created DataSets (CreateDataSet, DeleteDataSet) ... 696
7.15 General Server - Project structure ... 700
7.16 Server - Base Sample Project ... 704
7.17 Server – GOOSE Publisher (GetGoCBDataValues, SetGoCBDataValues) ... 704
7.18 Server - GOOSE Publisher (without Client-Server communication) ... 704
7.19 Server - Direct Control with normal security ... 707

7.20 Server - Direct Control with enhanced security ... 710
7.21 Server - SBO Control with normal security ... 713
7.22 Server - SBO Control with enhanced security ... 718
7.23 Server - User defined timestamp clock source ... 724
7.24 Server - Unbuffered Reporting ... 725
7.25 Server - Data access events ... 728
7.26 Server - Buffered Reporting ... 729
7.27 Multiple GOOSE publishers/subscribers on one network adapter (without client-server communication) ... 734
7.28 TLS - Secure client-server communication using certificates or PSK (pre-shared key) ... 742
7.29 Authentication, negotiation of supported services and optional communication parameters ... 748
7.30 SV Publisher/Subscriber according to UCA 61850-9-2 Light Edition - Transmission of sampled measured values ... 760

8 Appendix ... 772
8.1 Binary-Time LWORD format specification ... 772
8.2 Binary-Time string format specification ... 772
8.3 UTC-Time LWORD format specification ... 773
8.4 UTC-Time-Quality BYTE format specification ... 775
8.5 UTC-Time-Quality string format specification ... 776
8.6 UTC-Time string format specification ... 777
8.7 Quality-String format specification ... 778
8.8 Quality-WORD format specification ... 779
8.9 FAQ - frequently asked questions and answers ... 780

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been compiled with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without notice.
Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.





EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.
The distribution and reproduction of this document, as well as the use and communication of its contents without express authorization, are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event that a patent, utility model, or design are registered.

Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: https://www.beckhoff.com/trademarks.

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

⚠ CAUTION
There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product



This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

2 Overview

IEC 61850 is a communication protocol for data exchange in electrical substations. The TwinCAT product for IEC 61850 offers this type of communication directly from the TwinCAT control system in the form of a PLC programming interface. The entire communication stack is mapped in the PLC. The products can also be used on the protocols based on IEC 61850. This is particularly true in the field of wind turbines for the data models of IEC 61400-25. In addition to the PLC programming interface, there is also a configurator for the data model. It can read and generate the IEC 61850 icd files. It can also automatically generate the necessary PLC code for the mapping of the data model, allowing the user to concentrate on the actual application. In addition to the client/server communication, TwinCAT also supports the GOOSE protocol, which is based directly on the Ethernet layer.

Product components

The product TF6510 IEC 61850 consists of the following components:

- **PLC libraries:**
  Tc3_IEC61850.compiled-library,
  Tc3_IEC61850_8_1.compiled-library,
  Tc3_Gse.compiled-library,
  Tc3_Mms.compiled-library,
  Tc3_Acsi.compiled-library,
  Tc3_Ber.compiled-library,
  Tc3_Rfc1006.compiled-library,
  Tc3_Collections.compiled-library,
  Tc3_Tpkt.compiled-library,
  Tc3_Ulosi.compiled-library,
- **Other products:** TF6310 TCP/IP Server
- **TwinCAT Telecontrol Configurator:** Configurator for the IEC 61850 / IEC 61400-25 data model

3 Installation

3.1 System requirements

Development environment

A pure development environment describes a computer on which PLC programs are developed but not executed. The following components must be installed on a development computer:

- TwinCAT 3.1.4024.0 XAE (Engineering) or higher
- TwinCAT 3 function TF6510
- A 7-day trial license can be used (multiple times, if required) for the development environment (see Licensing [▶ 19]).

Runtime environment

A runtime environment describes a computer on which PLC programs are executed. The following components must be installed on a runtime computer:

- TwinCAT 3.1.4024.0 XAR (Runtime) or higher
- TwinCAT 3 function TF6510
- Licenses for TC1200 PLC and TF6510
- A 7-day trial license can be used for testing purposes (see Licensing [▶ 19])

Development and runtime environment on one computer

If a computer is to be used as a runtime and development environment (to test a PLC program before it is loaded into the runtime environment, for example), the following requirements must be met:

- TwinCAT 3.1.4024.0 XAE (Engineering) or higher
- TwinCAT 3 function TF6510
- Licenses for TC1200 PLC and TF6510
- A 7-day trial license can be used for testing purposes (see Licensing [▶ 19])

3.2 Installation

Depending on the TwinCAT version and operating system used, this TwinCAT 3 Function can be installed in different ways, which are described in more detail below.

Update installation

An update installation always uninstalls the previous installation.

- Create a backup of your configuration files before the update.

TwinCAT 3 Package Manager: Installation (TwinCAT 3.1 Build 4026)

If you are using TwinCAT 3.1 Build 4026 (and higher) on the Microsoft Windows operating system, install this function via the TwinCAT Package Manager (TcPkg). Detailed instructions on installing products can be found in the chapter Managing TwinCAT Software in the TwinCAT 3.1 Build 4026 installation instructions.

NOTICE

Unprepared TwinCAT restart can cause data loss

The installation of this function may result in a TwinCAT restart.

- Make sure that no critical TwinCAT applications are running on the system or shut them down in an orderly manner first.

TwinCAT Package Manager UI

Install the following workload via the TcPkg **U**ser **I**nterface (UI) to use the product:

- **TF6510 | TwinCAT 3 IEC-61850/IEC-61400-25**

TwinCAT Package Manager CLI

You can use the TcPkg Command Line Interface (CLI) to display the available workloads on the system:

```
tcpkg list -t workload
```

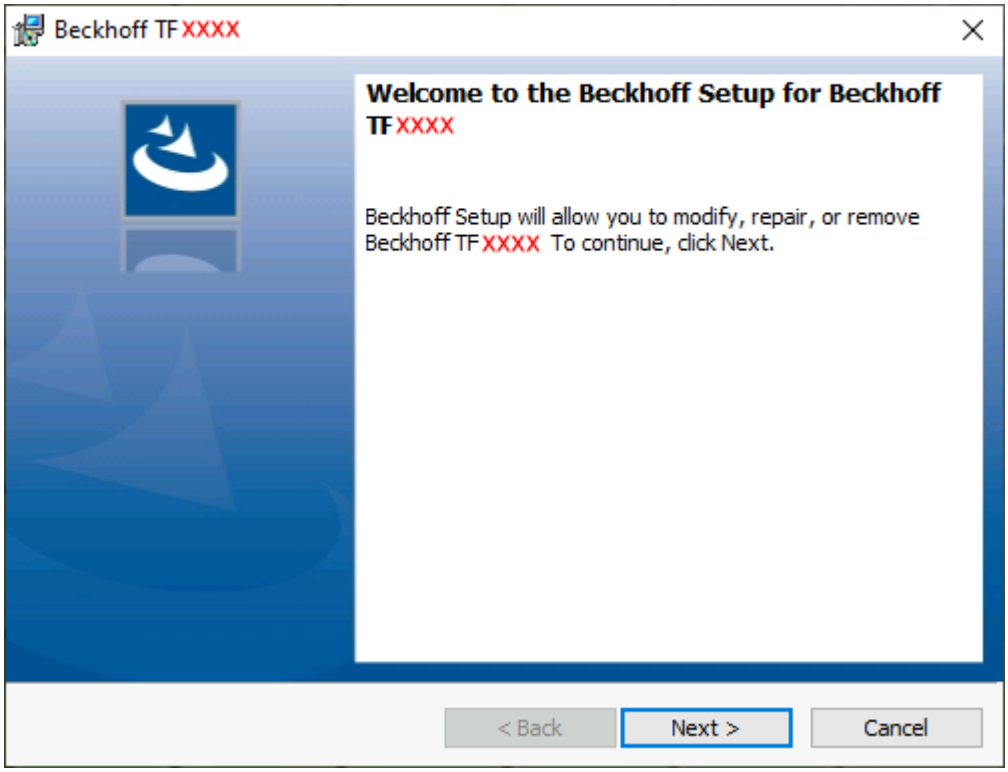
Use the following command to install the Workload:

```
tcpkg install TF6510.IEC61850-IEC61400-25.XAE
```

TwinCAT 3 setup: Installation (TwinCAT 3.1 Build 4024 and earlier)

If you are using TwinCAT 3.1 Build 4024 on the Microsoft Windows operating system, install this function via a setup package, which you can download from the Beckhoff website at https://www.beckhoff.com/download.

Depending on the system on which you need the function, the installation can be done on either the engineering or runtime side. The following screenshot is an example of the Setup interface.



To complete the installation process, follow the instructions in the Setup dialog.

NOTICE

Unprepared TwinCAT restart can cause data loss

The installation of this function may result in a TwinCAT restart.

- Make sure that no critical TwinCAT applications are running on the system or shut them down in an orderly manner first.

Windows CE: Installation

If you are using Microsoft Windows CE as your operating system, install this function via the respective CAB files, which you can download as separate files from the Beckhoff website at https://www.beckhoff.com/download.

After the download, the CAB files can be transferred to the Windows CE device via file transfer and executed there. The CAB files then install and register the function on the respective system.

Always use the appropriate CAB file for your system. Specifically, this means:

- CE-ARMV4I: Arm®-based devices, e.g. CX8190, CX9020
- CE-X86: x86-based devices, e.g. CX51xx, CX52xx, CX20xx

The CAB file can be transferred to the device either via the CF/SD card or the FTP server integrated in Windows CE.

Device restart

In order to use the function, the device must be restarted once the function has been installed.

3.3 Licensing

The TwinCAT 3 function can be activated as a full version or as a 7-day test version. Both license types can be activated via the TwinCAT 3 development environment (XAE).

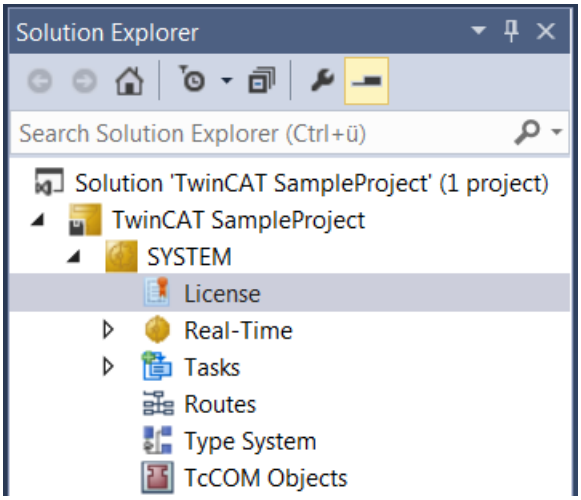
Licensing the full version of a TwinCAT 3 Function

A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

Licensing the 7-day test version of a TwinCAT 3 Function

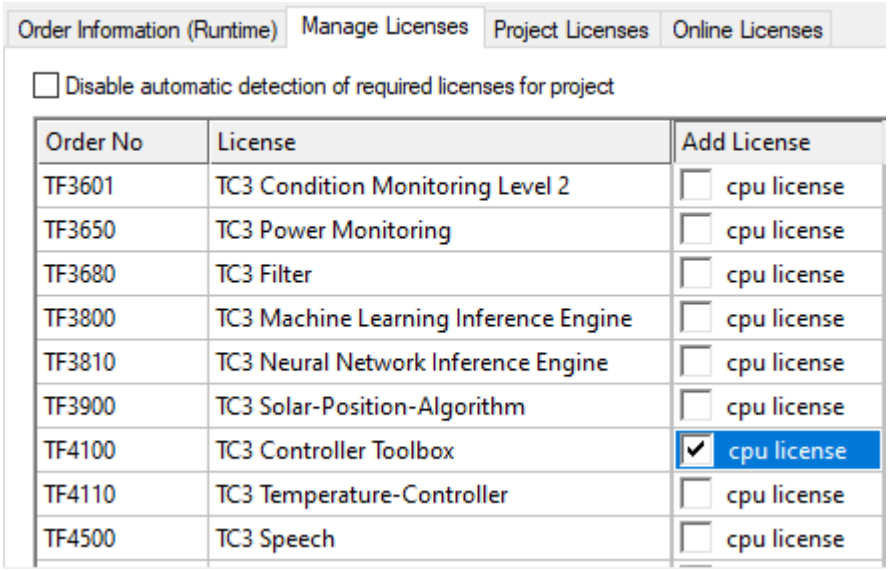
A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.
4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.

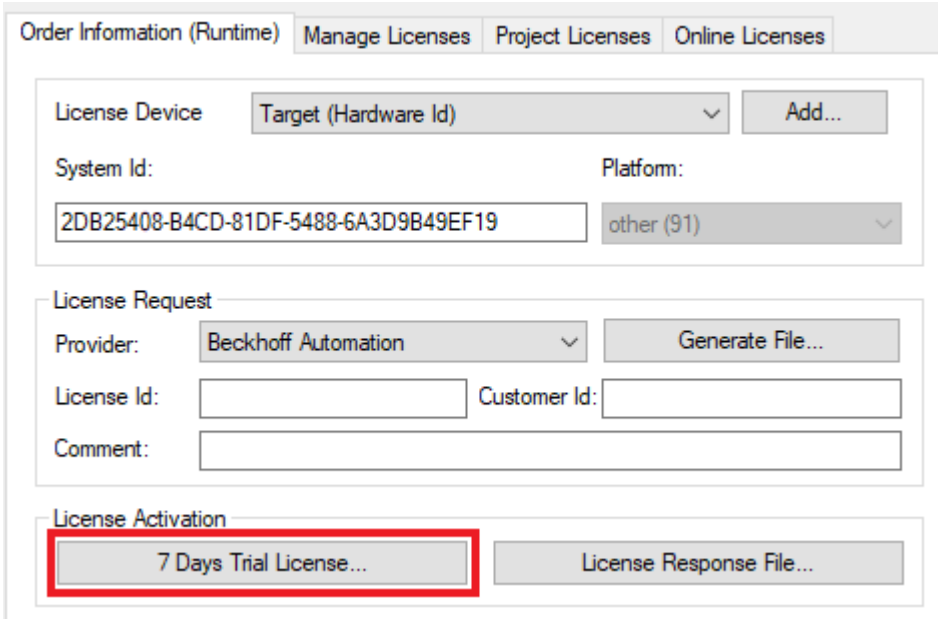


⇨ The TwinCAT 3 license manager opens.

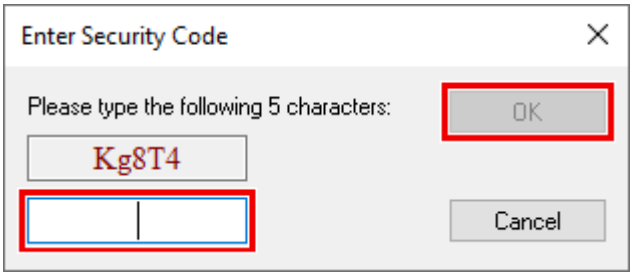
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").



6. Open the **Order Information (Runtime)** tab.
   ⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”**.**
7. Click **7-Day Trial License...** to activate the 7-day trial license.



   ⇨ A dialog box opens, prompting you to enter the security code displayed in the dialog.



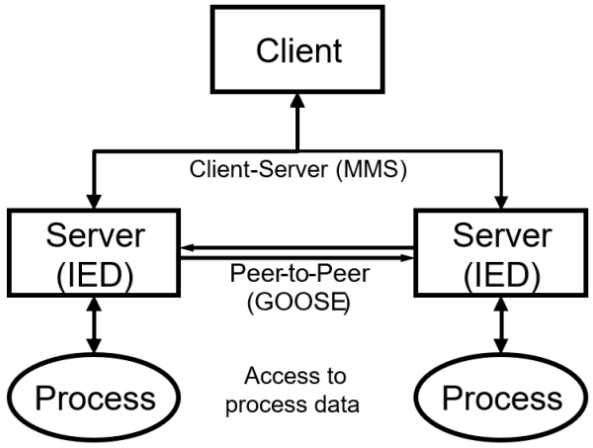
8. Enter the code exactly as it is displayed and confirm the entry.
9. Confirm the subsequent dialog, which indicates the successful activation.
   ⇨ In the tabular overview of licenses, the license status now indicates the expiry date of the license.

10. Restart the TwinCAT system.
⇨ The 7-day trial version is enabled.

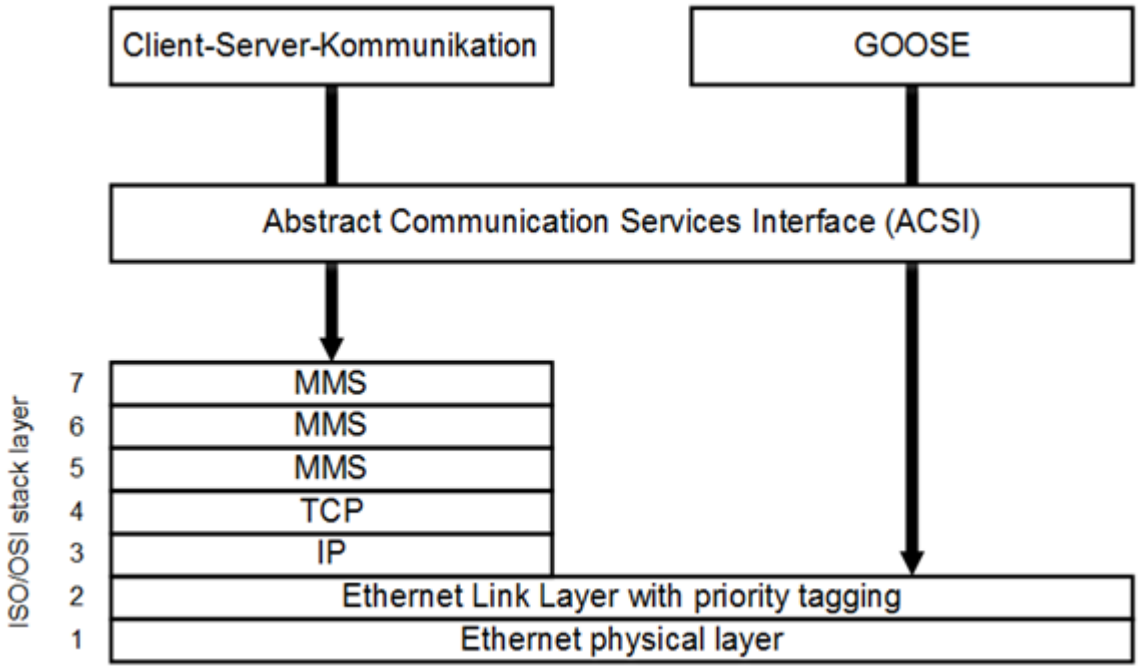
4 Technical introduction

4.1 Overview of standards

IEC 61850 is a general transmission protocol for the protective and control equipment in electrical substations in medium and high voltage systems (station automation). This communication is physically based on Ethernet technology. The server-client communication is currently supported in TwinCAT via MMS (Manufacturing Message Specification) and GOOSE.



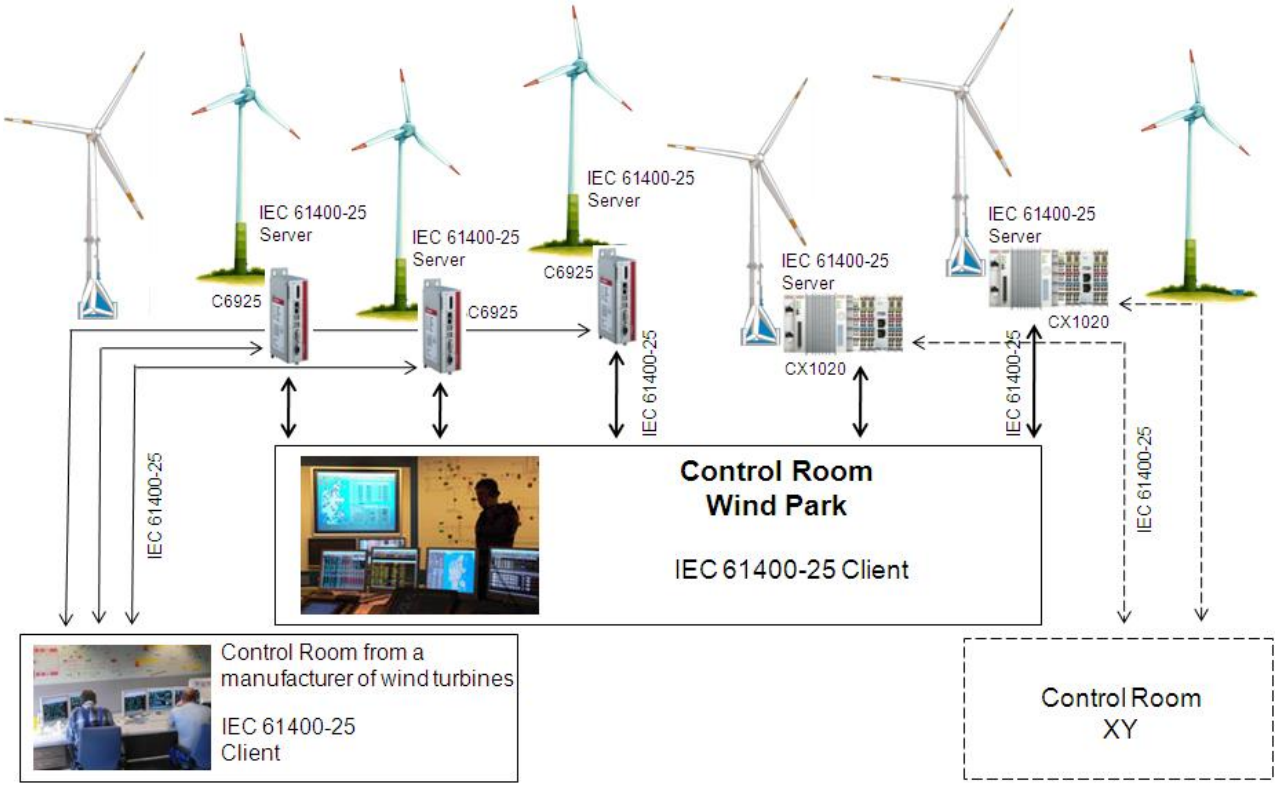
The communication stack, as shown in the illustration below, is implemented layer by layer in the TwinCAT PLC in the form of libraries.



IEC 61850, which was published in 2004, is strictly object-oriented and its data model is hierarchically structured. The various data models are explained in the following chapter.

Application example

In this case, the use of IEC 61400-25 enables all wind turbines from different manufacturers in a wind farm to communicate with a central station.
Standardization avoids the use of vendor-specific protocols, which would lead to increased application expenditure. BECKHOFF supports this method with the implementation of IEC 61850 or IEC 61400-25 respectively in the TwinCAT Automation suite.

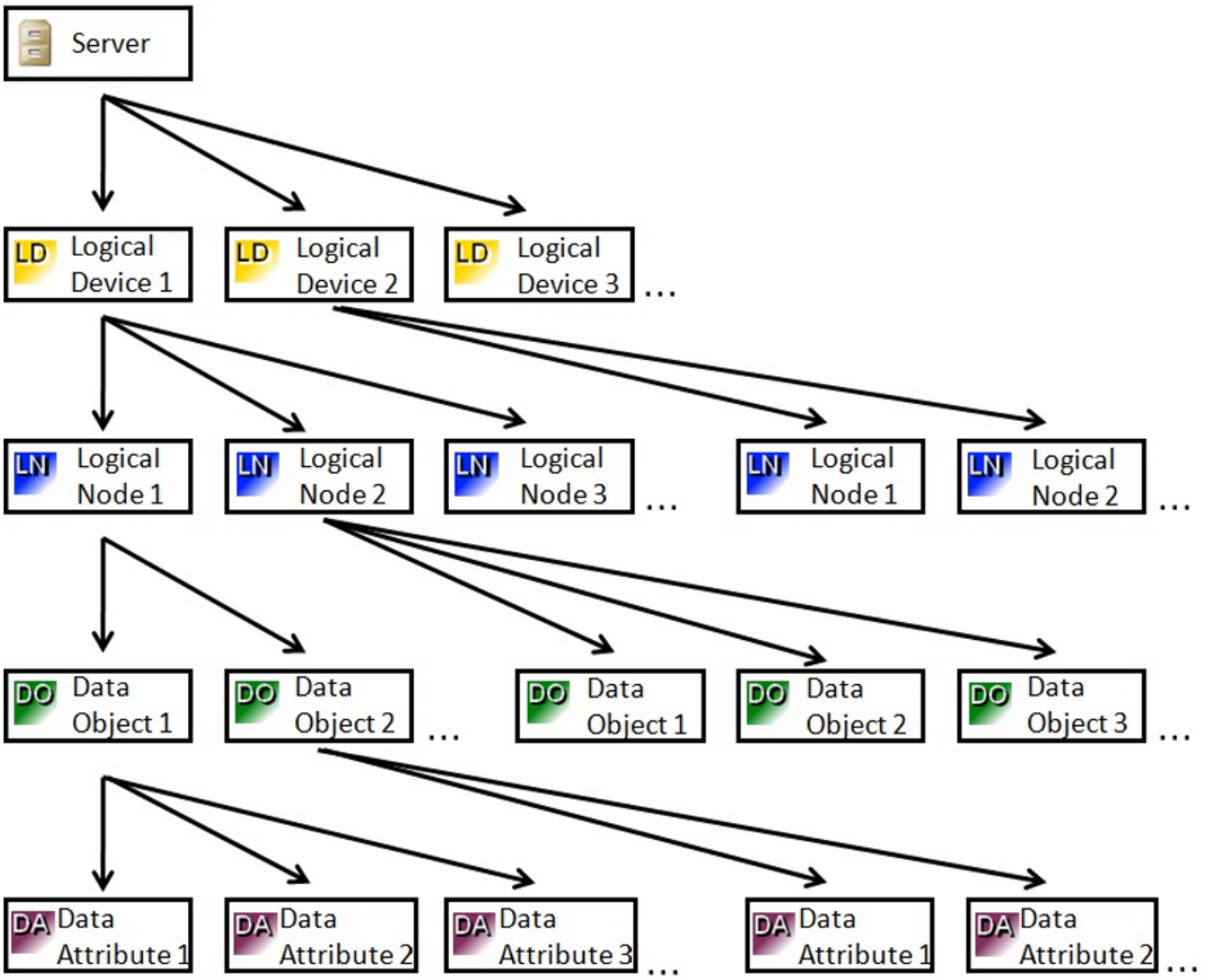


4.2 Data model

IEC 61850 is the basic standard for various application specializations. Each specialization defines its own data model or extends the basic data model of IEC 61850.

Standard	Data models
IEC 61850	Basic data models for protective and control equipment in electrical substations
IEC 61850-7-410 (former: IEC 62344)	Data models for communication in hydroelectric power plants
IEC 61850-7-420 (former: IEC 62350)	Data models for distributed/decentralized power generation and storage
IEC 61400-25	Data models for the communication and monitoring of wind turbines

In principle, the data models are divided into five hierarchical levels. The corresponding standards specify all necessary Logical Nodes, Data Objects and Data Attributes. Depending on the application, Logical Devices or server/client must be configured on the basis of the defined models. In the associated Beckhoff product, the configuration can be done in the TwinCAT Telecontrol Configurator [▶ 42] tool.


Server
LD Logical Device 1
LD Logical Device 2
LD Logical Device 3
...
LN Logical Node 1
LN Logical Node 2
LN Logical Node 3
...
LN Logical Node 1
LN Logical Node 2
...
DO Data Object 1
DO Data Object 2
...
DO Data Object 1
DO Data Object 2
DO Data Object 3
...
DA Data Attribute 1
DA Data Attribute 2
DA Data Attribute 3
...
DA Data Attribute 1
DA Data Attribute 2
...

Hierarchical levels

 **Server**	The server forms the first hierarchical level in the data model of IEC 61850 or IEC 61400-25 respectively. The server offers a connection point for a device, allowing corresponding Ethernet-based communication systems to be connected. In this case TwinCAT, with the libraries belonging to the protocol, would represent such a server on an Industrial PC or embedded device.
 **Logical Device**	The Logical Devices form the second hierarchical level in the data model of IEC 61850 and IEC 61400-25. This level subdivides a single physical device into several separate parts known as Logical Devices. The advantage of this subdivision is that functions or objects that belong together can be placed together on account of their commonalities and, for example, switched to a different operation mode. Different Logical Nodes from the third hierarchical level can be implemented, depending on the device.
 **Logical Node**	The Logical Nodes form the second hierarchical level in the data model of IEC 61850 and IEC 61400-25. They represent the information of all conceivable sub-functions that can occur in digital station control technology and, among other things, in wind turbines. These include all protective functions as well as automation functions and functions for the processing of measured values and counter values. For all standardized IEC 61850 and IEC 61400-25 Logical Nodes there are defined identifiers that always consist of an abbreviation with four letters. For example, the abbreviation *XSWI* stands for *Circuit Switch*. A Logical Device may consist of several such nodes.
 **Data Object**	A Logical Node can consist of several Data Objects, which form the fourth hierarchical level of the IEC 61850 or IEC 61400-25 data model. A special feature of this hierarchical level is that data objects in turn can be hierarchically nested. For instance, a data object can consist of several sub-data objects before a basic or complex data type is ultimately accessed in a Common Data Class (CDC).
 **Data Attribute**	The data attributes form the lowest hierarchical level of IEC 61850 and IEC 61400-25. They represent the detailed information or values of the data objects. Since the same number of data attributes is always defined for many data objects, the attributes of an object are combined in so-called Common Data Classes (CDC). The associated CDC is specified for each data object; the data objects are therefore specializations of the CDC.

4.3 Services supported

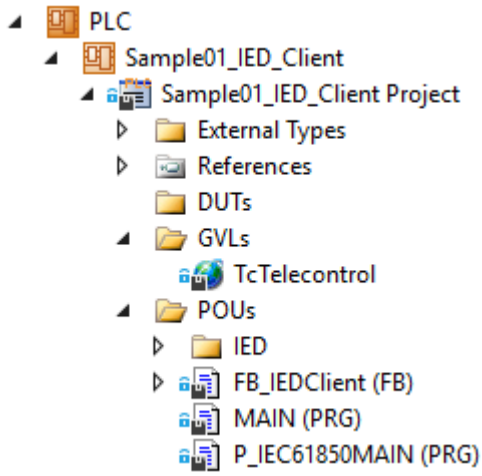
In general, the TF6510 supports server and client modes, as well as GOOSE in accordance with IEC 61850. Here you will find an overview of the IEC 61850 services supported by TwinCAT.

Service model	Supported services
Server (Device)	GetServerDirectory
Application association	Associate Abort Release
Logical device	GetLogicalDeviceDirectory
Logical Node	GetLogicalNodeDirectory GetAllDataValues
Data	GetDataValues SetDataValues GetDataDefinition GetDataDirectory
Data set	GetDataSetValues SetDataSetValues GetDataSetDirectory CreateDataSet DeleteDataSet
Reporting	Buffered RCB: Report GetBRCBValues SetBRCBValues Unbuffered RCB: Report GetURCBValues SetURCBValues
Generic substation events (GSE)	GOOSE support for publisher and subscriber GOOSE CB: SendGOOSEMessage GetGoCBValues SetGOCBValues
Control	Select SelectWithValue Cancel Operate CommandTermination TimeActivatedOperate
Transmission of sampled values (SV)	MSV support for publisher and subscriber MSV CB: SendMSVMessage GetMSVCBValues SetMSVCBValues

4.4 General Client - Project structure

Each of the client sample projects follows a general structure, which this document aims to explain. This structure includes the nesting of the function blocks in the Solution Explorer and the structure of the state machine in the client function block, which establishes the connection to the server and implements the data exchange. The basic structure of the TwinCAT IEC 61850 sample projects is based on the PLC project structure automatically generated by the TwinCAT Telecontrol Configurator. The TwinCAT Telecontrol Configurator in turn uses the "TwinCAT XAE Project (XML format)" as a template when generating the solution. The data models communicated in the samples reflect different IEC 61850 servers. They differ from sample to sample and are enclosed with the source code as icd files. The ICD files can also be used by third-party software for the simulation of a server.

First of all, a brief examination of the structure of a TwinCAT IEC 61850 PLC client project:



The generated solution name (unless specified otherwise) corresponds to the TwinCAT Telecontrol Configurator project name. The automatically generated TwinCAT PLC project name (unless specified otherwise) on the other hand has the following structure: "[Project name]_[IEDName]_Client".

As standard, every sample project has a "DUTs", a "GVLs" and a "POUs" folder. A Global Variable List (GVL) with the name "TcTelecontrol" is stored in the "GVLs" folder. The following function blocks are instantiated and initialized in this Global Variable List (see source code below):

- A Client function block instance of the type: FB_[IEDName]Client (connection management and data exchange).
- An IED data model function block instance of the type: FB_IED_[IEDName].
- Optional (depending on the GOOSE Subscriber configuration): one or more function block instances of the type: "FB_[IEDName]Gse" for GOOSE communication and GSE management.

In addition, the code version used during code generation and the version of the TwinCAT Telecontrol Configurator used are also stored there.

Namespace: TcTelecontrol

Type: Global Variable List (GVL)

```
VAR_GLOBAL
    ipCreator         : I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=2).SetGui
Ver(major:=1, minor:=0, build:=93, revision:=10);
    fb[IEDName]       : FB_IED_[IEDName];
    fb[IEDName]Client : FB_[IEDName]Client := (fbConnection:=(ipIED:=fb[IEDName],settings:=(sRemoteH
ost:='127.0.0.1')));
    fb[IEDName]Gse    : FB_[IEDName]Gse := (fbAdapter:=(ipIED:=fb[IEDName], settings:=(sMulticastAdd
r:='01-0C-CD-01-00-00', eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
END_VAR
```

In the "POUs" folder there is a further folder: "[IEDName]", which contains the entire hierarchical structure of the IED data model as function blocks. This folder also contains the IED function block that is instanced in the global variable list and has already been mentioned. The Client function block: "FB_[IEDName]Client", which implements the establishment of the connection and the data exchange with an IEC 61850 Server, is located on the same level. The TwinCAT PLC project contains a "MAIN" program as standard. This is called cyclically by a TwinCAT task and in turn calls the program "P_IEC61850MAIN". The program

"P_IEC61850MAIN" encapsulates the call of the client function block and of the optional GSE function block, separates the IEC 61850 communication from the remainder of the PLC machine program and helps, for example, with the implementation of further clients.

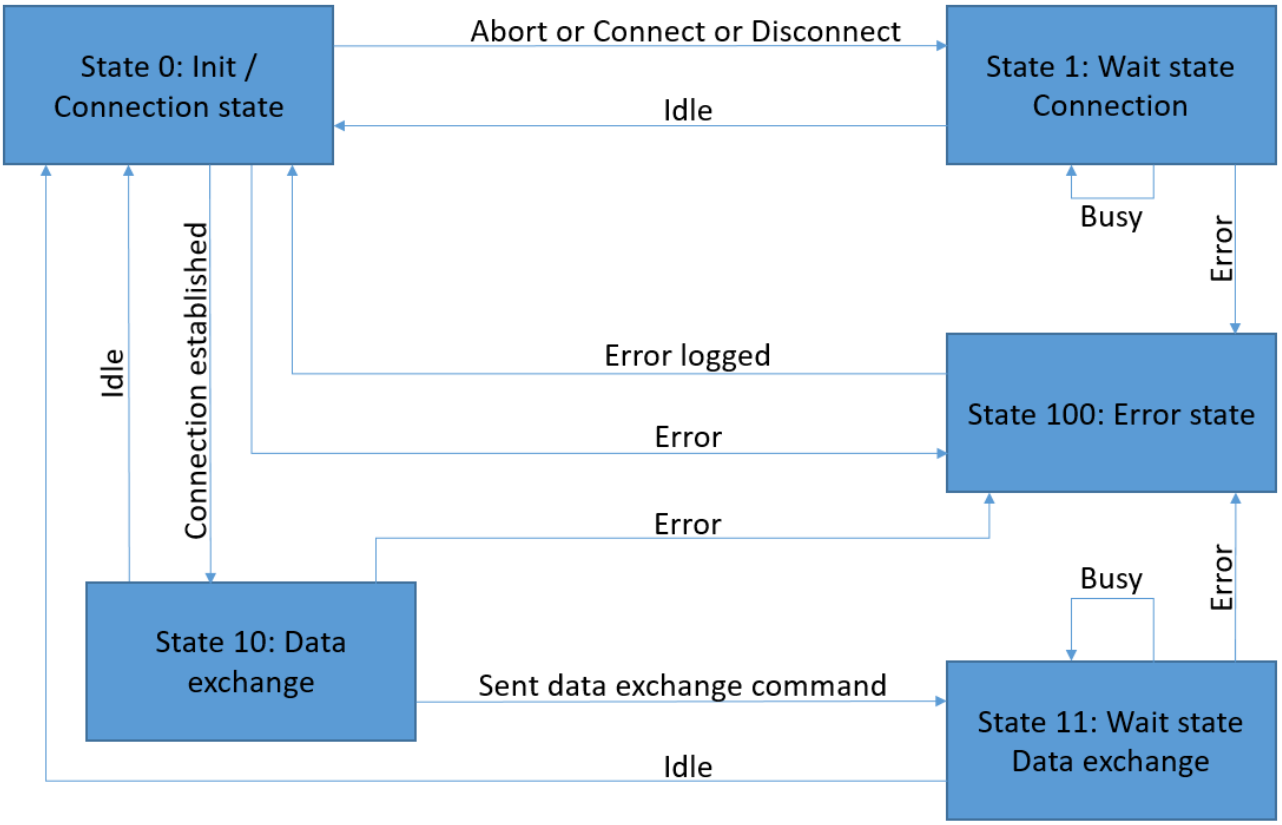
```
PROGRAM MAIN
VAR
END_VAR
```

```
P_IEC61850MAIN();
```

```
PROGRAM P_IEC61850MAIN
VAR
END_VAR
```

```
fb[IEDName]Client();
fb[IEDName]Gse();
```

In the FB_[IEDName]Client function block, there is a state machine, the basic states of which are used in every client sample. These states are graphically illustrated in the following diagram:



State 0 (Init state): the state machine is in this state as soon as the PLC program has been started. Commands for the management of the client/server connection are handled here (and in State 1). This is mainly controlled via four Boolean variables. When set, these variables then activate the corresponding commands (in this case these are once-only method calls at the client function block).

- _bAbort: calls the method "AbortReq", which activates the command to abort the client connection to the server.
- _bConnect: calls the method "AssociateReq", which activates the command to establish a new client connection to the server.
- _bDisconnect: calls the method "ReleaseReq", which activates the command for the controlled release of an existing client connection to the server.
- _bReconnect: also calls the method "AssociateReq" if the Client connection to the Server has been aborted/disconnected, but is to be restored automatically.

The methods listed above, which are called once only in this state, require longer than one PLC cycle for their execution. For this reason the state machine switches to a wait state (State 1), in which the termination of the activated command is awaited.

If the client connection to the server has already been established beforehand, the state machine will switch to the data transmission state (State 10).

State 1 (Wait State): in this state, the client waits until the command processing for the management of the client/server connection is no longer busy. As long as the connection is established, terminated or aborted, the state machine is in state 1. As soon as the command has been successfully processed, the state machine is returned to state 0 (Init state).

State 10 (Data exchange): if the state machine is in this state, then the client connection to the server has already been successfully established. The client is ready for the data transmission to the server. During the data transmission, commands are activated for the transmission or reception of the data.

In this state the sample projects differ from one another. Different methods or auxiliary function blocks are called here, depending on the desired functionality or logic in the application. In addition, the client function block can be extended by further states.

All method calls that activate commands for data transmission require several PLC cycles for successful execution, therefore the state machine must be set to State 11 (Wait state) after calling such a method.

In the case of an active connection and no data transmission, the state machine switches between States 0 and 10. The state machine is reset to State 0 in order to react to changes in the client-server connection status and to handle them in State 0.

State 11 (Wait state): this state is a further wait state. As soon as a data transmission command (activated in State 10) has been executed, the state machine is set to State 11 and waits until the command execution is no longer busy. The state machine then switches to State 0.

State 100 (Error state): as soon as an error occurs during the activation or processing of a command, the state machine is set to State 100. The error is logged here and the state machine reset to State 0.

```
FUNCTION_BLOCK FB_[IEDName]Client
VAR_INPUT
    fbConnection              : FB_iec61850ClientClass;
END_VAR
VAR
    _bAbort                   : BOOL;
    _bDisconnect              : BOOL;
    _bConnect                 : BOOL;
    _bReconnect               : BOOL := TRUE;
    _bReadAllData             : BOOL := TRUE;
    state                     : BYTE;
    eState                    : E_AsyncEnvironmentState;
    bBusy                     : BOOL;
    bSuccess                  : BOOL;
    ipResult                  : I_AsyncServiceResultClass;
    sLastErrorResult          : T_MaxString;
    fbAbortReason             : FB_ServiceErrorClass := (stError:=SUCCESS_EVENT);
    sLastAbortReason          : T_MaxString;
    nInvokeID                 : UDINT;
    eServiceError             : E_AcsiServiceError;
    nServiceError             : UDINT;
    nCmdError                 : UDINT;
    sObjReference             : T_AcsiObjectReference;
    sCtrlReference            : T_AcsiObjectReference;

    bGetServerDirectory       : BOOL := TRUE;
    bGetLogicalDeviceDirectory: BOOL := TRUE;
    bGetLogicalNodeDirectory  : BOOL := TRUE;
    bGetAllServerValues       : BOOL := TRUE;
    bGetAllServerValues       : BOOL := TRUE;
    bGetAllDataValues_LLN0_ST : BOOL := TRUE;
    bGetDataValues_LLN0_ST_Beh: BOOL := TRUE;
END_VAR
```

```
fbConnection.Execute();
eState:= fbConnection.eState;

CASE state OF
    0:
        IF _bAbort THEN
            _bAbort:= FALSE;
            bSuccess:= fbConnection.AbortReq(ipReason:=fbAbortReason, ipSink:=0, ipResult=>ipResult)
;
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Idle AND (_bConnect OR _bReconnect) THEN
            _bConnect:= FALSE;
            bGetAllServerValues:= SEL(_bReadAllData, bGetAllServerValues, TRUE);
            bSuccess:= fbConnection.AssociateReq(ipSink:=0, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 1);
```

```
        ELSIF eState = E_AsyncEnvironmentState.Established AND _bDisconnect THEN
            _bDisconnect:= FALSE;
            bSuccess:= fbConnection.ReleaseReq(ipSink:=0, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Established THEN
            state:= 10;
        END_IF
        _bConnect:= FALSE;
        _bDisconnect:= FALSE;
    1:
        IF ipResult <> 0 THEN
            ipResult.Execute();
            IF NOT (bBusy:=ipResult.IsBusy()) THEN
                state:= SEL(ipResult.IsCompleted(), 100, 0);
            END_IF
        END_IF
    10:
        IF bGetServerDirectory THEN
            bGetServerDirectory:= FALSE;
            bSuccess:= fbConnection.GetServerDirectoryReq(ipServer:=fb[IEDName], eClass:=E_AcsiServe
rDirectoryClass.LogicalDevice, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetLogicalDeviceDirectory THEN
            bGetLogicalDeviceDirectory:= FALSE;
            bSuccess:= fbConnection.GetLogicalDeviceDirectoryReq(ipLogicalDevice:=fb[IEDName].IEDLD1
, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetLogicalNodeDirectory THEN
            bGetLogicalNodeDirectory:= FALSE;
            bSuccess:= fbConnection.GetLogicalNodeDirectoryReq(ipLogicalNode:=fb[IEDName].IEDLD1.LLN
0, eClass:=E_AcsiLogicalNodeClass.DataSet, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipRe
sult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetAllServerValues THEN
            bGetAllServerValues:= FALSE;
            bSuccess:= fbConnection.GetAllServerValuesReq(ipServer:=fb[IEDName], hUser:=0, ipSink:=0
, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetAllDataValues_LLN0_ST THEN
            bGetAllDataValues_LLN0_ST:= FALSE;
            bSuccess:= fbConnection.GetAllDataValuesReq(ipLogicalNode:=fb[IEDName].IEDLD1.LLN0, eFc:
=E_AcsiFc.ST_, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetDataValues_LLN0_ST_Beh THEN
            bGetDataValues_LLN0_ST_Beh:= FALSE;
            bSuccess:= fbConnection.GetDataValuesReq(ipData:=fb[IEDName].IEDLD1.LLN0.Beh, eFc:=E_Acs
iFc.ST_, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSE
            state:= 0;
        END_IF
    11:
        IF ipResult <> 0 THEN
            ipResult.Execute();
            IF NOT (bBusy:=ipResult.IsBusy()) THEN
                state:= SEL(ipResult.IsCompleted(), 100, 0);
            END_IF
        END_IF
    100:
        state:= 0;
        IF ipResult <> 0 THEN
            nCmdError:= nCmdError + 1;
            sLastErrorResult:= ipResult.Dump();
        END_IF
END_CASE
```

GOOSE Subscriber (optional)

TwinCAT Telecontrol Configurator can also generate the PLC code for a GOOSE subscriber in a client project during PLC code generation (see code sample below). This requires that the user has previously created the GOOSE components such as GoCBs (goose control blocks) in the TwinCAT Telecontrol Configurator or imported them from an SCL file (e.g. ICD file).
By default, a function block with the name: "FB_[IEDName]Gse" is instantiated during code generation and added to the Global Variable List "TcTelecontrol". This function block establishes the connection between a network adapter of the TwinCAT control computer, the IED data model and the GOOSE configuration in the GoCBs. The GoCBs are instantiated in the IED data model (usually in LLN0). Each GoCB has a function block subelement with the name: "Subscriber". The subscription can be enabled or disabled from the PLC

code via the "Subscriber" method calls. By default, the subscription is enabled for all GoCBs when the PLC program is started. This is controlled by the "bSubscriber" variable initialized with "TRUE". A rising edge at the "bUnsubscribe" variable can be used to disable the subscription for all GoCBs. Subscriber commands issued through these methods are executed immediately, without wait cycles or further states required to complete command processing.
The "Subscriber" described here reads the configuration and updates the status of the GoCB (attribute "GoEna" is set to "TRUE" or "FALSE", for example), but it does not use the client-server services such as "SetGoCBValues" or "GetGoCBValues" to enable or disable the "Publisher" on the server side. This means that the generated code already implements a subscriber that can be enabled or disabled, for example, in the first PLC cycle or from the PLC code at any time. The required GoCB configuration settings (GoCB attribute values) can be made via initialization values. However, the GoCBs can already be configured in the TwinCAT Telecontrol Configurator. The initialization values are automatically generated and assigned during code generation. If the Subscriber has been activated and the configuration of the GoCB and the network adapter matches the receiving GOOSE frame, then the GOOSE data is copied (mapped) into the TwinCAT IED data model. The "Execute" method must be called cyclically the rest of the time. Among other things, it is responsible for updating the status information in the GoCB.

```
FUNCTION_BLOCK FB_[IEDName]Gse IMPLEMENTS I_GseLinkStatusEventSink
VAR_INPUT
    fbAdapter    : FB_GseAdapterClass := (ipLinkStatus:=THIS^);
END_VAR
VAR
    eLinkStatus  : E_GseLinkStatus;
    bSuccess     : BOOL;
    ipError      : I_ServiceErrorClass;
    bSubscribe   : BOOL := TRUE;
    bUnsubscribe : BOOL;
END_VAR
```

```
bSuccess:= fbAdapter.Execute(ipError=>ipError);
IF bSubscribe THEN
    bSubscribe:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Subscriber.Enable(ipAdapter:=fbAdapter, ipError=>ipErr
or);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Subscriber.Enable(ipAdapter:=fbAdapter, ipError=>ipErr
or);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Subscriber.Enable(ipAdapter:=fbAdapter, ipError=>ipErr
or);
ELSIF bUnsubscribe THEN
    bUnsubscribe:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Subscriber.Disable(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Subscriber.Disable(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Subscriber.Disable(ipError=>ipError);
ELSE
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Subscriber.Execute(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Subscriber.Execute(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Subscriber.Execute(ipError=>ipError);
END_IF
```

In the project tree under the I/O-Device branch you will find a network adapter instance named "GSE (RT Ethernet adapter)". This adapter instance must be configured accordingly, i.e. the I/O configuration must be adapted to the existing hardware and to the target platform on which the project is to run.
A new I/O configuration is also necessary if you change the target platform. This configuration must be done manually in TwinCAT XAE. In addition to the I/O configuration of the network adapter, a link must be established between the network adapter and the PLC function blocks for Goose communication. The link can be used to forward the data received from the network adapter to the instance of the function block: "FB_[IEDName]Gse". In the opposite direction the instance of the function block "FB_[IEDName]Gse" can forward the data to be sent to the network adapter.

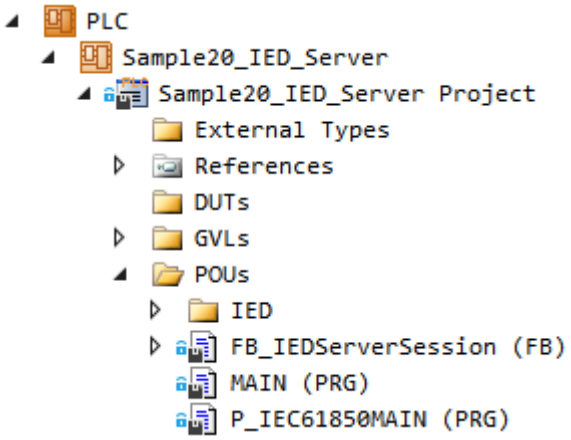
Here you can find more information: RT Ethernet adapter Configuration [▶ 35].

4.5 General Server - Project structure

All sample Server projects also follow a general structure, similar to the sample Client projects. In contrast to the client, the connection at the server is not actively established on the server side. After the program start, the server waits for a connection request from a client and accepts or rejects it. In addition, it must be possible for the server to establish connections to multiple clients.

This results in a different server project structure than for the client. The basic structure of the TwinCAT IEC 61850 Server sample projects is based on the PLC project structure automatically generated by the TwinCAT Telecontrol Configurator. The TwinCAT Telecontrol Configurator in turn uses the "TwinCAT XAE Project (XML format)" as a template when generating the solution. The data models communicated in the samples reflect different IEC 61850 servers. They differ from sample to sample and are enclosed with the source code as ICD files. The ICD files can also be used by third-party software for simulation purposes.

Structure of a TwinCAT IEC 61850 PLC Server project:



The generated solution name (unless specified otherwise) corresponds to the TwinCAT Telecontrol Configurator project name. The automatically generated TwinCAT PLC project name (unless specified otherwise) on the other hand has the following structure: "[Project name]_[IEDName]_Server".

As standard, every sample project has a "DUTs", a "GVLs" and a "POUs" folder. A Global Variable List (GVL) with the name "TcTelecontrol" is stored in the "GVLs" folder. The following function blocks are instantiated and initialized in this Global Variable List (see source code below):

- A server function block instance of the type: "FB_iec61850ServerClass" (server configuration settings and connection management).
- At least one instance of the server session function block of the type: "FB_[IEDName]ServerSession[1..n]" (connection management of a single connection and data exchange with a client). At this point, manually add additional instances of the Server Session function block if you want the server to communicate simultaneously with more than one client.
- An IED data model function block instance of the type: "FB_IED_[IEDName]".
- Optional (depending on the GOOSE publisher configuration): one or more function block instances of the type: "FB_[IEDName]Gse" for GOOSE communication and GSE management.

In addition, the code version used during code generation and the version of the TwinCAT Telecontrol Configurator used are also stored there.

Namespace: TcTelecontrol

Type: Global Variable List (GVL)

```
VAR_GLOBAL
    ipCreator                : I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=2
).SetGuiVer(major:=1, minor:=1, build:=94, revision:=1);
    fb[IEDName]              : FB_IED_[IEDName];
    fb[IEDName]Server        : FB_iec61850ServerClass := (ipIED:=fb[IEDName], settings:=(bEnable:=T
RUE, sLocalHost:='127.0.0.1'));
    fb[IEDName]ServerSession1 : FB_[IEDName]ServerSession := (fbConnection:=(ipServer:=fb[IEDName]Se
rver, settings:=(bEnable:=TRUE)));
    fb[IEDName]Gse           : FB_[IEDName]Gse := (fbAdapter:=(ipIED:=fb[IEDName], settings:=(sMult
icastAddr:='01-0C-CD-01-00-00', eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
END_VAR
```

In the "POUs" folder there is a further folder: "[IEDName]", which contains the entire hierarchical structure of the IED data model as function blocks. This folder also contains the IED data model function block that is instantiated in the Global Variable List and that has already been mentioned. The server session function block: "FB_[IEDName]ServerSession", which implements the establishment of the connection and the data exchange with an IEC 61850 Client, is located on the same level.

The TwinCAT PLC project contains a "MAIN" program as standard. This is called cyclically by a TwinCAT task and in turn calls the program "P_IEC61850MAIN". The program "P_IEC61850MAIN" encapsulates the call of the server, server session and the optional Gse function block, separates the IEC 61850 communication from the rest of the PLC machine program.

```
PROGRAM MAIN
VAR
END_VAR
```

```
P_IEC61850MAIN();
```

```
PROGRAM P_IEC61850MAIN
VAR
END_VAR
```

```
fb[IEDName]Server.Execute();
fb[IEDName]ServerSession1();
fb[IEDName]Gse();
```

In the FB_[IEDName]ServerSession function block, there is a state machine, the basic states of which are used in every server sample (see source code below). The server session function block is responsible for establishing connections and exchanging data with a single client. If there are multiple simultaneous client connections, multiple instances of this function block are required and instantiated.

State 0 (Init state): the state machine is in this state as soon as the PLC program has been started. Commands for the management of a client-server connection are handled here (and in State 1). Normally, the server session function block remains in this state and waits for a connection request from a client. The server then establishes a connection as soon as the connection request is detected. This happens automatically and does not require any further commands from the PLC application. If the server has established a connection to the client, the state machine switches to the data transmission state (State 10).

It is also possible to close an already established connection. This is controlled via two Boolean variables. When set, these variables enable the corresponding commands (in this case these are once-only method calls on the server session function block).

- _bAbort: calls the method "AbortReq", which activates the command to abort the server connection to the client.
- _bDisconnect: calls the method "ReleaseReq", which activates the command for the controlled release of an existing server connection to the client. As a rule, however, it is the client that should terminate its active connection to the server.

The methods listed above require longer than one PLC cycle for their execution. For this reason the state machine switches to a wait state (State 1), in which the termination of the activated command is awaited.

State 1 (Wait State): in this state, the server session function block waits until the command processing for the management of the server client connection is no longer busy. As long as the connection is being released or interrupted, the state machine is in State 1. The state machine is reset to State 0 (Init State) as soon as the command has been successfully executed.

State 10 (Data exchange): if the state machine is in this state, then the server connection to the client has already been successfully established. The server session function block is ready for data transmission to the server. The server data transmission to the client takes place automatically in the background and does not have to be initiated by the PLC application. It is the client that triggers a data transmission in the server by a request.

The state machine switches between states 0 and 10 when there is an active connection. The state machine is reset to State 0 in order to react to changes in the client-server connection status and to handle them in State 0.

State 100 (Error state): as soon as an error occurs during the activation or processing of a command, the state machine is set to State 100. The error is logged here and the state machine reset to State 0.

```
FUNCTION_BLOCK FB_[IEDName]ServerSession
VAR_INPUT
    fbConnection     : FB_iec61850ConnectionClass := (ipAbortInd:=THIS^, ipAssociateInd:=THIS^, ipRe
leaseInd:=THIS^);
END_VAR
VAR
    _bAbort          : BOOL;
    _bDisconnect     : BOOL;
    state            : BYTE;
    eState           : E_AsyncEnvironmentState;
```

```
    bBusy            : BOOL;
    bSuccess         : BOOL;
    ipResult         : I_AsyncServiceResultClass;
    sLastErrorResult : T_MaxString;
    fbAbortReason    : FB_ServiceErrorClass := (stError:=SUCCESS_EVENT);
    sLastAbortReason : T_MaxString;
END_VAR
```

```
fbConnection.Execute();
eState:= fbConnection.eState;

CASE state OF
    0:
        IF _bAbort THEN
            _bAbort:= FALSE;
            bSuccess:= fbConnection.AbortReq(ipReason:=fbAbortReason, ipSink:=0, ipResult=>ipResult)
;
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Established AND _bDisconnect THEN
            _bDisconnect:= FALSE;
            bSuccess:= fbConnection.ReleaseReq(ipSink:=0, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Established THEN
            state:= 10;
        END_IF
        _bDisconnect:= FALSE;
    1:
        IF ipResult <> 0 THEN
            ipResult.Execute();
            IF NOT (bBusy:=ipResult.IsBusy()) THEN
                state:= SEL(ipResult.IsCompleted(), 100, 0);
            END_IF
        END_IF
    10:
        state:= 0;
    100:
        state:= 0;
        IF ipResult <> 0 THEN
            sLastErrorResult:= ipResult.Dump();
        END_IF
END_CASE
```

GOOSE Publisher (optional)

TwinCAT Telecontrol Configurator can also generate the PLC code for a GOOSE publisher in a server project during PLC code generation (see code sample below). However, this is only possible if the user has previously created the GOOSE components such as GoCBs (goose control blocks) in the TwinCAT Telecontrol Configurator or imported them from an SCL file (e.g. ICD file). By default, a function block with the name: "FB_[IEDName]Gse" is instantiated during code generation and added to the Global Variable List "TcTelecontrol". This function block establishes the connection between a network adapter of the TwinCAT control computer, the IED data model and the GOOSE configuration in the GoCBs. The GoCBs are instantiated in the IED data model (usually in LLN0). Each GoCB has a function block subelement with the name: "Publisher". The "Publisher" method calls can be used to start or stop publishing from the PLC code. By default, publishing is started at PLC program start for all GoCBs. This is controlled by the "bStart" variable initialized with "TRUE". Publishing can be stopped for all GoCBs via a rising edge at the "bStop" variable. Publisher commands issued through these methods are executed immediately, without wait cycles or further states required to complete command processing. The "Publisher" described here reads the configuration and does update the status of the GoCB (attribute "GoEna" is set to "TRUE" or "FALSE", for example), but it does not use the client-server services such as "SetGoCBValues" or "GetGoCBValues" to start or stop the Publisher. This means that the generated code already implements a publisher that can be started or stopped, for example, in the first PLC cycle or from the PLC code at any time. The required GoCB configuration settings (GoCB attribute values) can be made via initialization values. However, the GoCBs can already be configured in the TwinCAT Telecontrol Configurator. The initialization values are then automatically generated and assigned during code generation. If the Publisher has been started and the configuration of the GoCB and the network adapter shows a match, then the Publisher immediately starts sending the first GOOSE frames (including frame repetitions). The "Update" method has a special significance. Every time it is called, the "Publisher" immediately sends a new GOOSE frame (and automatically new frame repetitions). This means that the application can set all relevant GOOSE dataset data in the IED data model first and then initiate the sending of the changed dataset data with the "Update" method call. The "Execute" method must be called cyclically the rest of the time. It is responsible for sending the frame repetitions and updating the status information in the GoCB.

```
FUNCTION_BLOCK FB_[IEDName]Gse IMPLEMENTS I_GseLinkStatusEventSink
VAR_INPUT
    fbAdapter   : FB_GseAdapterClass := (ipLinkStatus:=THIS^);
END_VAR
VAR
    eLinkStatus : E_GseLinkStatus;
    bSuccess    : BOOL;
    ipError     : I_ServiceErrorClass;
    bStart      : BOOL := TRUE;
    bStop       : BOOL;
    bUpdate     : BOOL;
END_VAR
```

```
bSuccess:= fbAdapter.Execute(ipError=>ipError);

IF bStart THEN
    bStart:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Publisher.Start(ipAdapter:=fbAdapter, ipError=>ipError
);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Publisher.Start(ipAdapter:=fbAdapter, ipError=>ipError
);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Publisher.Start(ipAdapter:=fbAdapter, ipError=>ipError
);
ELSIF bStop THEN
    bStop:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Publisher.Stop(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Publisher.Stop(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Publisher.Stop(ipError=>ipError);
ELSIF bUpdate THEN
    bUpdate:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Publisher.Update(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Publisher.Update(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Publisher.Update(ipError=>ipError);
ELSE
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Publisher.Execute(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Publisher.Execute(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Publisher.Execute(ipError=>ipError);
END_IF
```

The GSE block implements the "I_GseLinkStatusEventSink" interface. The method: "OnLinkStatusChange" belongs to this interface implementation and is called whenever the status of the network connection (at the network adapter) changes. The PLC application can, for example, query or check the network connection status via "eLinkStatus" variable.

```
METHOD OnLinkStatusChange
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    eStatus   : E_GseLinkStatus;
END_VAR
VAR
END_VAR
```

```
eLinkStatus:= eStatus;
```

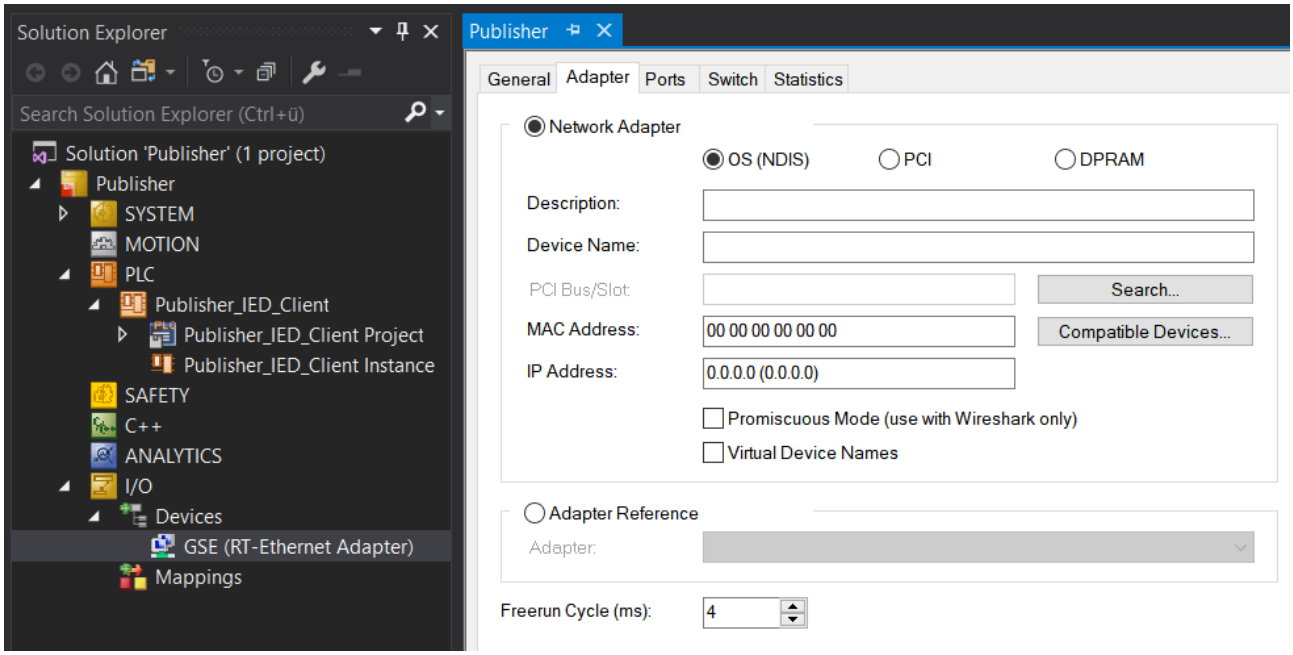
In the project tree under the I/O-Device branch you will find a network adapter instance named "GSE (RT Ethernet adapter)". This adapter instance must be configured accordingly, i.e. the I/O configuration must be adapted to the existing hardware and to the target platform on which the project is to run.
A new I/O configuration is also necessary if you change the target platform. This configuration must be done manually in TwinCAT XAE. In addition to the I/O configuration of the network adapter, a link must be established between the network adapter and the PLC function blocks for Goose communication. The link can be used to forward the data received from the network adapter to the instance of the function block: "FB_[IEDName]Gse". In the opposite direction the instance of the function block "FB_[IEDName]Gse" can forward the data to be sent to the network adapter.

Here you can find more information: RT Ethernet adapter Configuration [▶ 35].

4.6 RT Ethernet adapter configuration

The TwinCAT Telecontrol Configurator generates the PLC standard code for the "Publisher" or "Subscriber" whenever corresponding Goose or SV control block instances have been configured or imported from an ICD file. In order for these instances to be able to communicate with the outside world, a network adapter (RT Ethernet adapter) is required that can send (Publisher) or receive (Subscriber) the Goose or SV messages. The TwinCAT Telecontrol Configurator automatically creates such an adapter instance of the type "Real-Time Ethernet Adapter (Multiple Protocol Handler)" during project generation. It can be found later

in the project tree under the I/O Devices branch. This adapter instance must be configured accordingly, i.e. the I/O configuration must be adapted to the existing hardware and to the target platform on which the project is to run.
A new I/O configuration is also necessary if you change the target platform. This configuration must be done manually in TwinCAT XAE. The following image shows a network adapter instance created by the TwinCAT Telecontrol Configurator that has not yet been configured.
In order to use the real-time network adapter and thus the GOOSE or SV communication, the device used must have a compatible network adapter. The list of supported network chips can be found here: Supported network controllers.

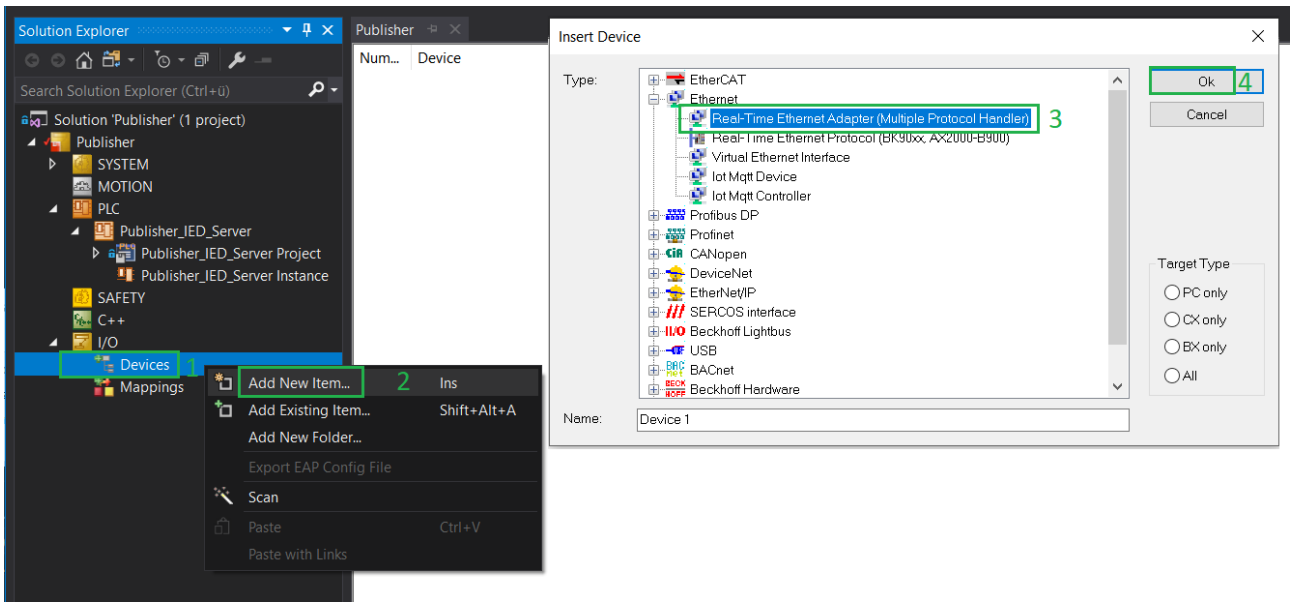


In addition to the I/O configuration of the network adapter, a link must be established between the network adapter and the PLC function blocks for Goose or SV communication. This link can be used to forward the data received from the network adapter to the instance of the Subscriber function block. In the opposite direction, the instance of the Publisher function block can forward the data to be sent to the network adapter. In Goose communication, for example, these are the PLC function block instances of "FB_{IEDName}Gse". See below for details.

Adding an adapter manually

If the network adapter is not yet present below the I/O configuration, it can be added later.

1. In the project tree, right-click **Devices.**
2. Select the context menu entry **Add New Item…**.
3. From the device list **Ethernet** select **Real-Time Ethernet Adapter (Multiple Protocol Handler)**.
4. Confirm your selection with **Ok**.

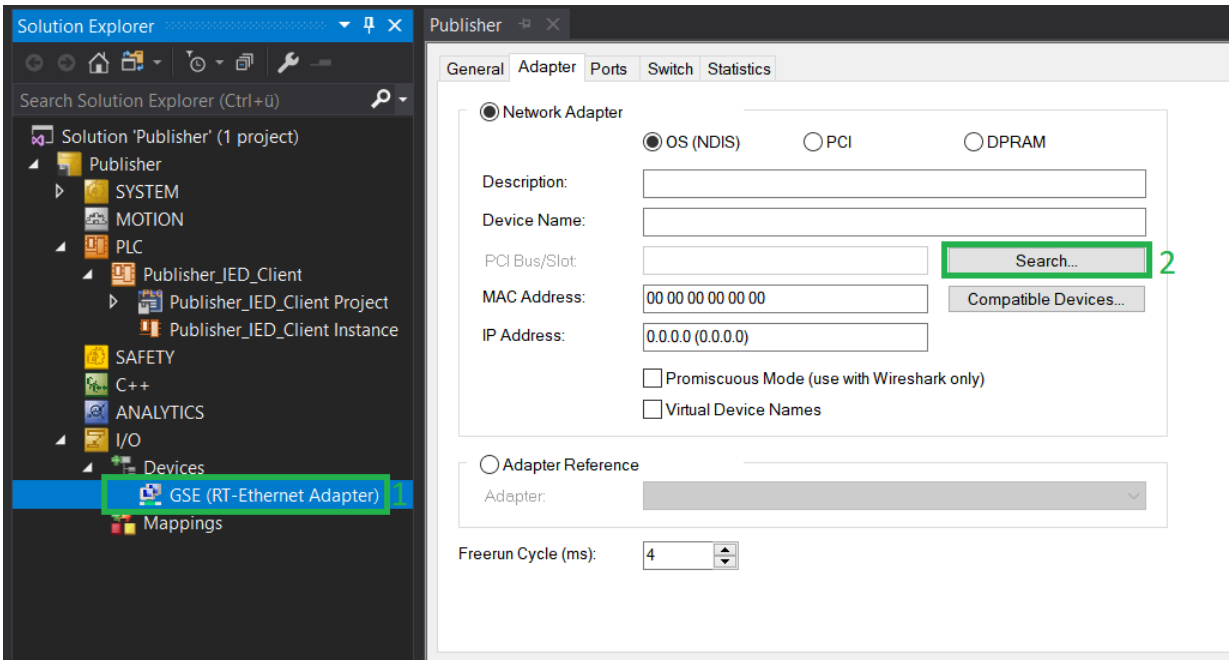


⇨ This adds the adapter.

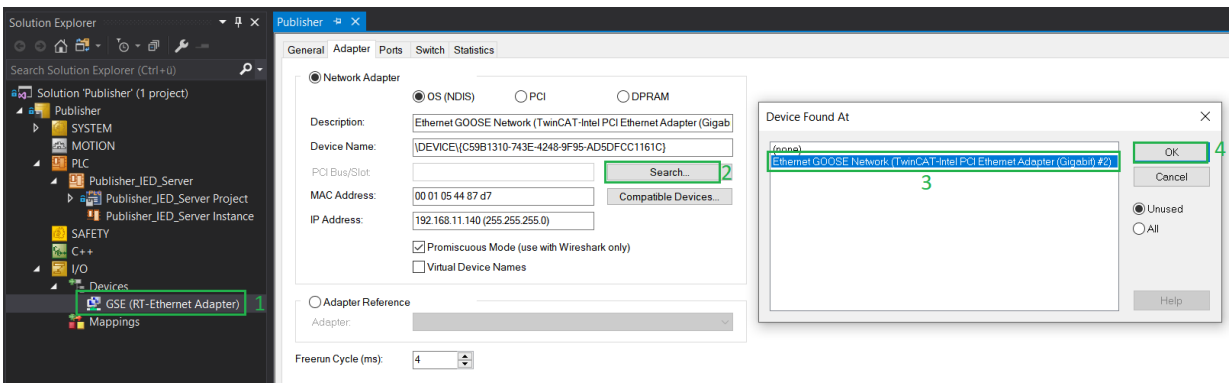
Adapter configuration

✓ Open the adapter properties dialog with a double left mouse click on the adapter.

1. Switch to the **Adapter** tab.
2. There select the command **Search...**.



3. Select an adapter from the list of found adapters and confirm this with **OK**.



⇨ This completes the adapter configuration. In the next step a link must be established between the adapter and the PLC function block instance.

Link adapter with PLC function block instance

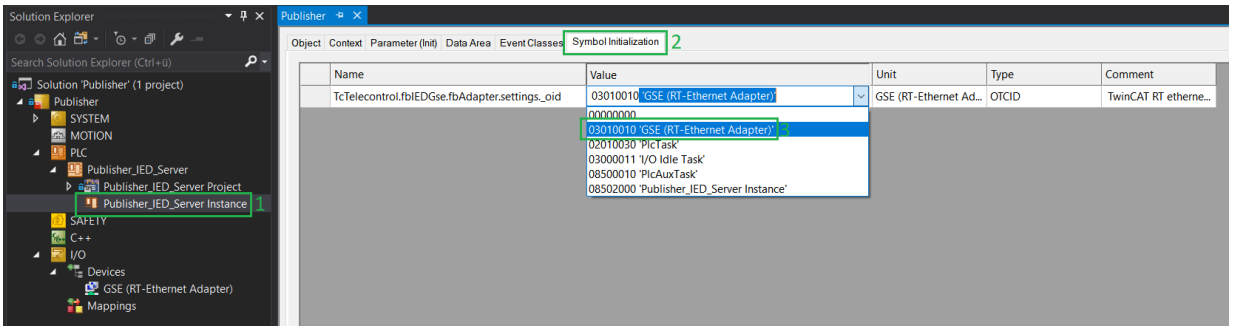
The link between the network adapter and the PLC function block instance is established via a so-called object ID (oid). The object ID's available in the project are displayed on the PLC project instance dialog under the **Symbol Initialization** tab. The project must have been previously compiled at least once for the tab and object IDs to appear. For Goose communication, these are the object IDs of the PLC function block instances of "FB_{IEDName}Gse" and for SV communication (9-2LE), these are the object IDs of the PLC function block instances of: "FB_{IEDName}_{LDName}_MSVCB01SvPublisher", "FB_{IEDName}_{LDName}_MSVCB02SvPublisher", "FB_{IEDName}_{LDName}_MSVCB01SvSubscriber" and "FB_{IEDName}_{LDName}_MSVCB02SvSubscriber" that are linked. The words in curly brackets are placeholders that are replaced with corresponding real values.

NOTICE

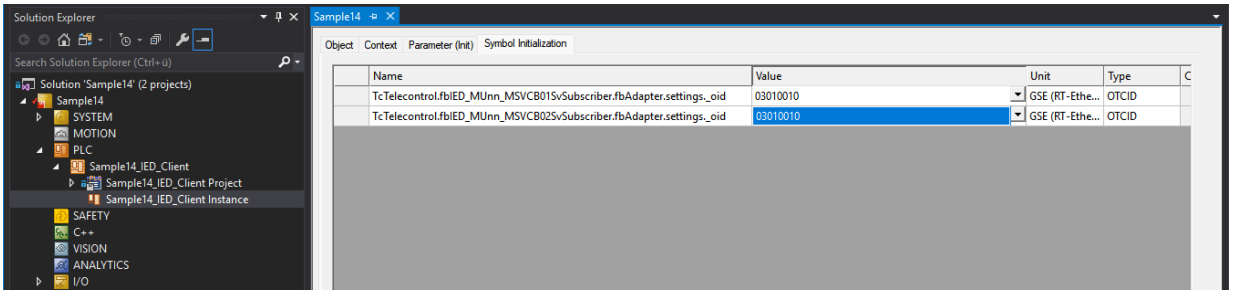
A real-time network adapter can be linked to a maximum of 20 PLC function block instances via the object ID.

✓ Open the adapter properties dialog with a double left mouse click on the adapter.

1. Switch to the **Symbol Initialization** tab.
2. In the column **Value** select your network adapter **GSE (RT Ethernet Adapter)** from the drop-down box.
3. Save all changes in the solution.



The image below shows the SV Subscriber and its Object-IDS's configuration.



⇨ This completes the configuration and linking of the real-time network adapter.

4.7 TCP Keep-Alive Messages

A Keep-Alive telegram is a confirmation message or an acknowledge. This makes it possible to check in the background whether a communication partner who has established a connection is still active and thus still participating in the communication. If a communication partner is no longer active, the communication channel is closed cleanly and regularly to be free for a new participant.

Keep-Alive under Windows

Keep-Alive can be configured under Windows by the keys "KeepAliveTime" and "KeepAliveInterval" in the registry. The default value for the keep alive time is set to two hours, the interval time, i.e. the time until a non-answered alive request is repeated, is set to one second by default. **All values are given in milliseconds**. If the corresponding keys do not already exist in the registry, they must be created as follows.

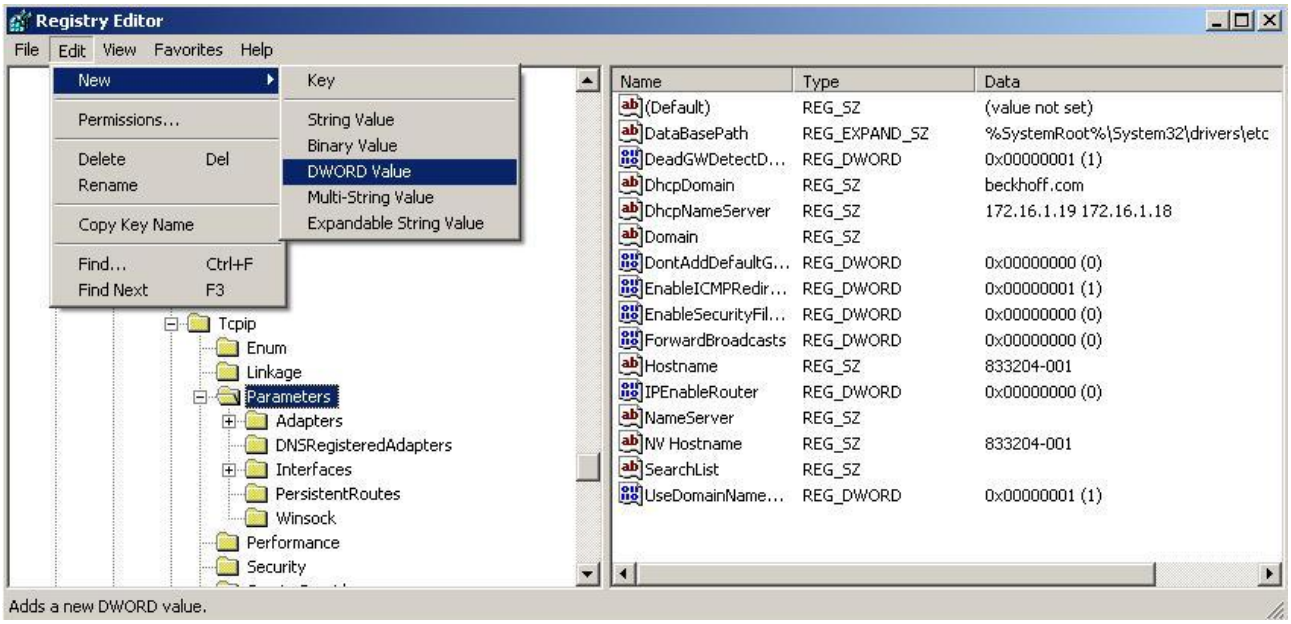
These are not TwinCAT settings, but specific settings of the operating system. Further information can therefore be found in the operating system documentation, for example, on the Microsoft website.

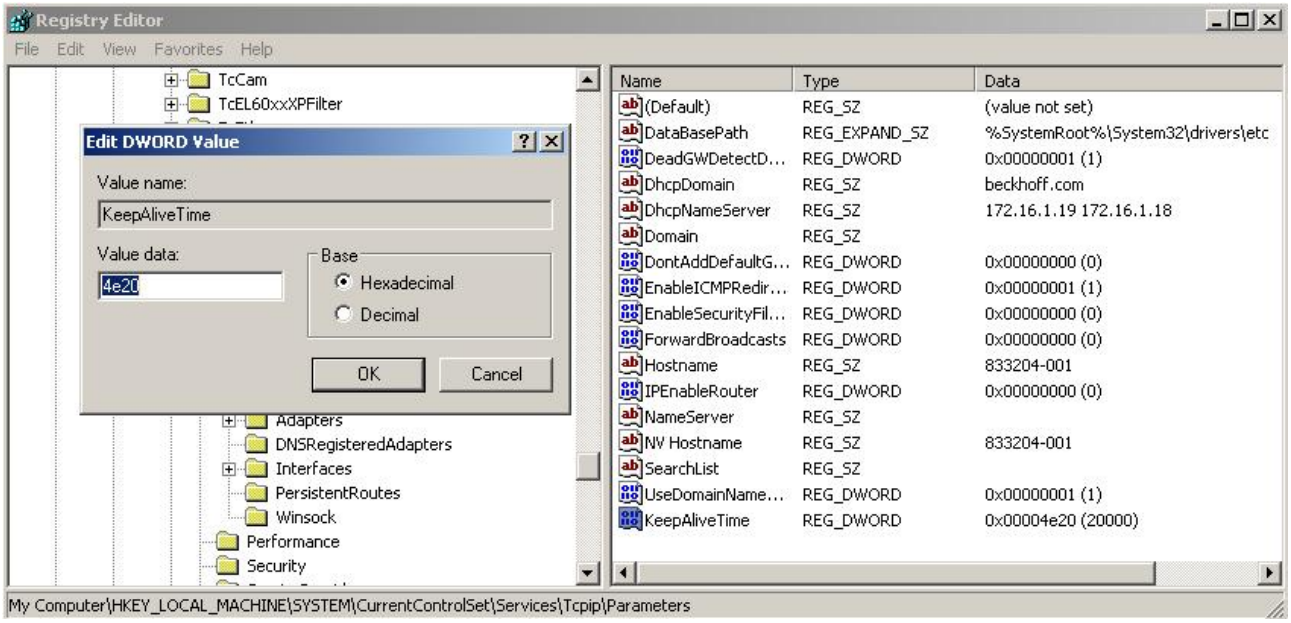
Under Windows W2K, XP, Windows Embedded Standard:

- In the folder *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters* an object named KeepAliveTime must be created as DWORD. To do this, add **Edit > New > DWORD Value**. If you set 0x4E20 as the value here, this corresponds to 20 seconds.
- In the folder *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters* an object named KeepAliveInterval must be created as DWORD. To do this, add **Edit > New > DWORD Value**. If you set 0x1388 as value here, this corresponds to 5 seconds.

Under Windows CE:

- In the folder *HKEY_LOCAL_MACHINE\Comm\TcpIp\Parms* an object with the name KeepAliveTime must be created as DWORD. To do this, add **Edit > New > DWORD Value**. If you set 0x4E20 as the value here, this corresponds to 20 seconds.
- In the folder *HKEY_LOCAL_MACHINE\Comm\TcpIp\Parms* an object named KeepAliveInterval must be created as DWORD. To do this, add **Edit > New > DWORD Value**. If you set 0x1388 as value here, this corresponds to 5 seconds.





Keep-Alive under TC/BSD

The "sysctl" parameters are used by the operating system to configure TCP Keep-Alive messages.

- `keepidle`: Amount of time, in milliseconds, that the connection must be idle before keepalive probes (if enabled) are sent. The default is 7200000 msec (2 hours).
- `keepintvl`: The interval, in milliseconds, between keepalive probes sent to remote machines, when no response is received on a keepidle probe. The default is 75000 msec.

```
> sysctl -d net.inet.tcp | grep -e keep

net.inet.tcp.keepidle: time before keepalive probes begin

net.inet.tcp.keepintvl: time between keepalive probes

net.inet.tcp.keepinit: time to establish connection

net.inet.tcp.keepcnt: Number of keepalive probes to send

net.inet.tcp.always_keepalive: Assume SO_KEEPALIVE on all TCP connections
```

Sample:

keepidle is set to 4000 (ms) and keepintvl to 4000 (ms).

```
sysctl net.inet.tcp.keepidle=4000

sysctl net.inet.tcp.keepintvl=4000
```

Permanent (persistent) configuration of TCP Keep-Alive messages is done in the file */etc/sysctl.conf*.

See also here: https://www.freebsd.org/cgi/man.cgi?query=tcp

4.8 Firewall settings

IEC 61850 uses TCP/IP as the transport protocol (T-profile) during the client/server communication (MMS mapping). Please ensure that the corresponding TCP port is enabled if a firewall is used. The table below lists ports that have to be taken into account when a firewall is used.

Description	Type	Protocol	Port
Connection-Oriented Transport Protocol (COTP, ISO 8073; RFC 1006 unencrypted)	Protocol	TCP	102
Secure ISO TP0 (Secure ISO Transport Protocol Class 0, RFC 1006 over TLS)	Protocol	TCP	3782

Configuration under Windows

The Windows Firewall is configured via the corresponding dialog in the Control Panel. For more information on configuration, refer to the Windows or firewall documentation.

NOTICE

Enable the port

If an embedded controller without monitor connector and USB is used, please ensure that the port for Remote Display (Windows CE) or Remote Desktop (Windows XP / Windows Vista) is enabled in the firewall. Otherwise the computer administration via the network is no longer possible.

Configuration under TC/BSD

✓ Log in to the console.
Default login credentials:
Login: `Administrator`
Password: `1`

✓ Alternatively, for example, via the Git BASH console:
`ssh Administrator@[YourIPAddress]`

1. Next, you need to modify the firewall configuration file to enable the corresponding TCP/IP port. The following command opens the configuration file in the editor:
   `doas ee /etc/pf.conf`
2. The following entry is added to the configuration file (if not already present):
   ```
   #allow traffic to TCL/IP port 102
   Pass in quick proto tcp to port 102 keep state
   ```
3. Escape calls the menu in the editor.
4. Save the changes and exit the editor.

⇨ The configuration is now complete.

Other useful commands:

Reboot device: `doas shutdown -r now`

Disable firewall: `doas pfctl -d`

Enable firewall: `doas pfctl -e`

5 Configuration

5.1 Basics and entry

The TwinCAT Telecontrol Configurator is intended for graphic configuration of the IEC 61850 or IEC 61400-25 data model. ICD files can be read, created and modified. The configuration for TwinCAT control is made available through automatic PLC code generation.

Functions of the main toolbar

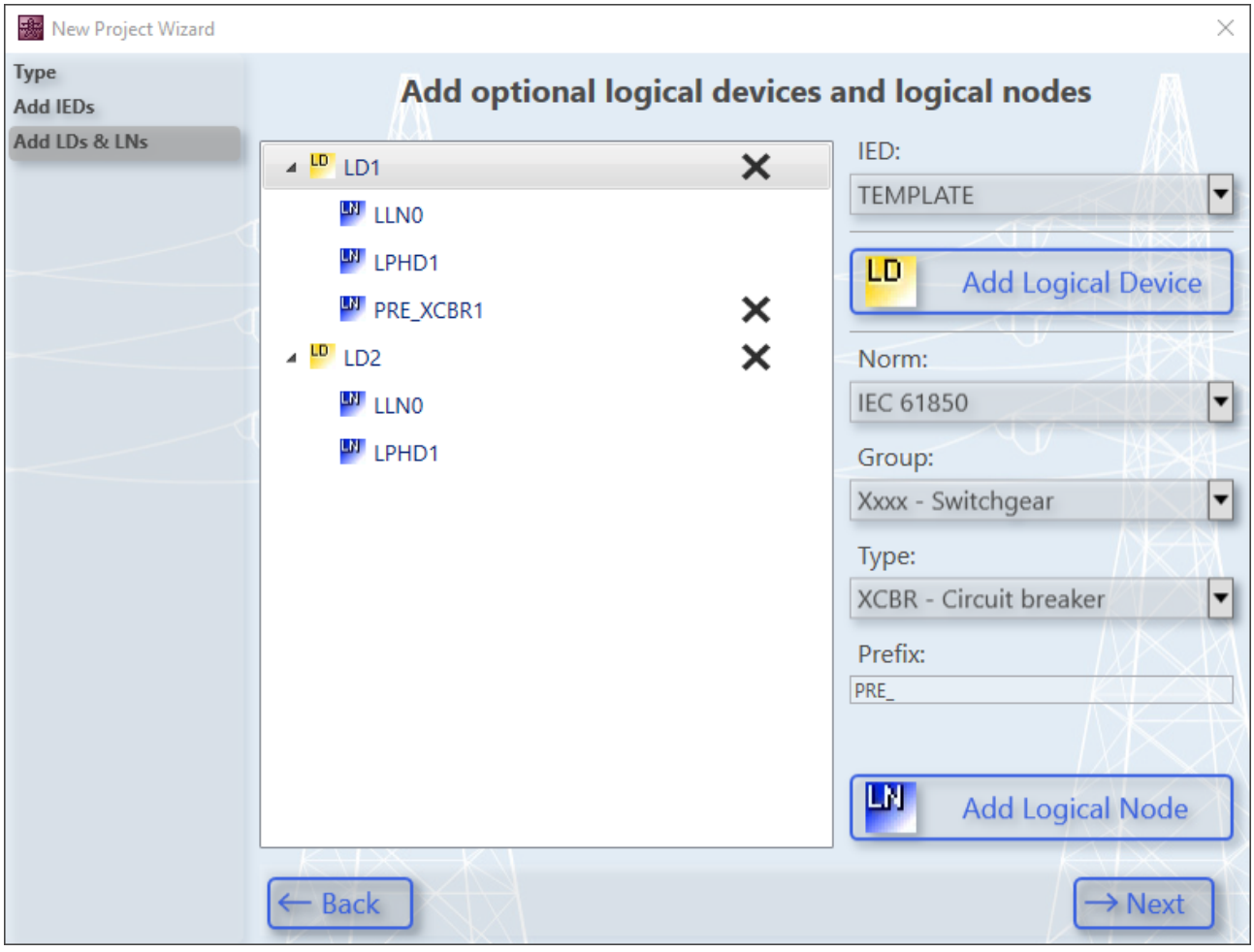


The buttons are described from left to right in the table below.

New Project	Creates a new configuration by initially allowing the selection of a storage location for a new ICD file. The following wizard opens for the preconfiguration of the data model.
Open ICD file	Imports/opens an ICD file
Save ICD-file	Exports/saves a currently opened ICD file
Save ICD-file as...	Exports/saves an ICD file under a new name
Generate PLC code	Generates PLC code for the created configuration

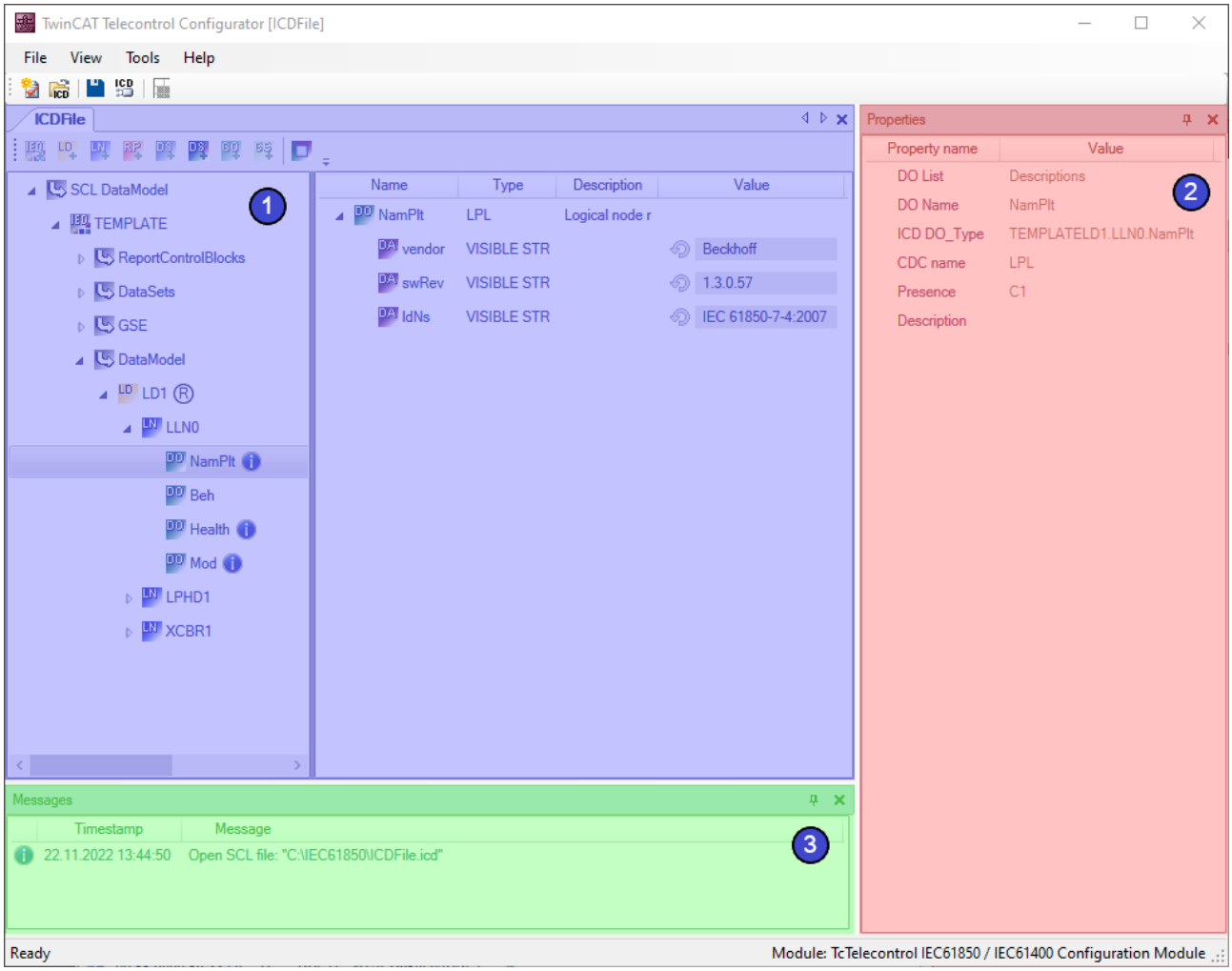
Actuation of the **New Project** button in the toolbar or in the Configurator menu first causes a dialog to open in which a new ICD file can be saved.

In the following, the wizard shown here opens for the preconfiguration of the data model. Here, it is possible to make common settings for the project and to add the supported control block types to the data model.



The **Importing/Opening** and **Exporting/Saving** of ICD files should be controllable intuitively. There is no configurator-specific file format. A further main function is the generation of PLC code. This can be started via the **Generate PLC code** button. The entire data model configuration is thereby converted to structured text, and basic function blocks and calls are generated at the same time.

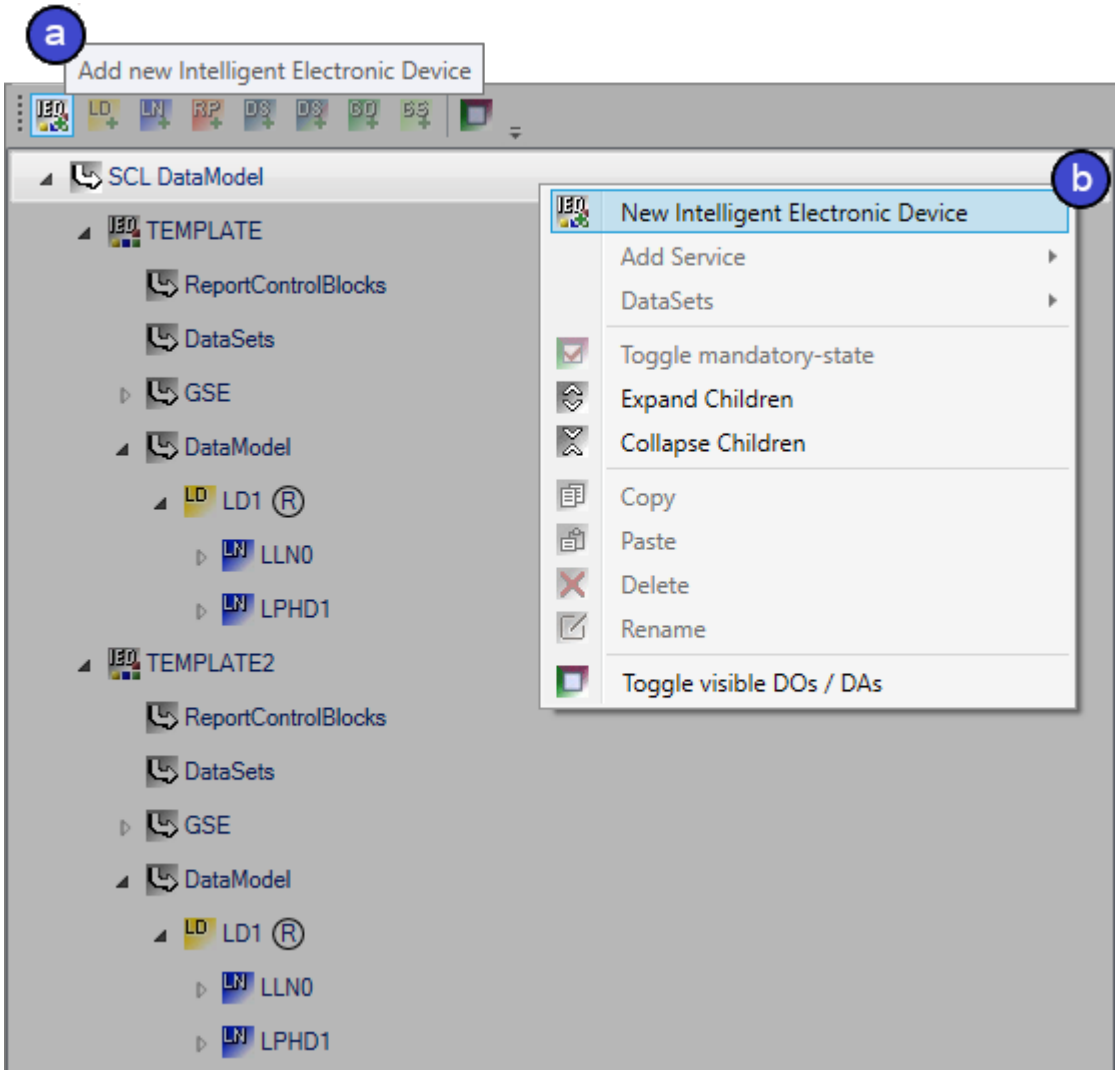
The basic architecture of the Telecontrol Configurator is divided into three main parts:



1. The working part in which the data model is defined. Among other things, optional objects can be added or deselected, data sets added, reports and GOOSE control blocks configured, and default values set.
2. Further options and information can be found in the Properties.
3. The Message window displays information and error messages.

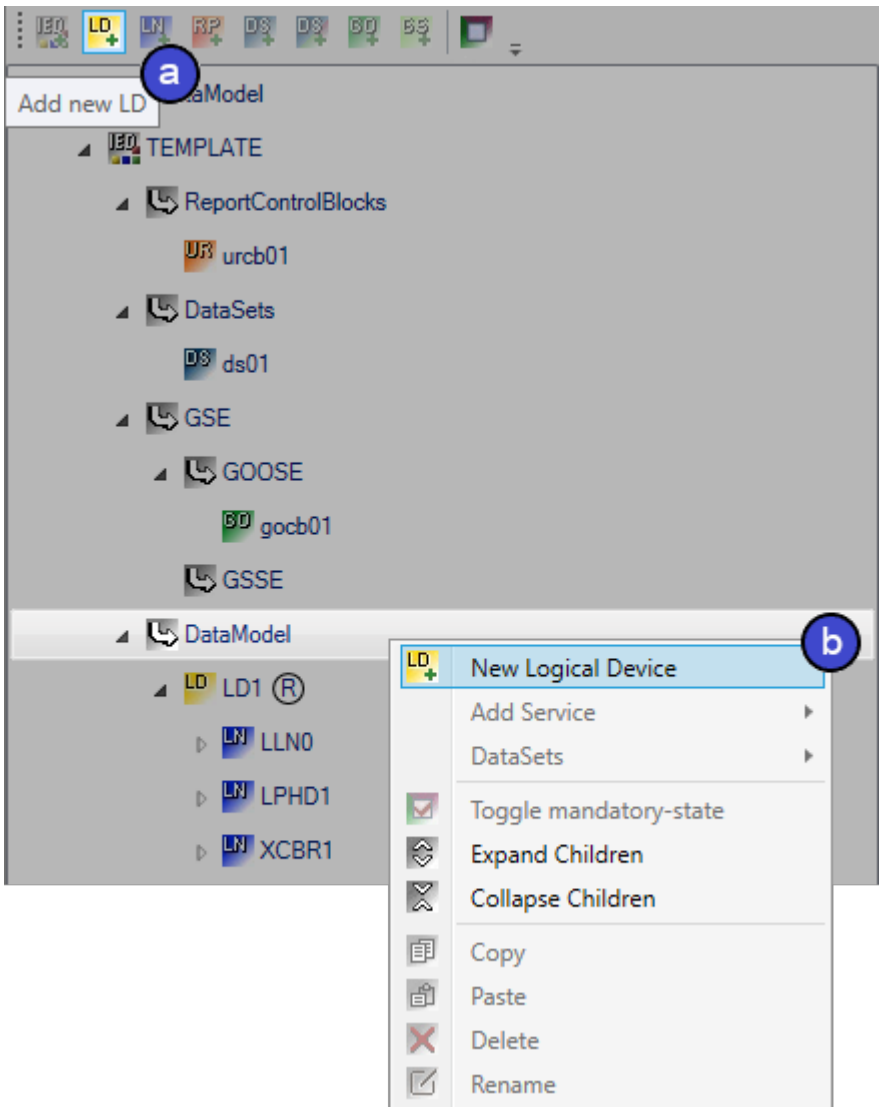
Add Intelligent Electronic Devices

To add more Intelligent Electric Devices, you can click the **SCL DataModel** node in the data model to execute a **New Intelligent Electronic Device** via the toolbar in the project window (**a**) or via the context menu for the node (**b**).



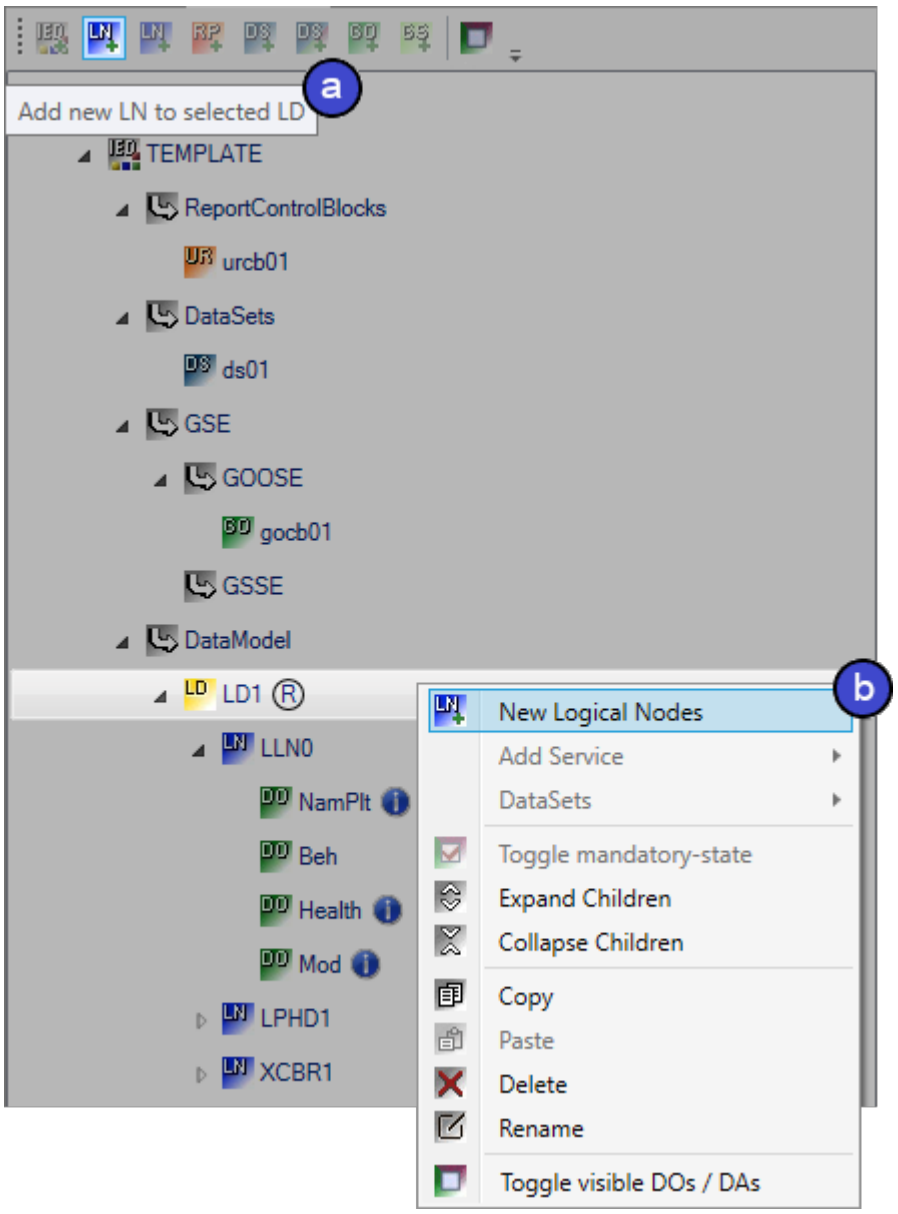
Add Logical Devices

To add more Logical Devices, you can click the **DataModel** node of the corresponding Intelligent Electronic Device to execute a **New Logical Device** via the toolbar in the project window (**a**) or via the context menu for the node (**b**).

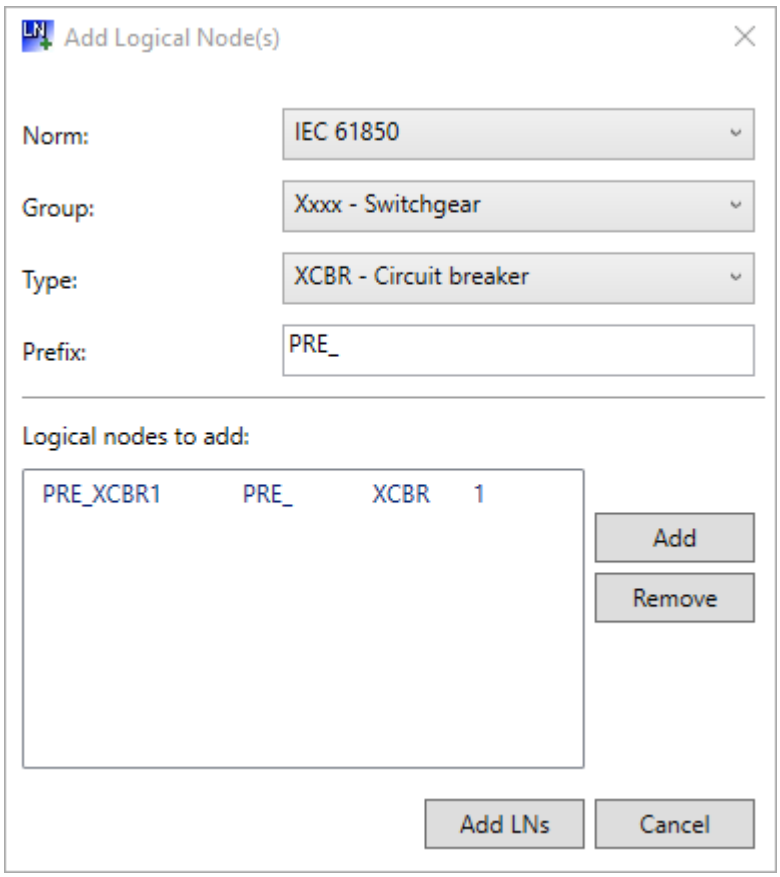


Add Logical Nodes

To configure additional Logical Nodes, you can click the device in order to execute **New Logical Nodes** via the toolbar in the project window (**a**) or via the context menu of the node (**b**).

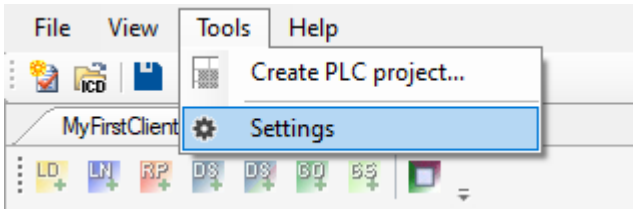


In this case the following wizard starts, where you can select the standard from which you wish to serve yourself. The group of Logical Nodes and the type of node itself in the following. A prefix can optionally be issued.



Default settings

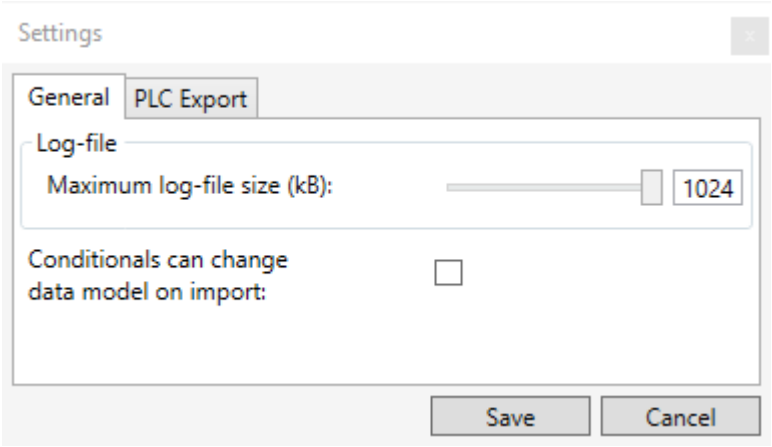
The Telecontrol Configurator also offers the possibility of saving some default settings across different projects. Open the settings via the **Tools** menu in the main toolbar and select **Settings**.



In the following, you will see a new window that is divided into 2 parts. On the one hand, **general configurator settings**; and on the other hand, default settings for **PLC code generation**. These default settings serve as initial values and can be adapted before PLC code generation.

You have the possibility of defining the maximum size of the log file in the **General configurator settings**. Here, you can select any amount between 1 kB and 1024 kB. If you set a value that is smaller than the current size of the log file, then the oldest log entries are removed until the file size corresponds to the new value.

In addition, there is an option here to handle conditional Data Objects during ICD file import. This option is used to import the data model in compliance with the standard. This setting is optional because this may change your existing data model. If this setting is enabled, you will be informed about changes made automatically to the data model in the message window during ICD import.

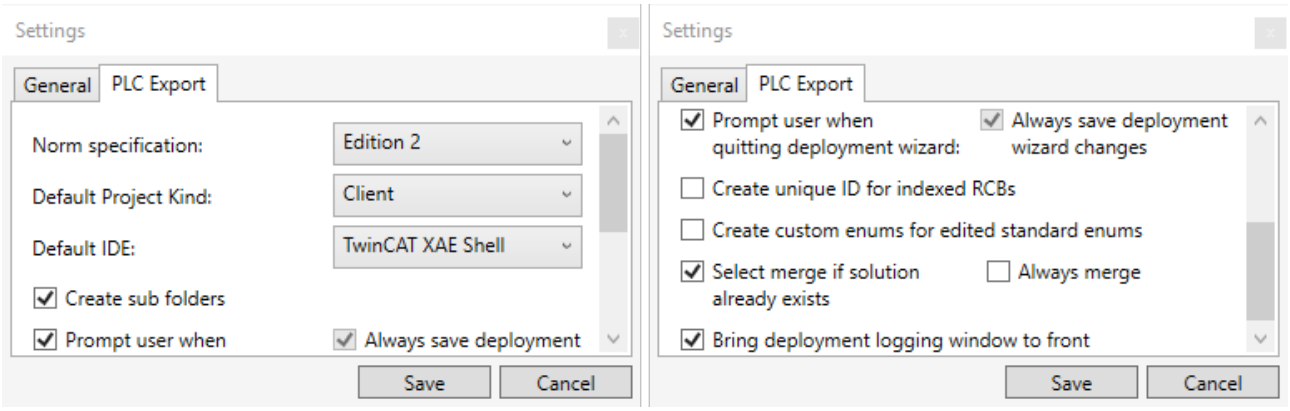


You can set default values in the **Settings for PLC code generation**. This allows you to specify whether you want to generate a client or a server by default, or which development environment you want to use for generation. Here you can also set whether to generate subfolders by default to organize the data model.

In addition, you can set whether a query is displayed when the Deploy TwinCAT Telecontrol dialog is closed prematurely, asking whether the settings made should nevertheless be saved until the next time the dialog is opened. Here you can also select that the settings made should never be saved.

The following options are for the PLC code to be generated. "Create unique ID for indexed RCBs" is used to generate unique report IDs for indexed report control blocks. If a standard-specific enum no longer matches the default assigned values due to editing, the "Create custom enums for edited standard enums" option can be used to specify that a new user-defined enum is created for this purpose instead of using the existing enum from the PLC library.

It is also possible to automatically select the "Merge to existing Project" option when starting the Deploy TwinCAT Telecontrol dialog, if a PLC project already exists at the storage location of the respective ICD file. With the "Always merge" option you can determine here whether this should be selected automatically at each start of the dialog or only at the first dialog start of the current ICD project. The last option allows you to set whether the log window should remain in the foreground by default during code generation.



Log file

Critical errors in the Telecontrol Configurator are stored in a log file. These can be found in the installation directory of the configurator. You can define the file size for the log file in the general configurator settings. By default it is set to 1024 kB. If the file size exceeds the set maximum value, then the oldest log entries are deleted from the file in order to comply with the maximum value.

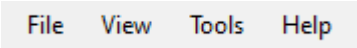
The log file is in XML format. A log entry always contains information about the severity of the message and when the message occurred. If the error occurs during PLC code generation, the PLC Export category is also saved.

Each log entry also contains a readable message entry for independent troubleshooting. This is provided for cases where an error occurs due to misapplication, for example when an attempt is made to import an invalid ICD file. The encrypted stack entry helps to localize the error, if it is a Telecontrol Configurator error that cannot be solved independently and you contact our support about it.

```
<Log Severity="Error" Timestamp="2021-07-14T15:38:56.1067575+02:00">
    <Message><![CDATA[Object reference not set to an instance of an object.]]></Message>
    <Stack><![CDATA[CD-58-70-38-BD-EF-BC-3B-03-06-B4-05-D1-B3-BA-97-14-BE-2B-D2-5A-BA-50-C8
</Log>
<Log Category="PLC Export" Severity="Error" Timestamp="2021-07-15T11:45:08.9454274+02:00">
    <Message><![CDATA[Default Library 'Tc3_Gse, * (Beckhoff Automation GmbH)' not found! (I
    <Stack><![CDATA[CD-58-70-38-BD-EF-BC-3B-03-06-B4-05-D1-B3-AD-9B-24-BE-00-D2-57-B0-4C-DI
</Log>
```

5.2 Menu bar

Navigation



File:

Basic functions such as creating/opening/saving/closing ICD files, exiting the Telecontrol Configurator.

View:

Functions for the Telecontrol Configurator interface.

Tools:

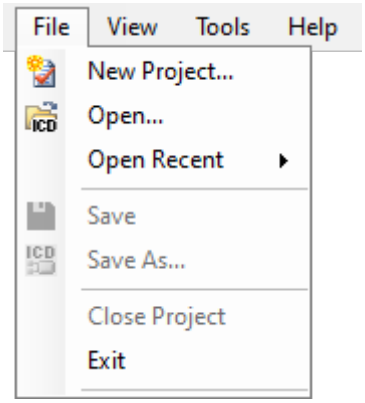
Function for PLC code generation and settings.

Help:

Information about the Telecontrol Configurator.

5.2.1 File

Basic functions such as creating/opening/saving/closing ICD files, exiting the Telecontrol Configurator.



New Project…

Creates a new configuration by initially allowing the selection of a storage location for a new ICD file. A wizard opens afterwards for the preconfiguration of the data model.

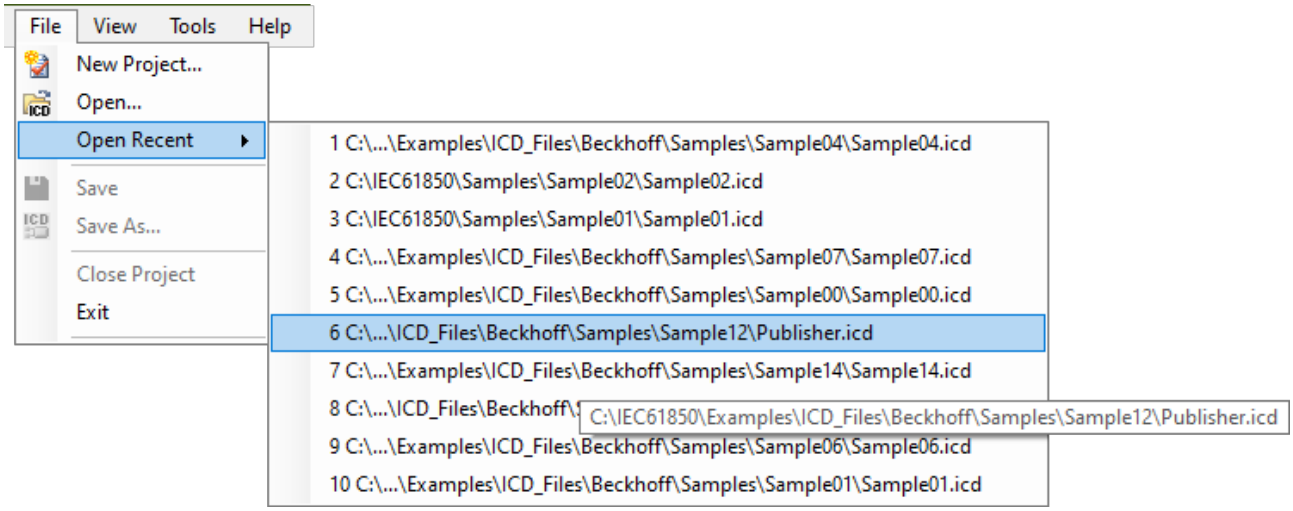
Full description [▶ 42]

Open…

Imports/opens an ICD file.

Open Recent ▶

Imports/opens an ICD file from a list of the last 10 projects opened with the Telecontrol Configurator.



Long paths are shortened, you can see the full path in the respective tooltip.

Save

Exports/saves a currently opened ICD file.

Save As…

Exports/saves an ICD file under a new name.

Close Project

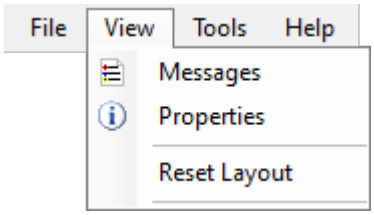
Closes the currently opened ICD file.

Exit

Exits the Telecontrol Configurator.

5.2.2 View

Functions for the Telecontrol Configurator interface.



Messages

Displays the Messages window.

Properties

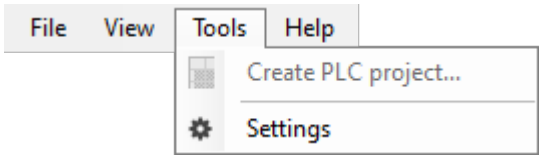
Displays the Properties window.

Reset Layout

Resets the layout of the Telecontrol Configurator.

5.2.3 Tools

Function for PLC code generation and settings.



Create PLC project…

Opens the dialog for automatic code generation.

Full description [▶ 80]

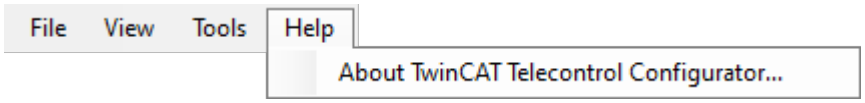
Settings

Opens the general default settings of the Telecontrol Configurator.

Full description [▶ 42]

5.2.4 Help

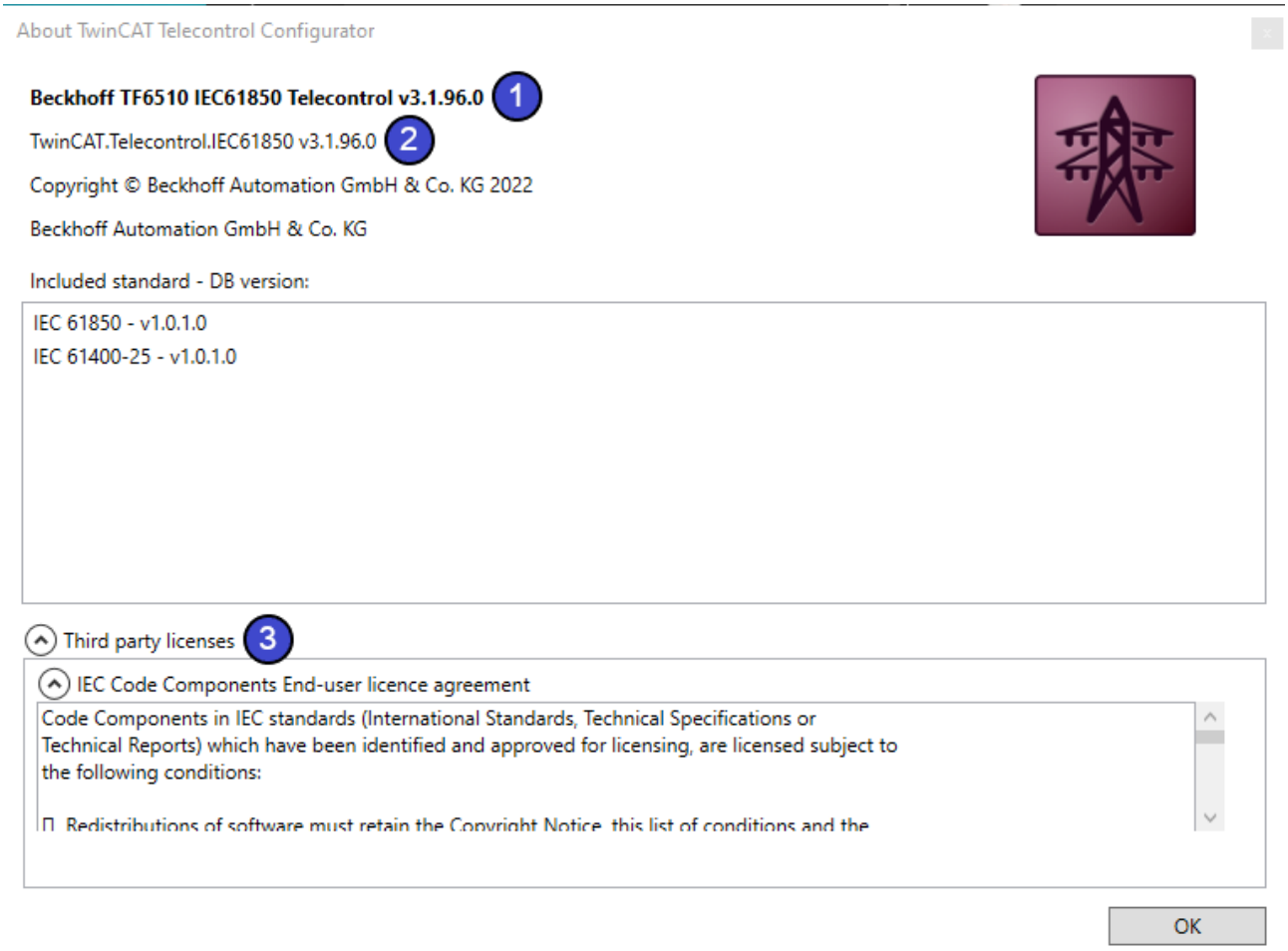
Information about the Telecontrol Configurator.



About TwinCAT Telecontrol Configurator...

Opens a dialog with current information about the Configurator. Here you can see, among other things, the currently installed version number and the licenses of the third-party providers used (**3**).

The version numbers stand for the product version (**1**) and Configurator version (**2**).

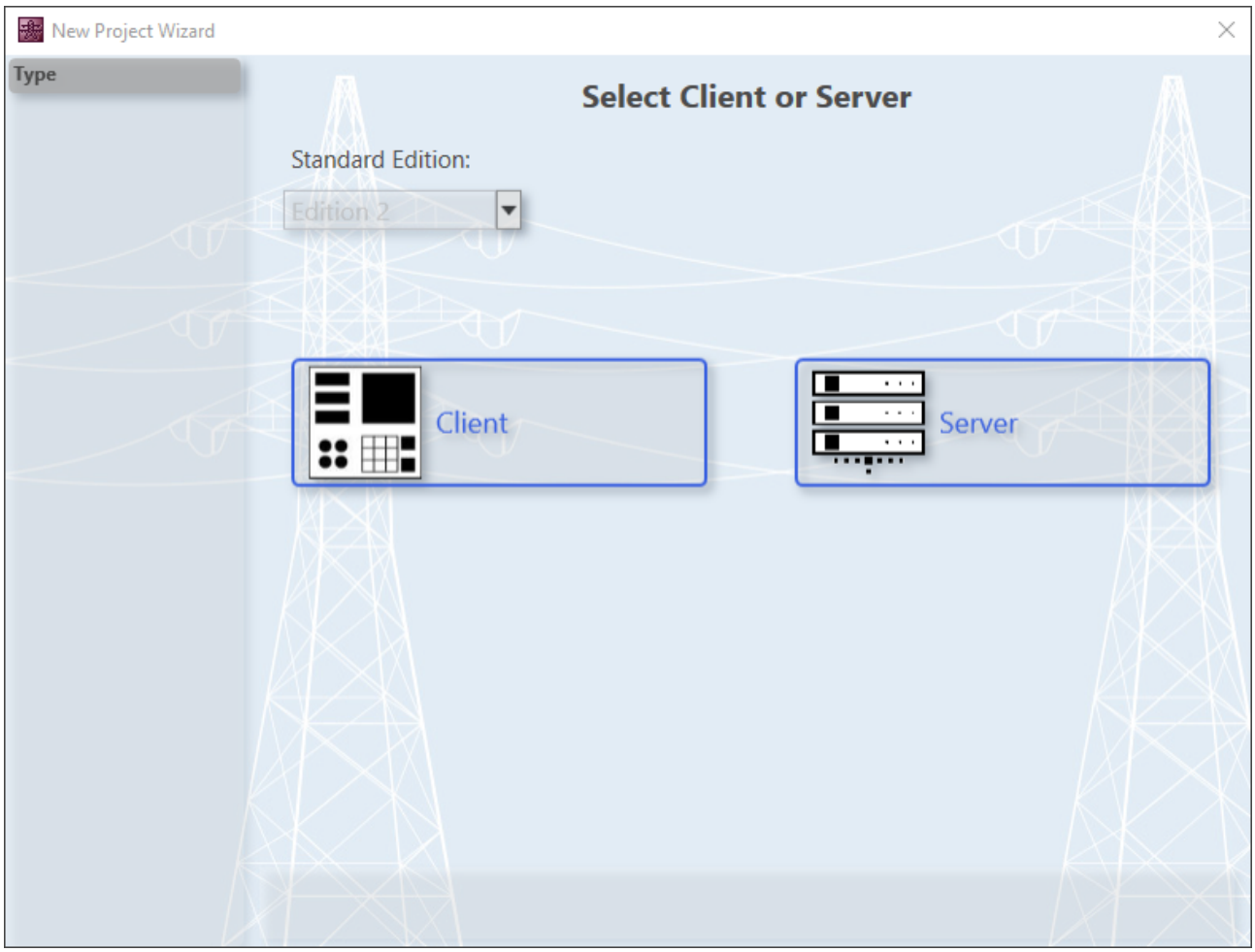


5.3 New Project Wizard

You can use the New Project wizard to preconfigure a new data model by following the steps. It is possible to make common settings for the project and to add the supported control block types to the data model.

Select project type

At the beginning of the wizard, you can decide whether you want to configure a Client or Server. This has no effect on the data model. This default setting can be adapted at any time in the PLC code generation.

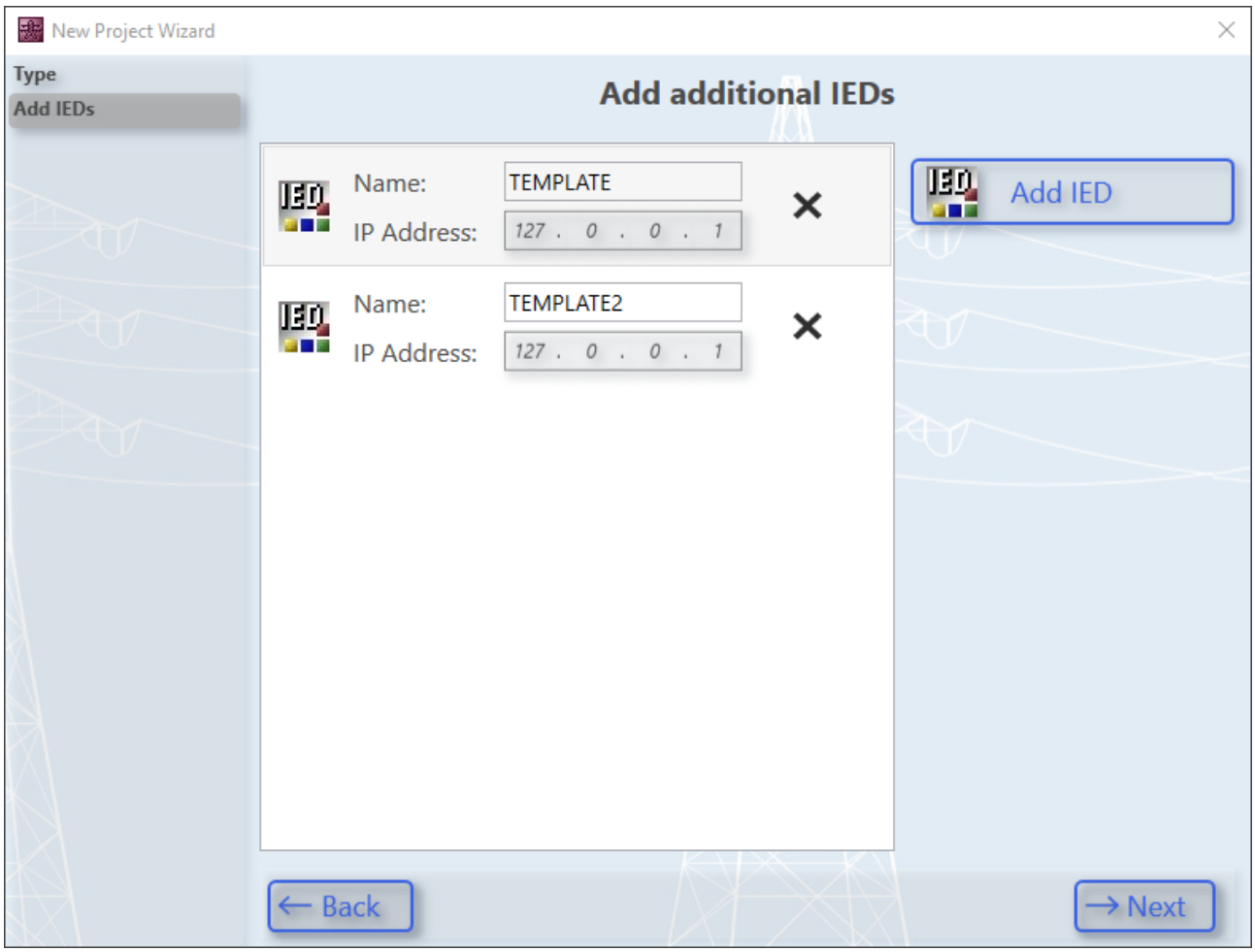


Add Intelligent Electronic Devices

On the following page you can add additional Intelligent Electronic Devices, short IEDs to the new project. At least one IED is required for configuration.

You can assign a name and an IP address to each Intelligent Electronic Device. In the case of a server configuration, this is the IP address at which the server can be reached. For client configuration, the IP address of the server to be connected.

The data model is preconfigured with an Intelligent Electronic Device by default.

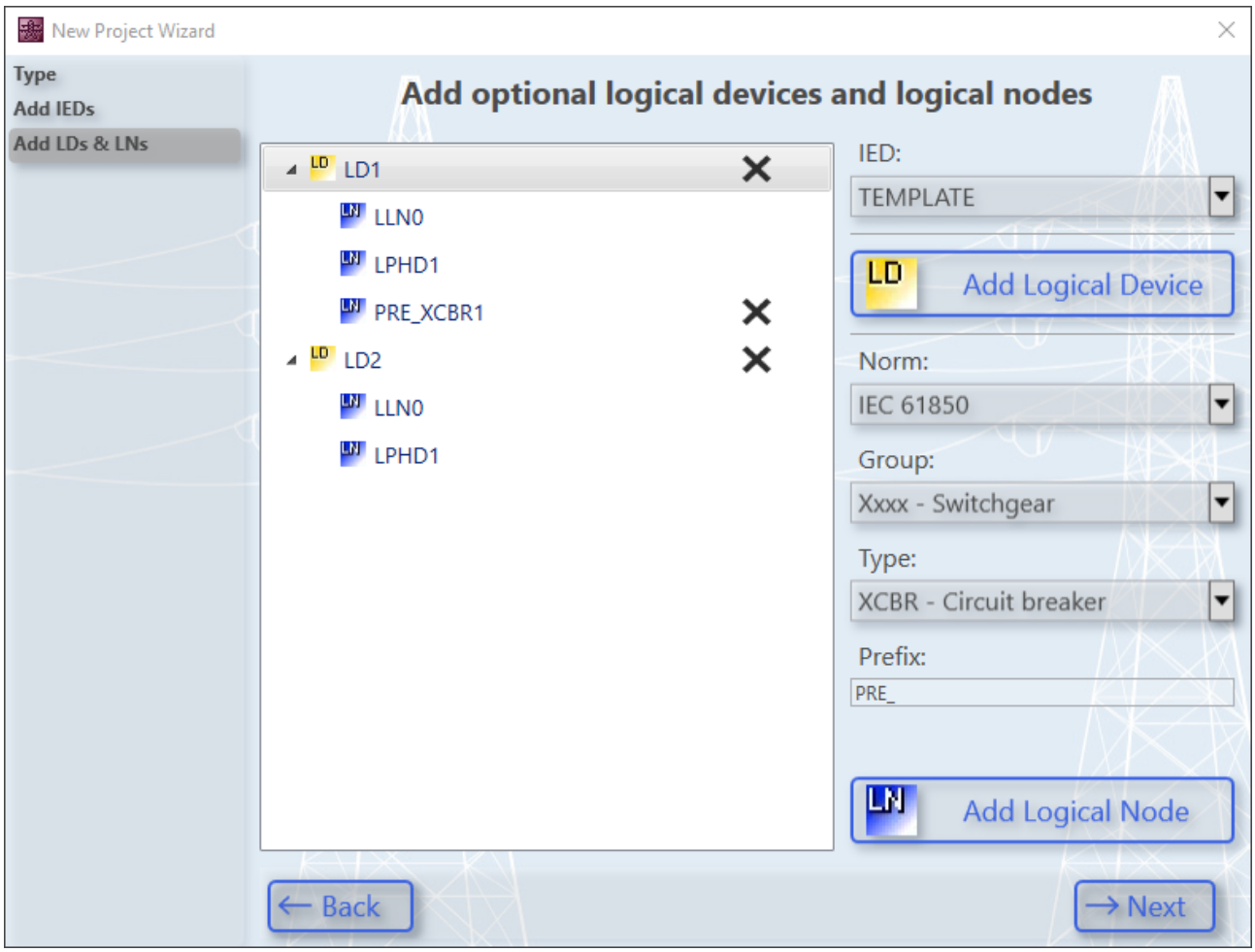


Add Logical Nodes

On the following page you can add Logical Devices to the selected Intelligent Electronic Device. At least one Logical Device per Intelligent Electronic Device is required for configuration.

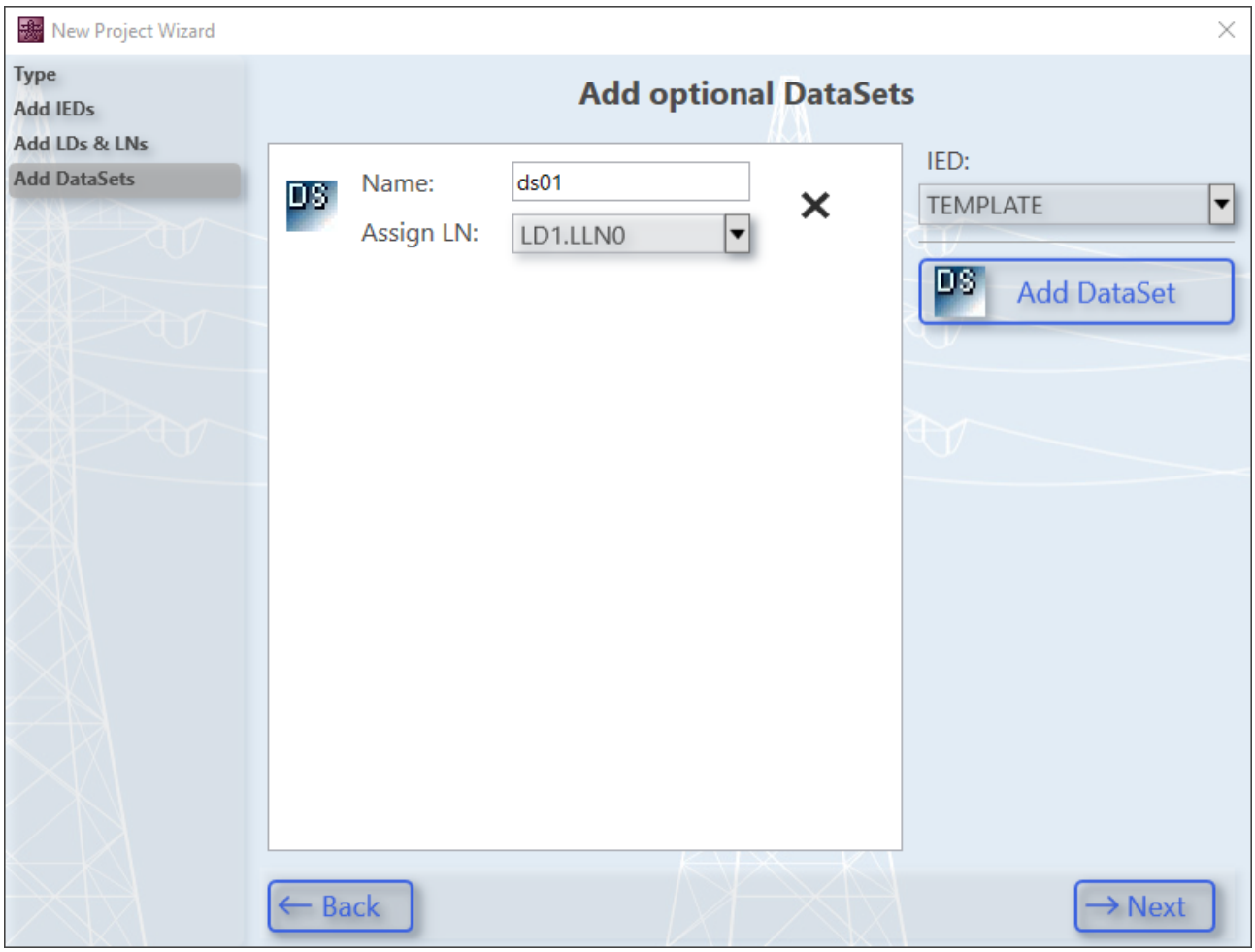
In addition, you can add any Logical Nodes to the selected Logical Device. When selecting Logical Nodes, you can select the standard. The group of Logical Nodes and the type of node in the following. A prefix can optionally be issued.

Each Intelligent Electronic Device is preconfigured by default with a Logical Device and the two mandatory Logical Nodes LLN0 and LPHD.



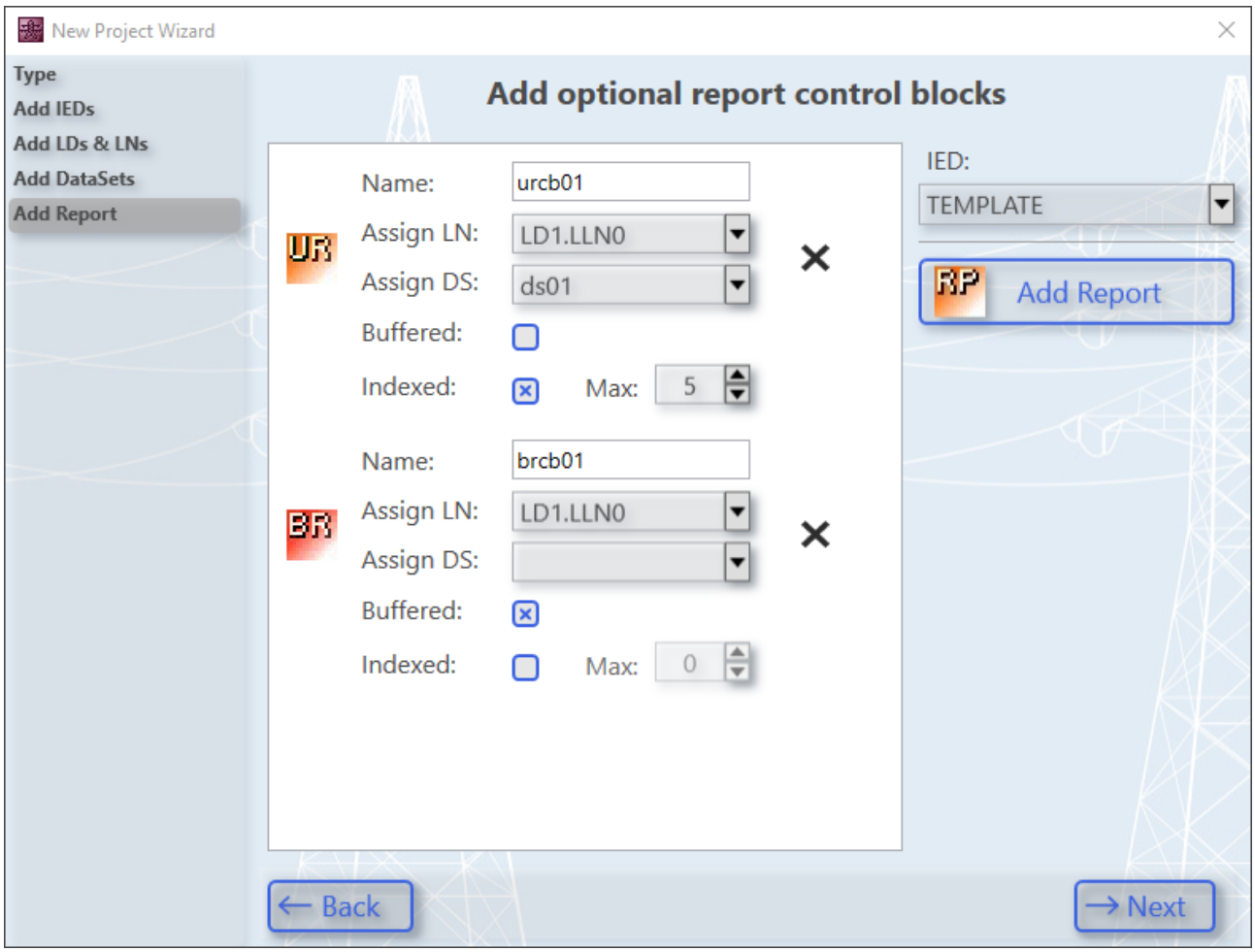
Add DataSets

On the following page you can add any DataSets to the selected Intelligent Electronic Device. You can assign a name and assign the DataSet to a Logical Node. In the New Project Wizard, it is not possible to add Functional Constraints and DataAttributes to the DataSets. You can do this after completing the wizard in the working area of the configurator.



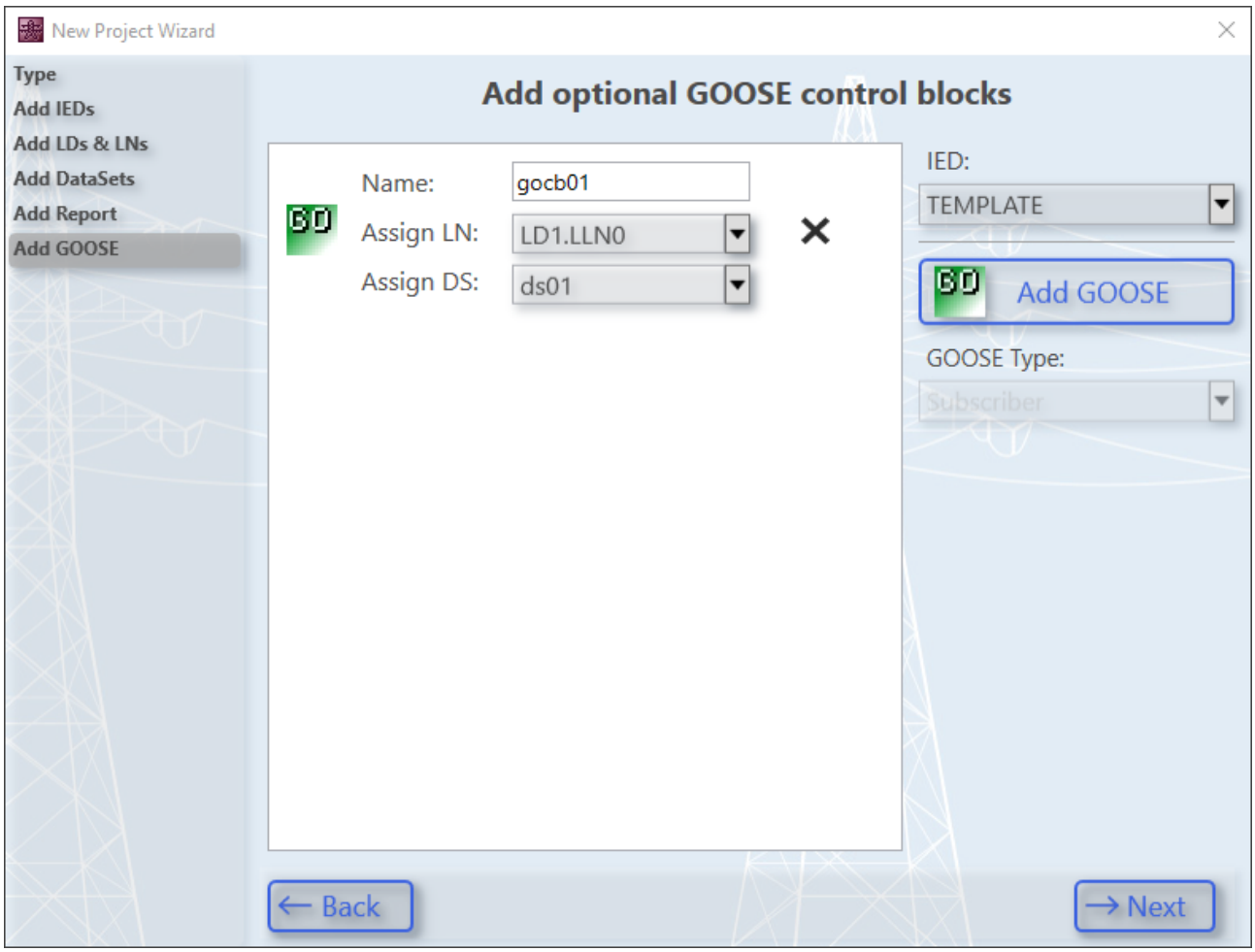
Add Report Control Blocks

On the following page you can add any Report Control Blocks to the selected Intelligent Electronic Device. You can select the name and the assigned Logical Node. Additionally you can assign a DataSet, select between **Unbuffered** and **Buffered** Report and index the selected Report Control Block with the selected maximum value.



Add GOOSE Control Blocks

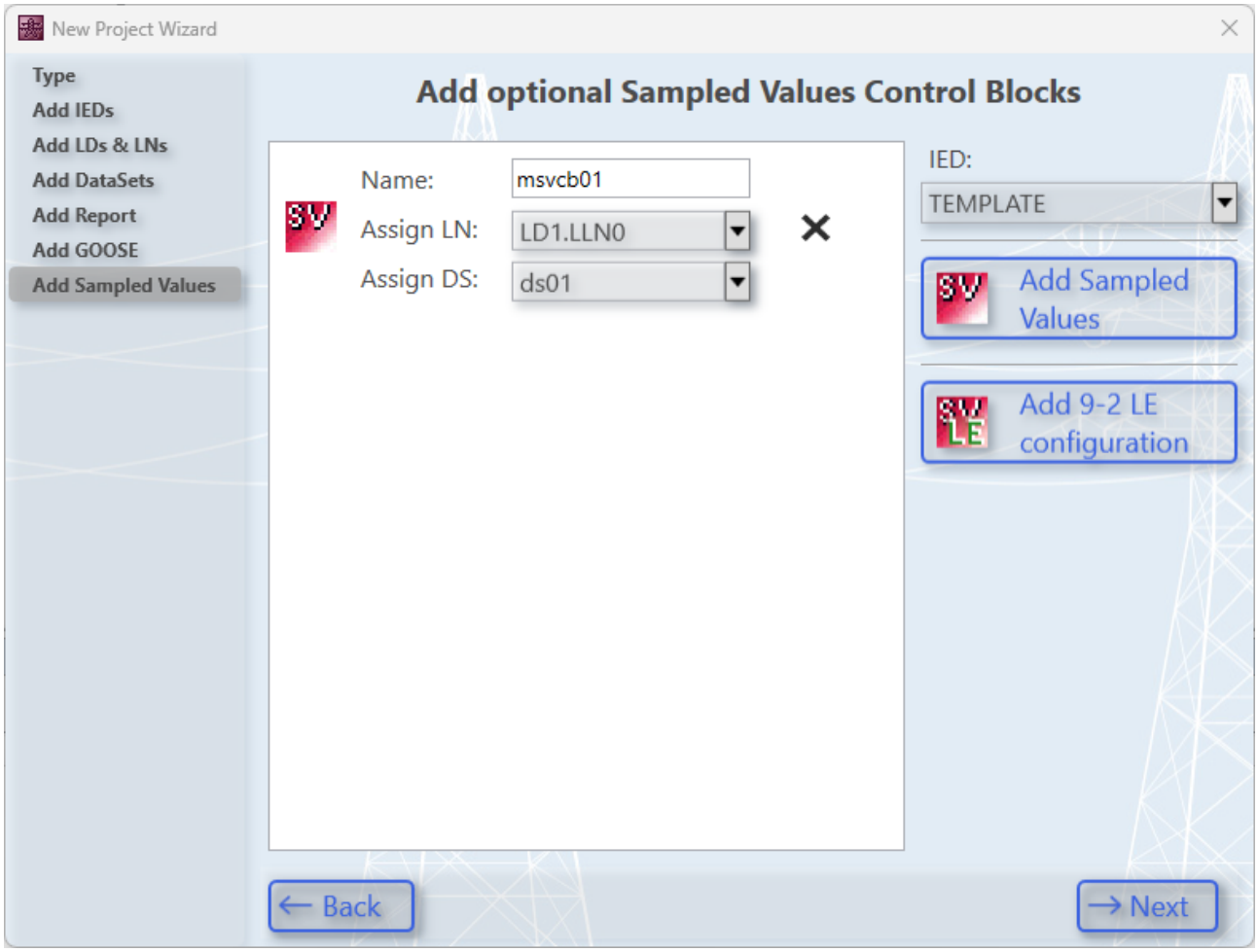
On the following page you can add any GOOSE Control Blocks to the selected Intelligent Electronic Device. In addition to the options to select the name and the assigned Logical Node, you can still also assign a DataSet here. As soon as a GOOSE Control Block has been added, the GOOSE type of the project is displayed on the right side. In the event of a Client configuration, it is a GOOSE Subscriber; in the event of a Server configuration, it is a GOOSE Publisher.



Add Sampled Values Control Blocks

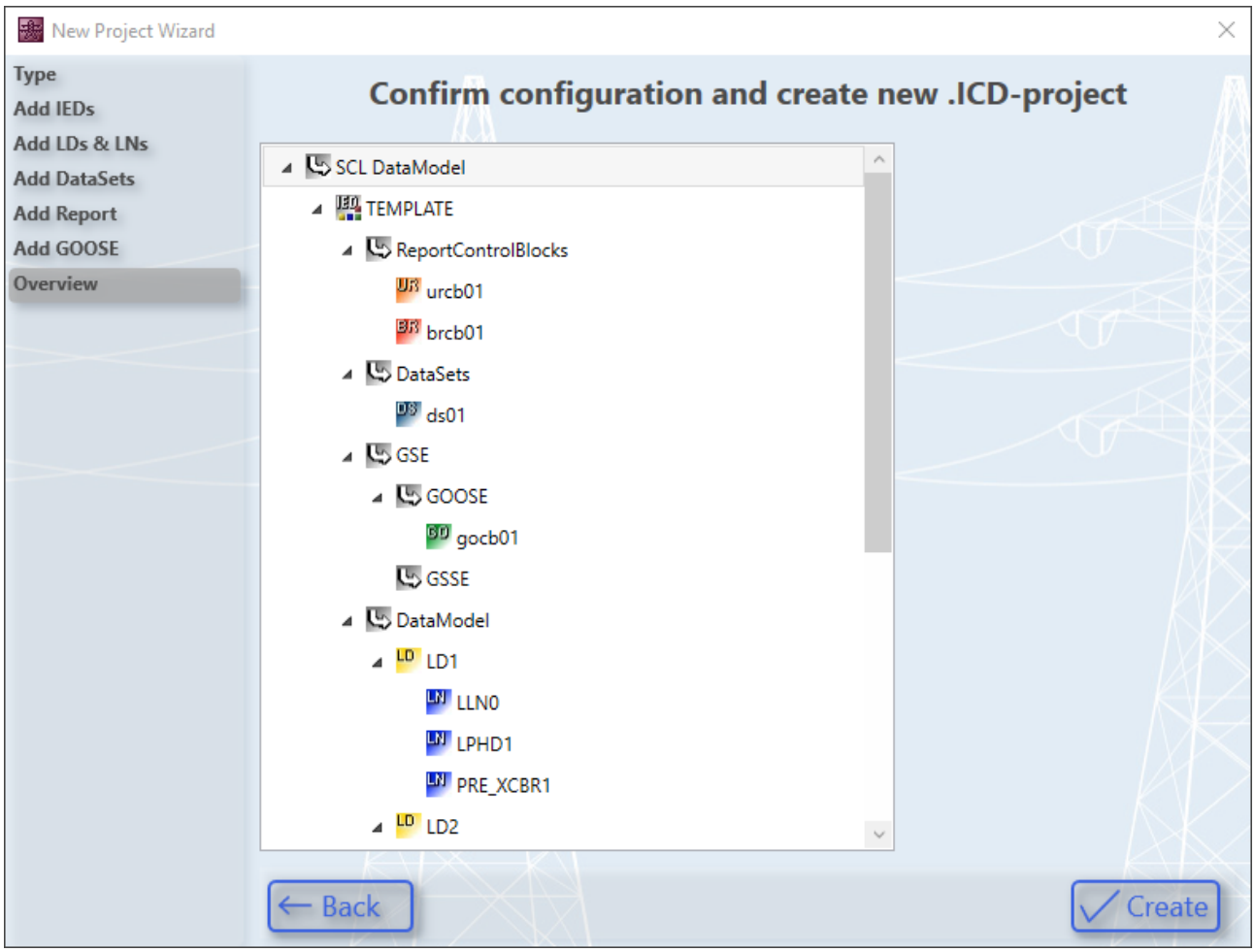
On the following page you can add any Sampled Value Control Blocks to the selected Intelligent Electronic Device. You can specify the name, select the assigned Logical Node and assign a DataSet in each case.

You can also add a configuration in accordance with IEC 61850-9-2-LE (Light Edition) using the separate button. Further information on configuring the Light Edition can be found here: Sampled Values [▶ 75]



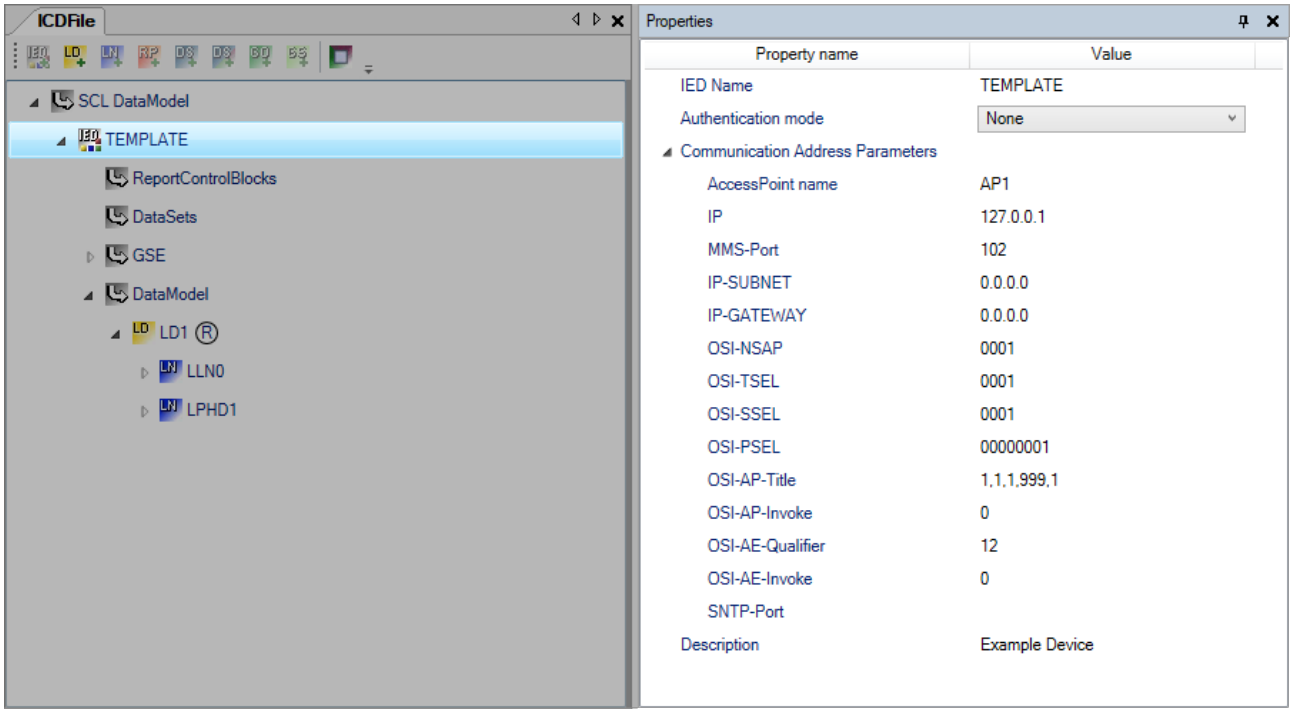
Overview

At the end of the wizard, you will once again see the previously configured data model. Once you have completed the preconfiguration, you can press the Create button to save the preconfiguration under the previously selected file name. Now you can make further settings or generate PLC code in the configurator. Alternatively, you can close the wizard at any time by clicking the Close button. If this is the case, the preconfiguration is **not** saved.



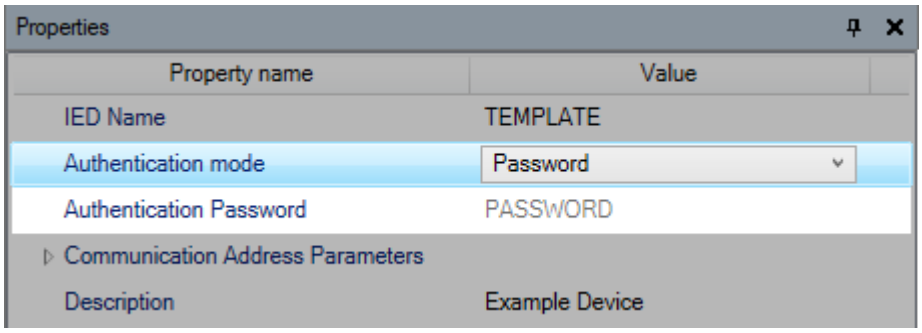
5.4 Device configuration

You can configure various device settings in properties.



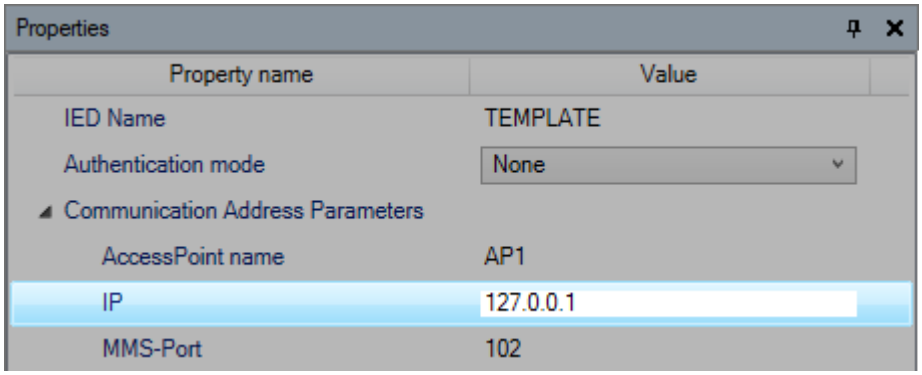
Set authentication options

In the configurator, you can choose between two authentication options: "none" and "password". If you select "password", another entry appears with a default password that is generated during automatic code generation. The password cannot be set in the configurator, as IEC 61850 does not provide for the password to be saved in the ICD file. Displaying the default password only serves to improve traceability in the automatically generated code.



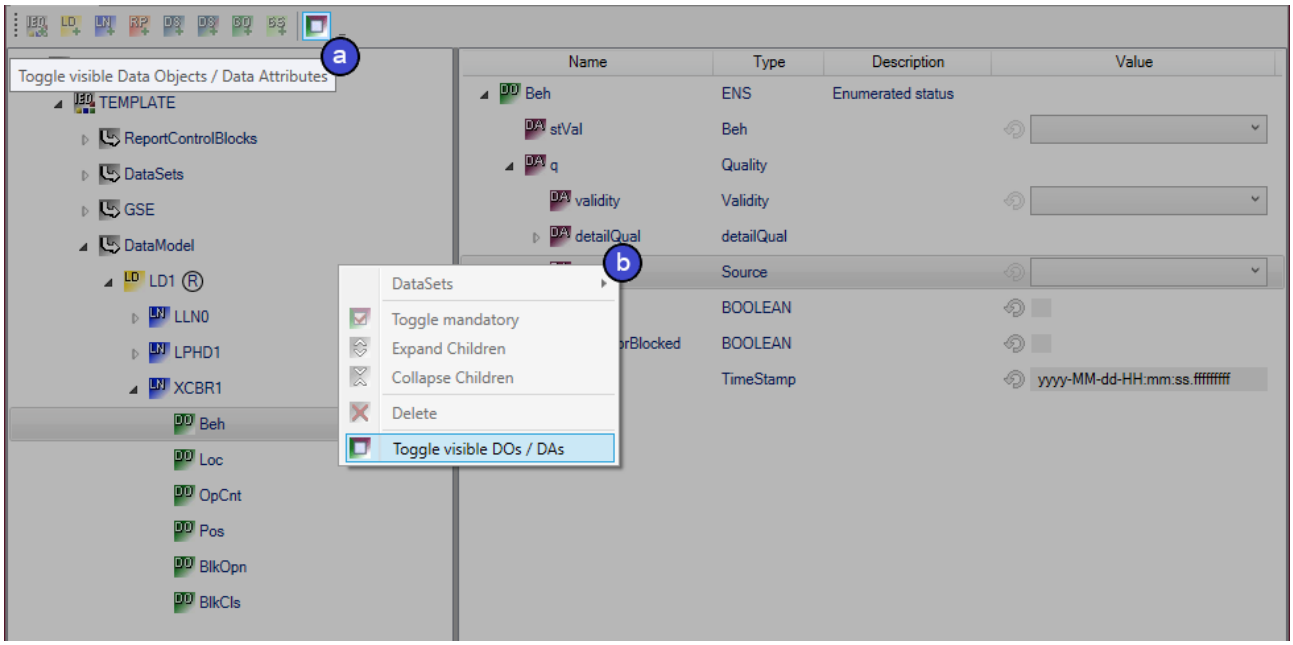
Set communication parameters

You can set the IP address of the device in the communication parameters.

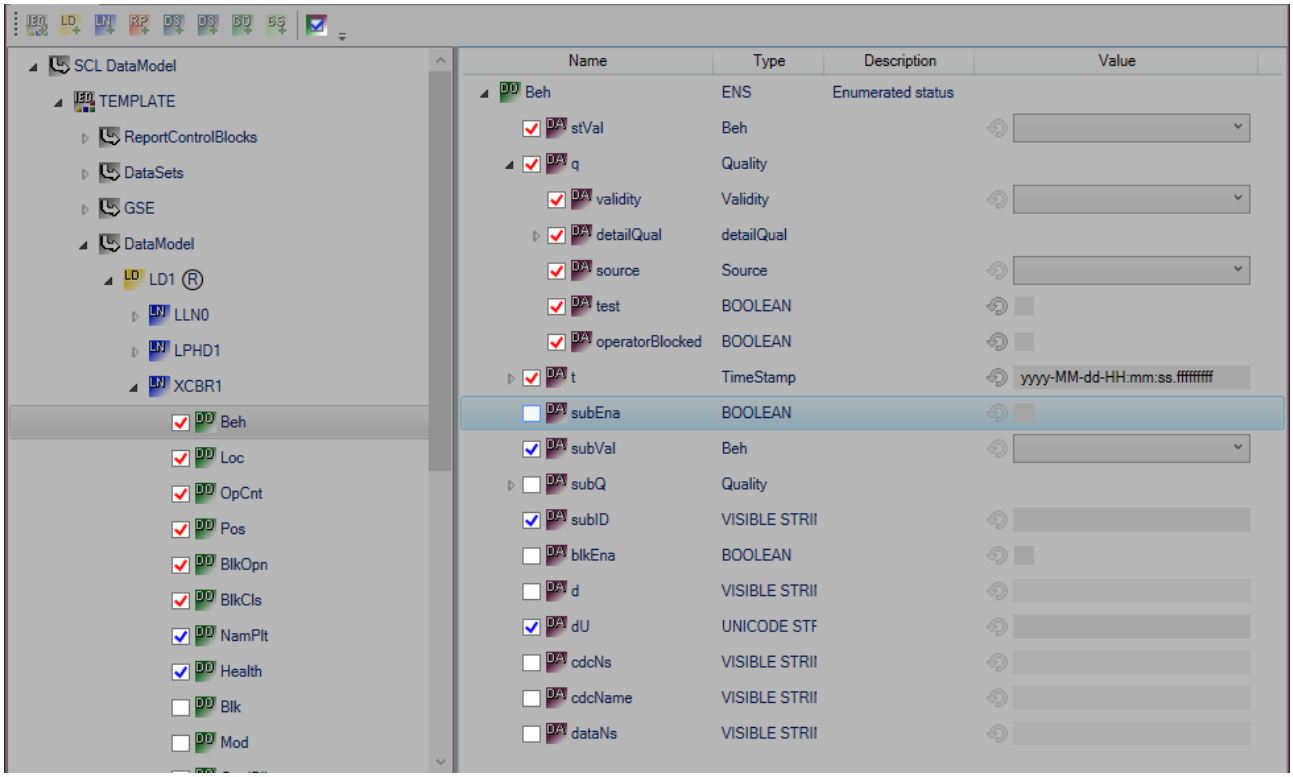


5.5 Configuration of the data objects

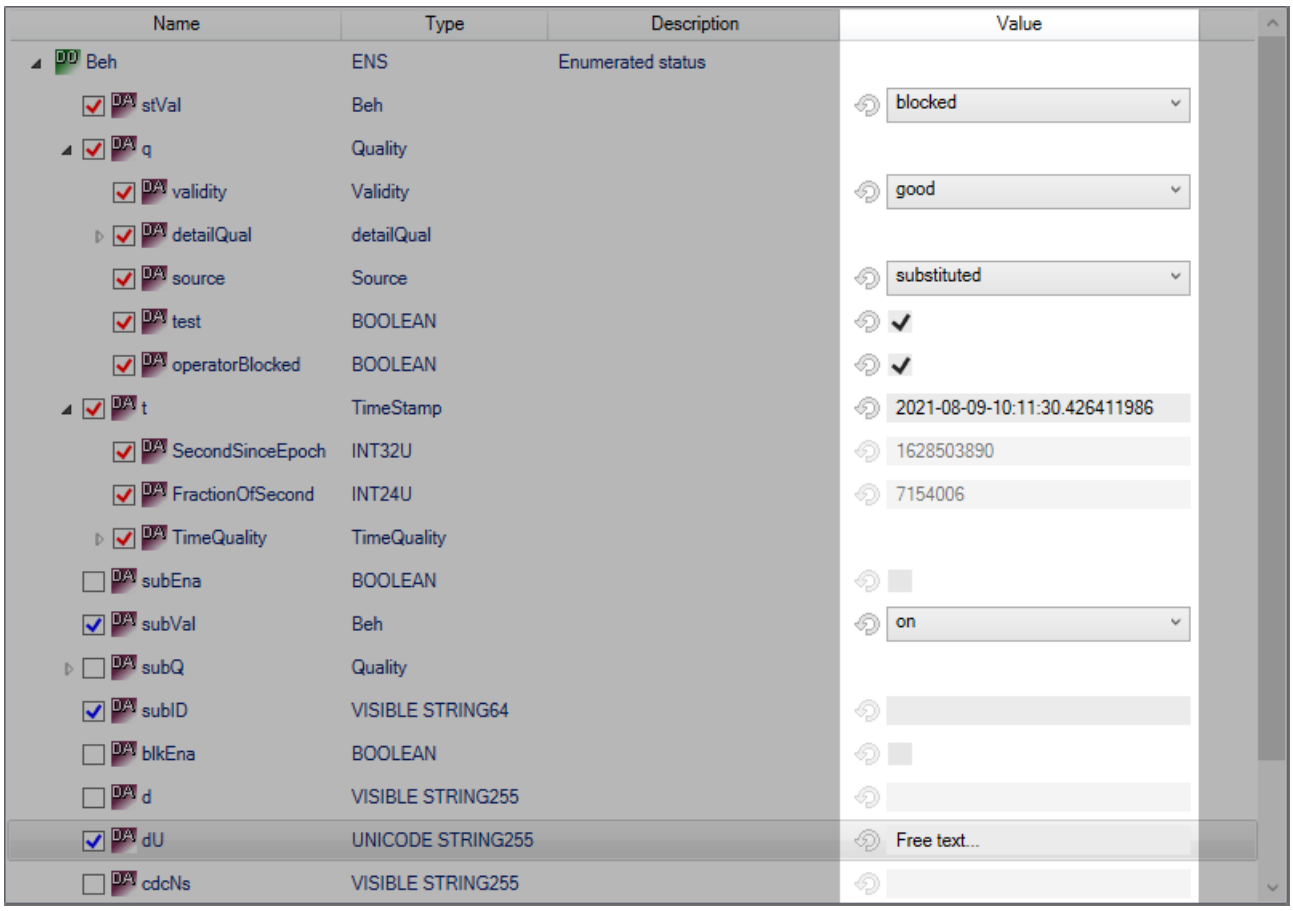
In the working area for the data model, you can show all of the objects additionally defined for the node in the standard via the toolbar (**a**) or via the context menu (**b**) using the command **Toggle visible DOs / DAs**.



In addition, all objects are given a checkbox with which the optional objects can be added or deselected. The checkbox tick is blue for optional objects and red for required objects. Required objects can also be switched on and off via a command in the context menu of the respective object. However, in the interests of the standard this is not advisable.

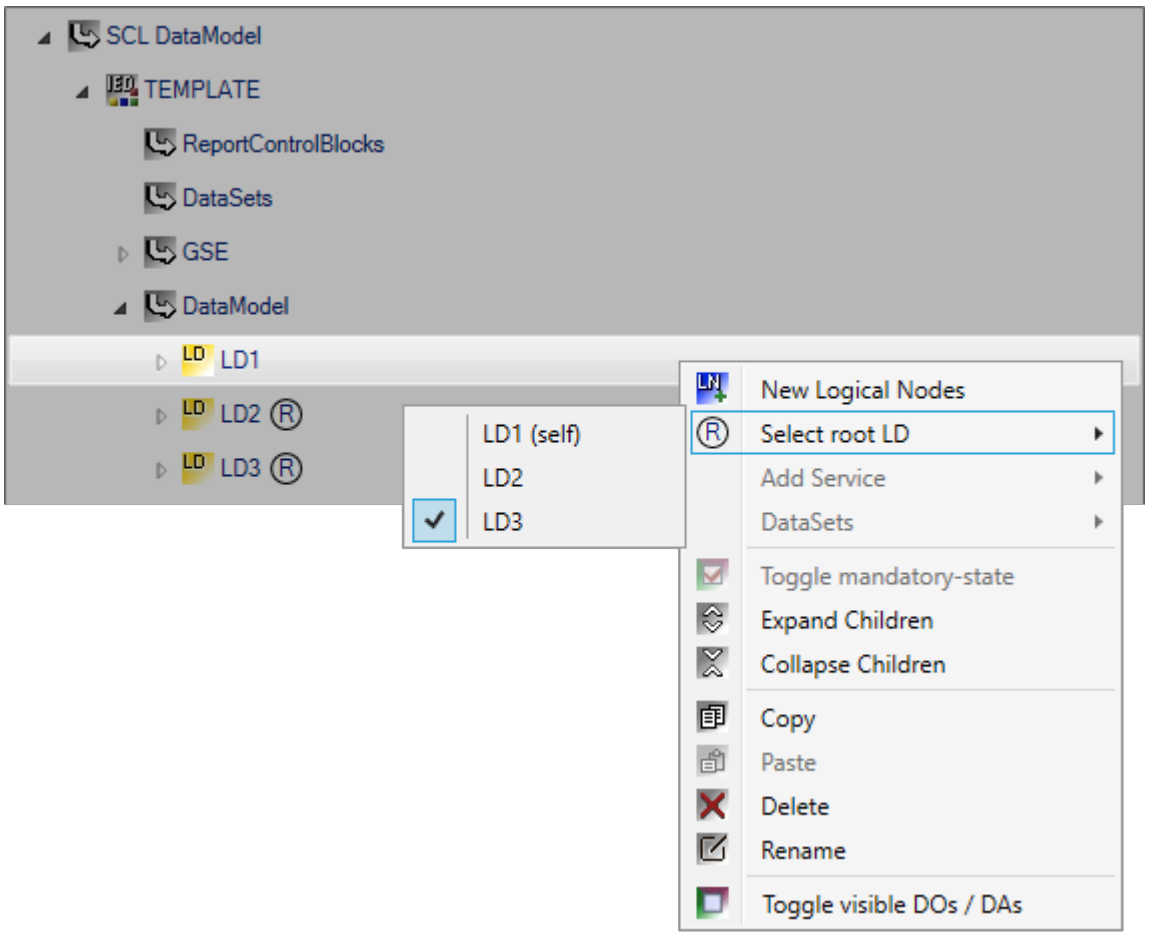


Default values can be set for the selected attributes in the Value column. To avoid unintentional entries, this is not possible for deselected attributes. Many objects offer enums, which enable a selection via a drop-down box. Numbers and texts can be entered directly. These values are later exported to the PLC code. The adjacent reset button allows you to delete the configured value.



Logical Devices Management Hierarchy

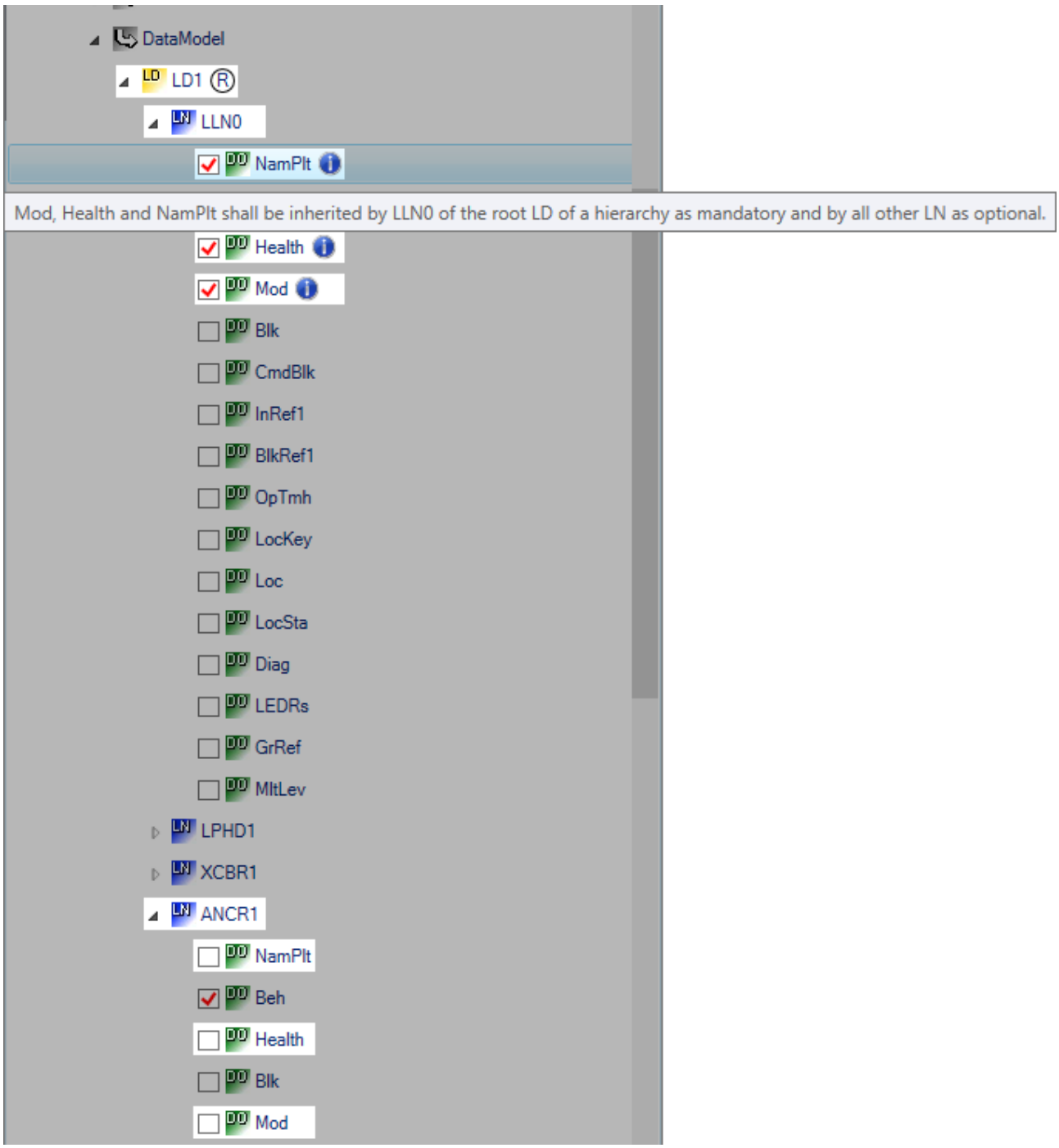
To model complex functions, it may be necessary to group several logical devices into a management hierarchy. This is built up with the attribute **setSrcRef** of the object **GrRef** in the Logical Node **LLN0** by optionally referencing a Logical Device via the value of the attribute. If no other logical device is referenced in a logical device, it is itself considered a root logical device. Representative for this, in the TwinCAT Telecontrol Configurator the name of the own Logical Device is provided with a **(self)** label to make this clear. The root logical device can be additionally specified in the context menu of the logical device.



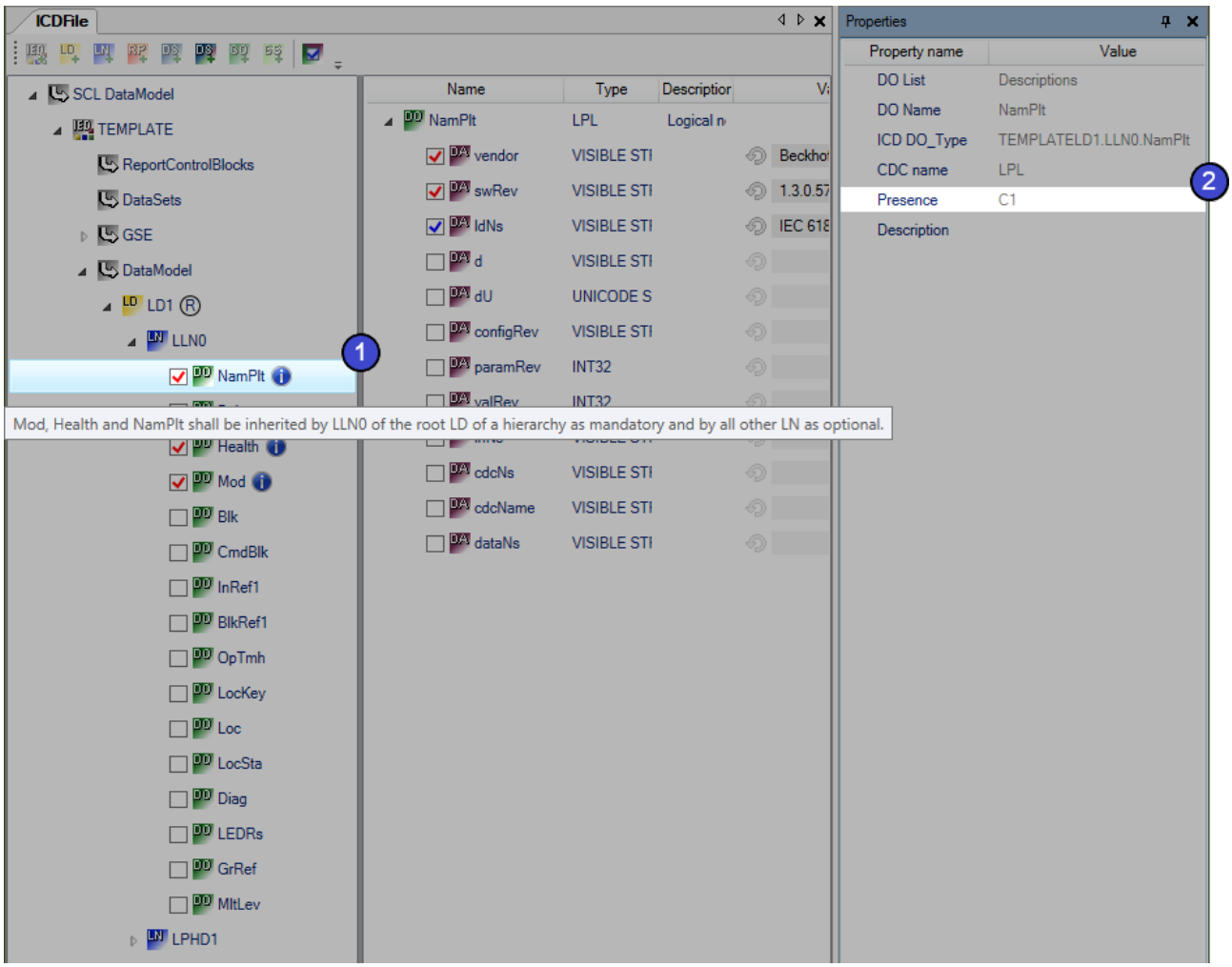
Condition-dependent data objects

In addition to the required and optional objects, there are also the **condition-dependent data objects**, which may be present, be added or not be present in the data model, depending on the condition. Condition-dependent data objects are handled differently in the Configurator, depending on the condition. If a condition is not automatically handled by the Configurator, a message is displayed at the respective object. In addition, a ToolTip is displayed which contains the respective condition description.

The following example shows how the condition C1 of the **Common LN** Logical Node was integrated. This Logical Node represents the basis of all Logical Nodes and is inherited from them. In the Logical Node **LLN0**, the three objects **NamPlt**, **Health** and **Mod** are added as required according to the standard. In the **ANCR1** Logical Node, however, they are listed only as optional, since the **LLN0** Logical Node of the Root Logical Device is not involved.

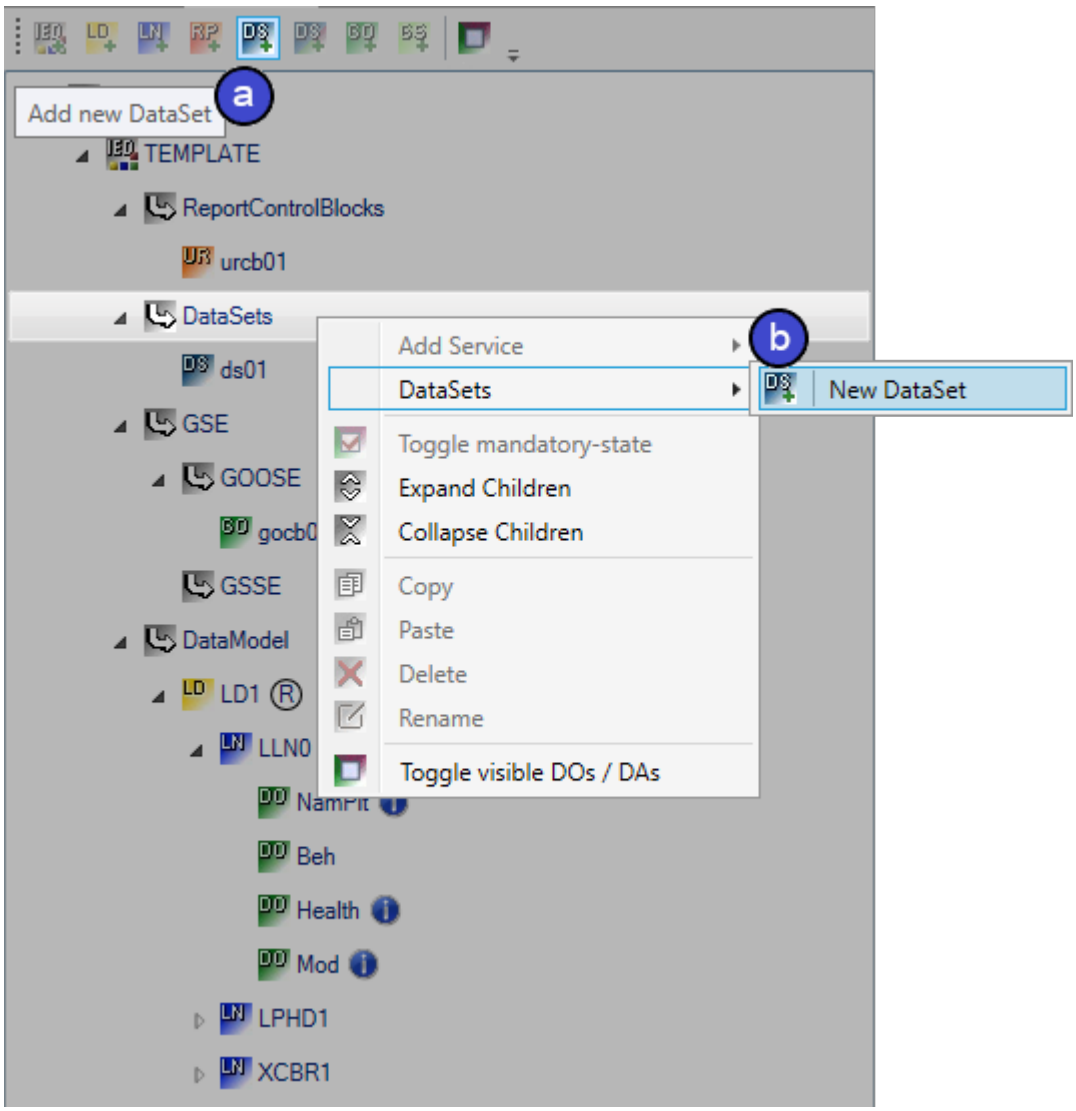


In the Properties window of the respective data object, you can see the condition designation under which you can find the condition in the IEC61850 standard.

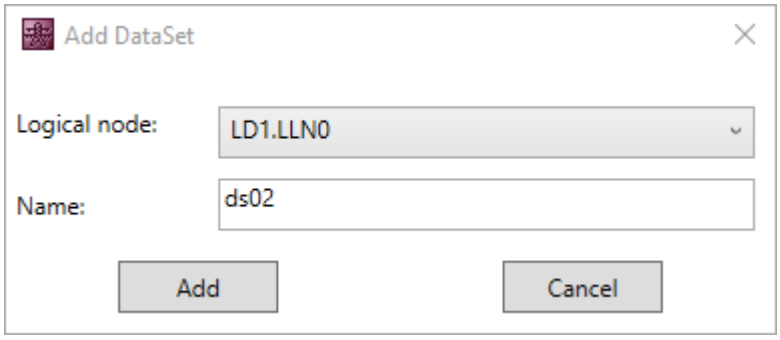


5.6 DataSets

DataSets can be added on the left-hand side in the data model tree. On the one hand via the toolbar (**a**) and on the other hand via the context menu (**b**). A dialog then opens in which you can enter the name of the DataSet. After confirming the dialog, the **DataSet** is listed in the DataSetlist.

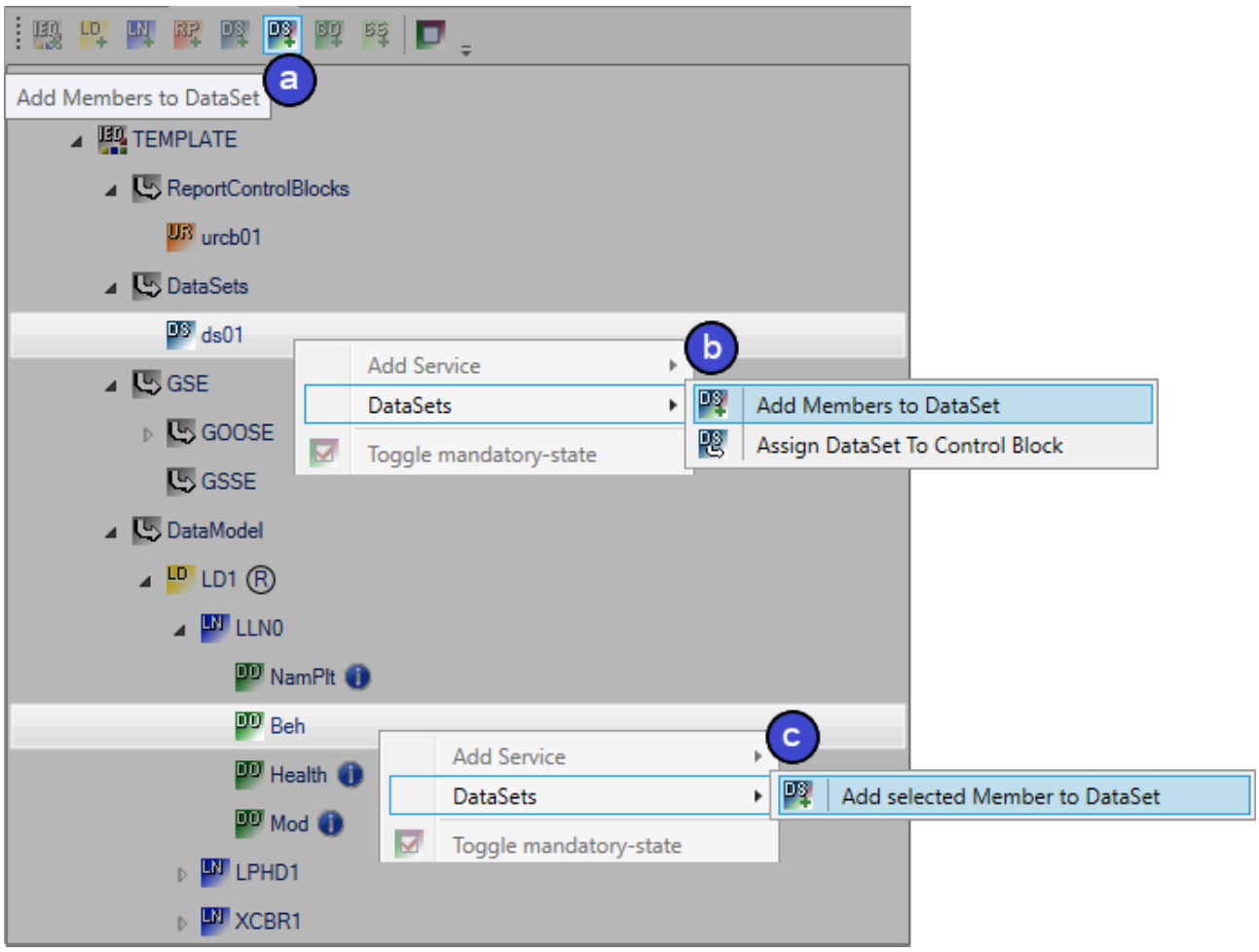


Create DataSet

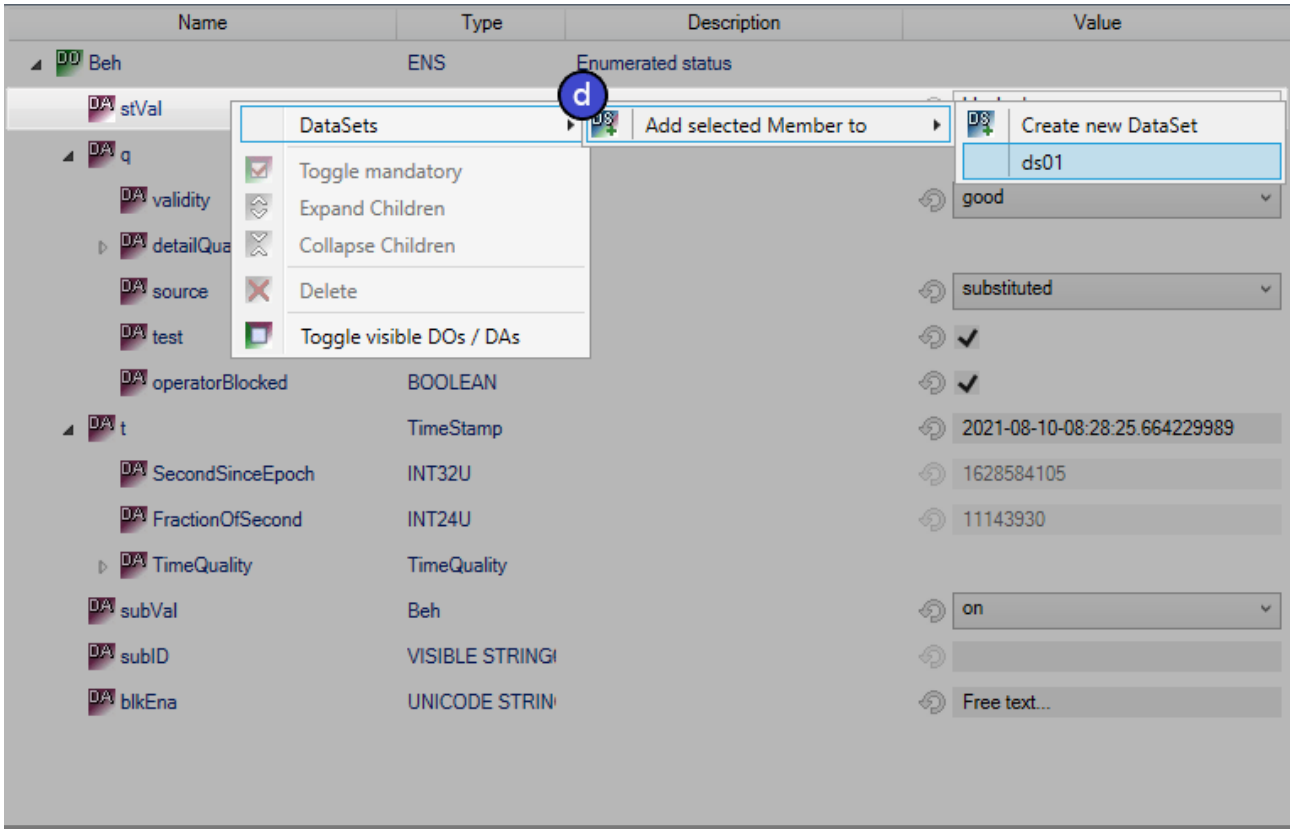


Adding elements to the DataSet

DataAttributes and Functional Constraints can be added to the DataSet via the toolbar (**a**) or via the context menu. Depending on which element is selected in the data model tree, this can be done in several ways.

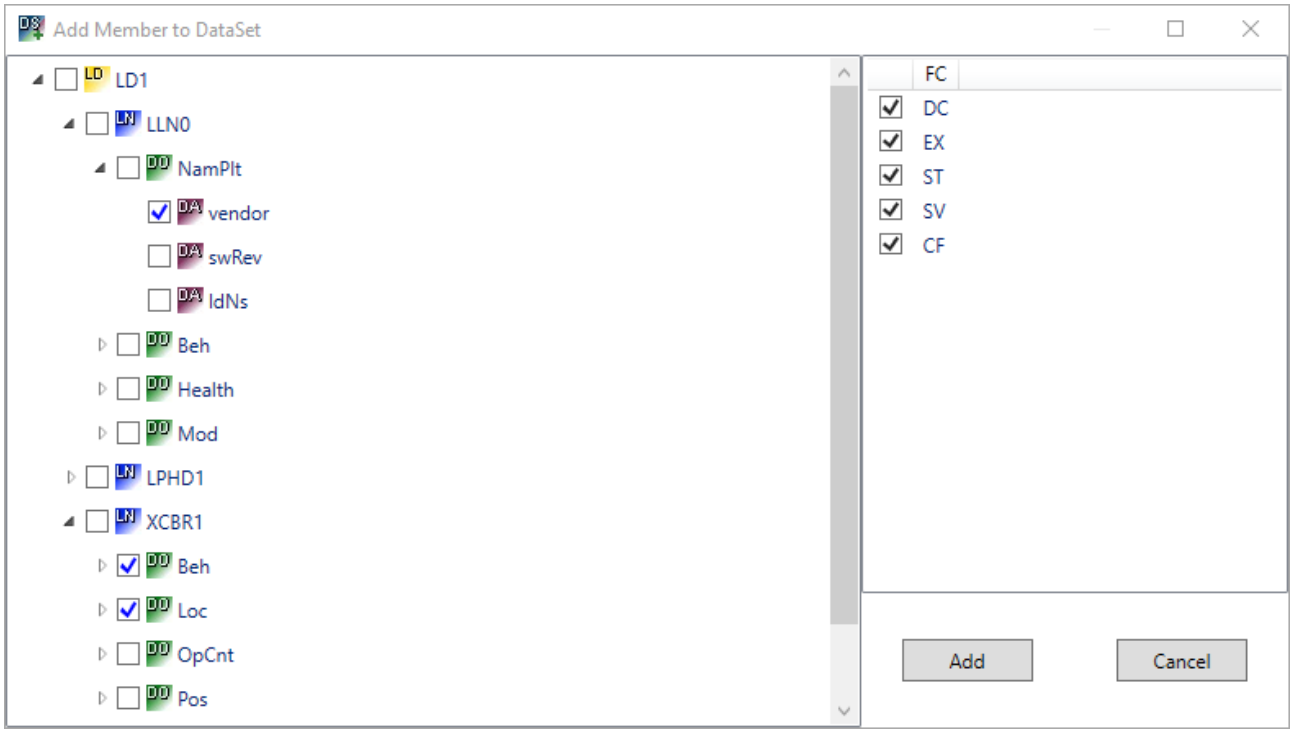


With DataAttributes, it is possible to complete the assignment directly via the context menu (**d**).



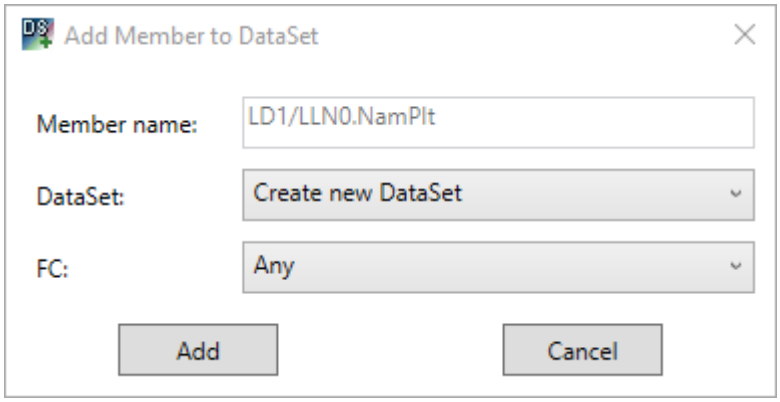
Configure DataSet independently (b)

With the following dialog, it is possible to configure a DataSet clearly by ticking the desired elements. You can filter the displayed Functional Constraints using the right column.



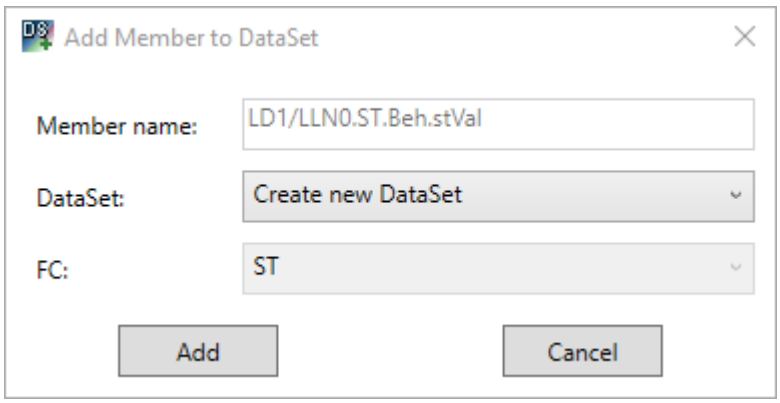
Add Functional Constraints (c)

In the following dialog, you can select to which new or existing DataSet the Functional Constraint is to be added. You can also choose to add a specific Functional Constraint or all Functional Constraints present in the DataObject.

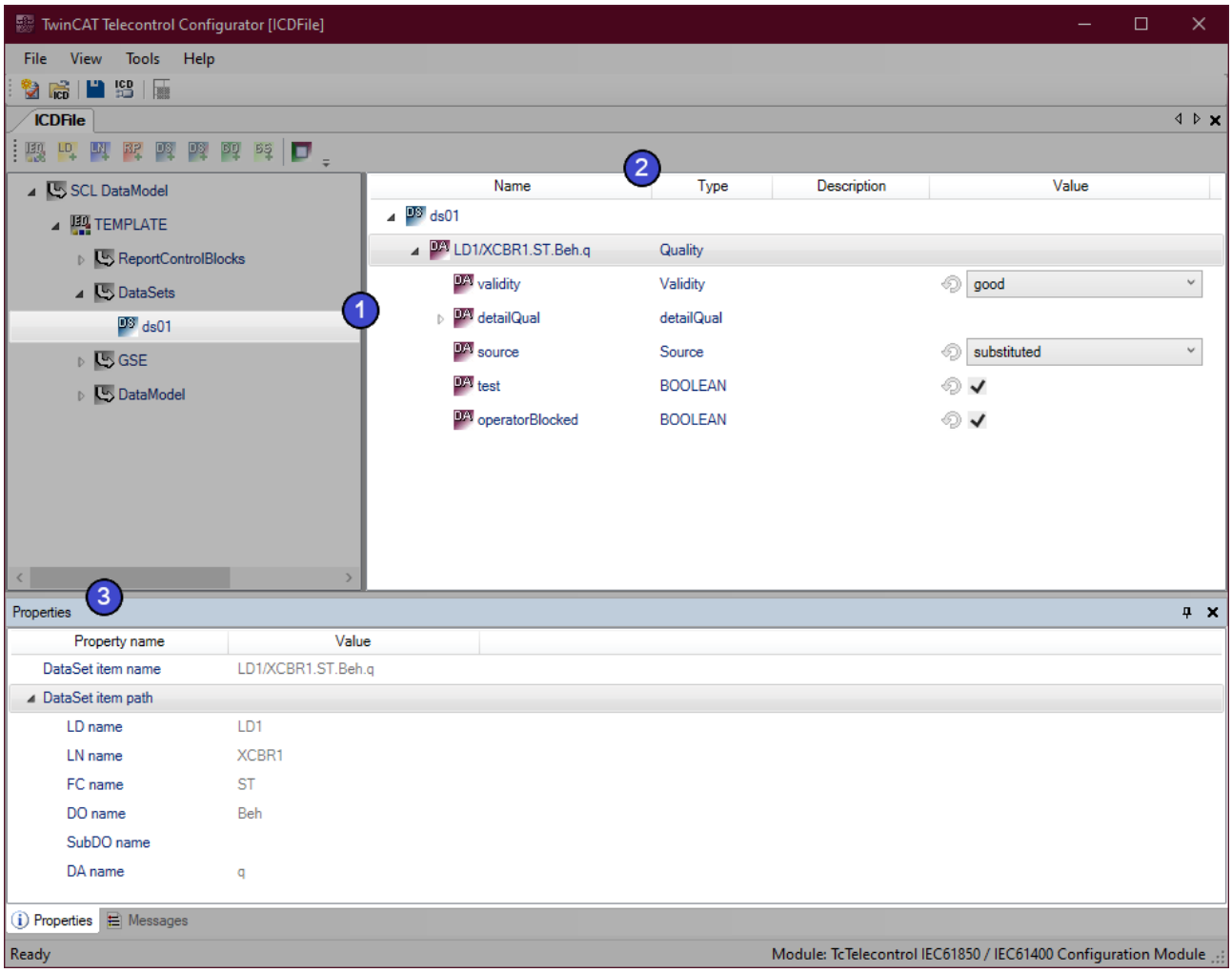


Add DataAttributes (d)

In addition to the configuration via the context menu, you can also add a DataAttribute to the DataSet via the toolbar. This opens the following dialog, where you can select to which new or existing DataSet the DataAttribute is to be added.



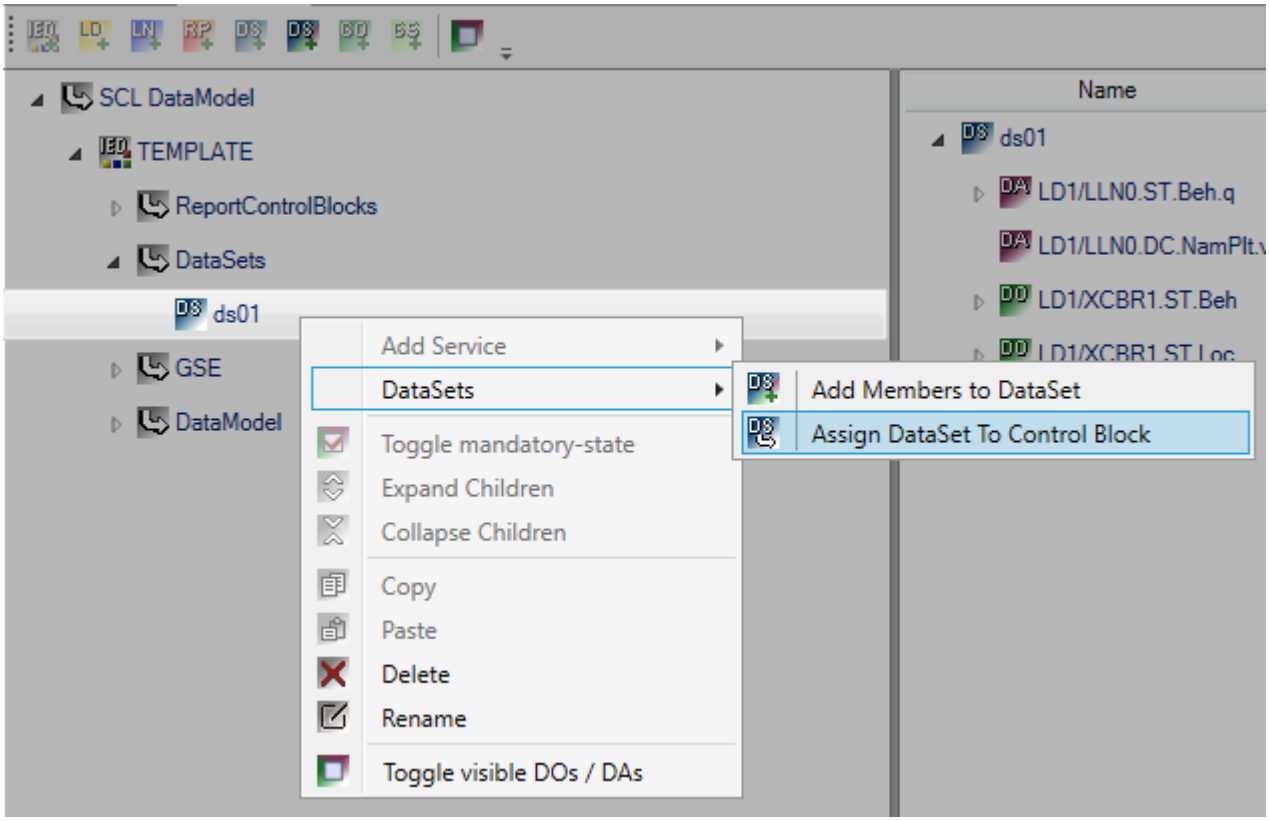
If you select a **DataSet** from the DataSetlist from (**1**), all data attributes and Functional Constraints associated with the **DataSet** are displayed in the right-hand window (**2**).



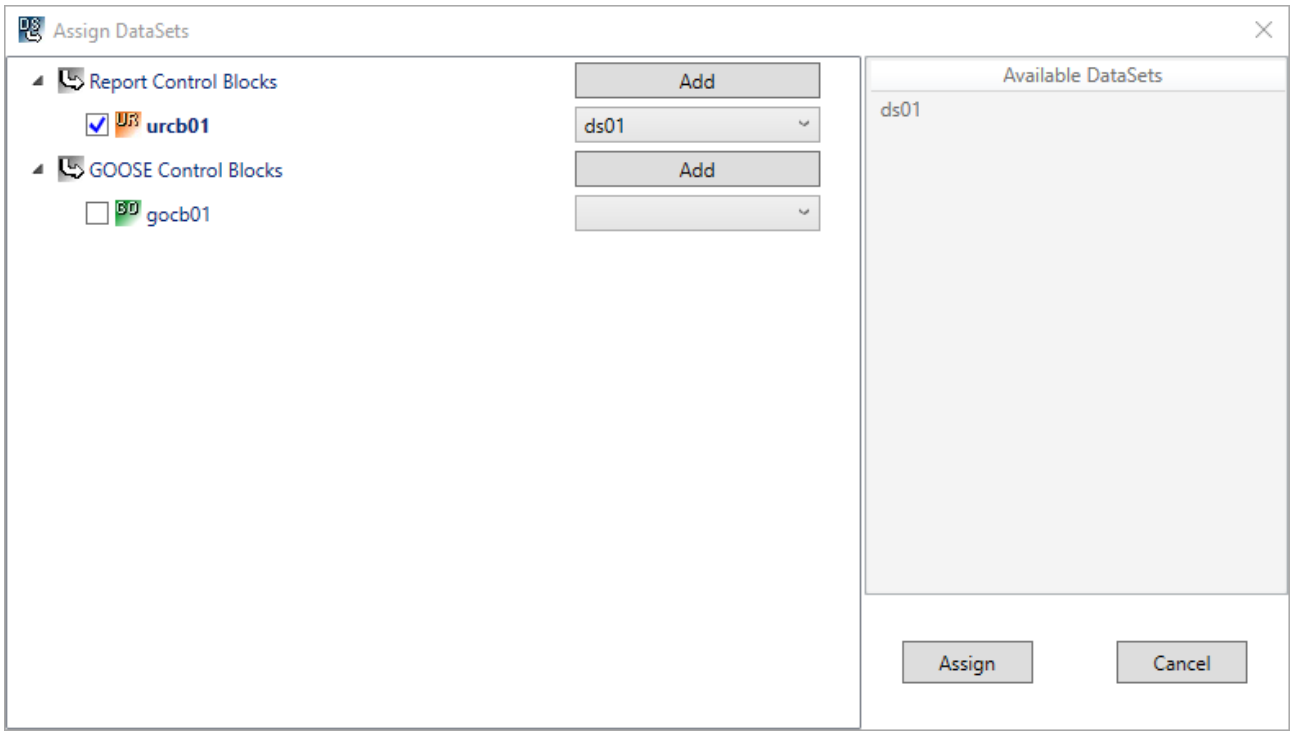
The source of the selected attribute is documented in the properties (**3**).

Assign DataSets to control blocks

DataSets can be linked directly in the properties of the respective control block. On the other hand, you can assign the respective DataSet to multiple control blocks directly via the associated context menu.

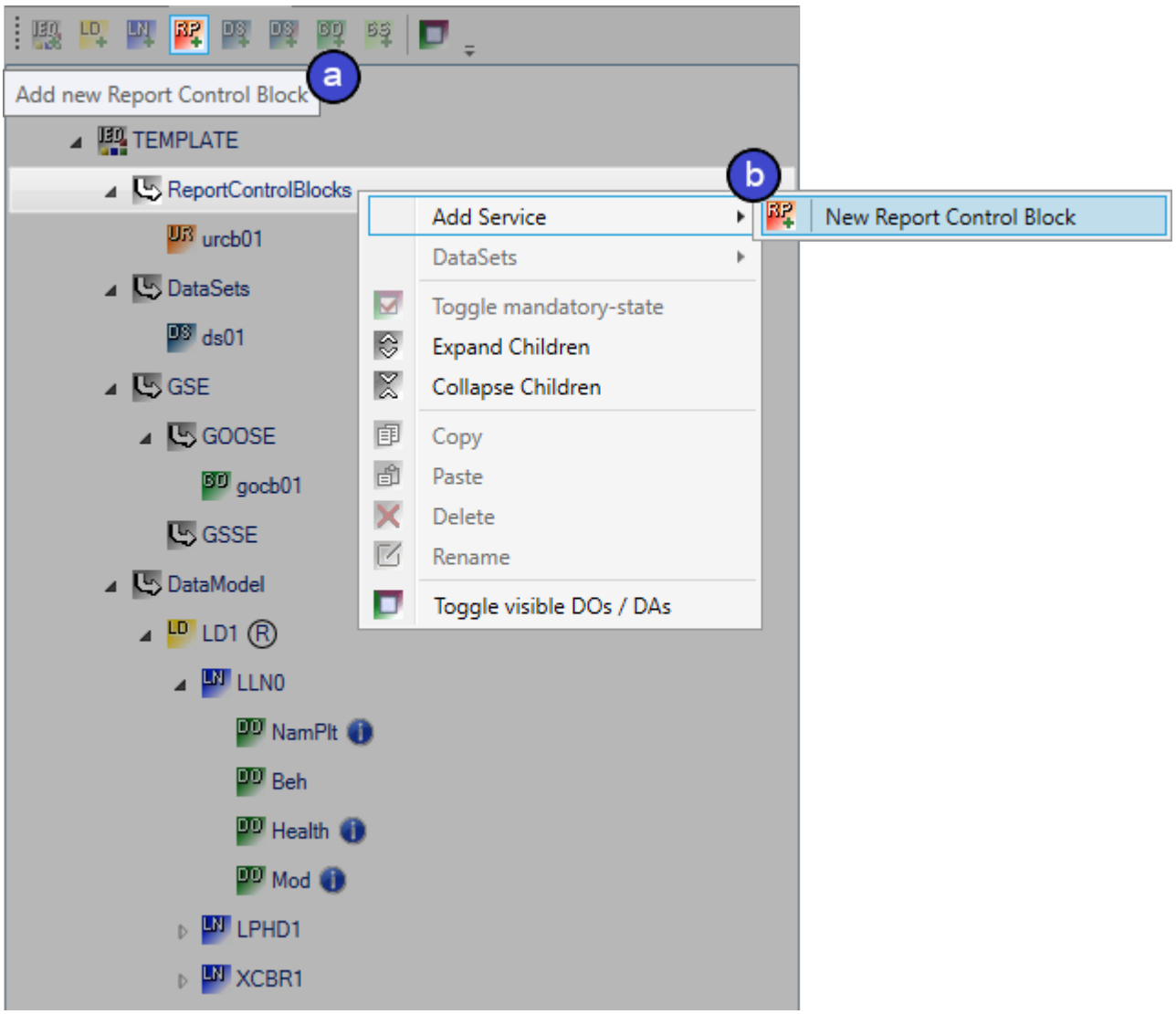


The following dialog shows all existing control blocks and the respective DataSet currently assigned. It is also possible to add new control blocks. To assign the DataSet selected in the data model tree to a control block, you can tick it directly. Alternatively, it is possible to assign other DataSets that already exist to the respective control blocks via the selection field. All currently available DataSets are displayed in the right column.

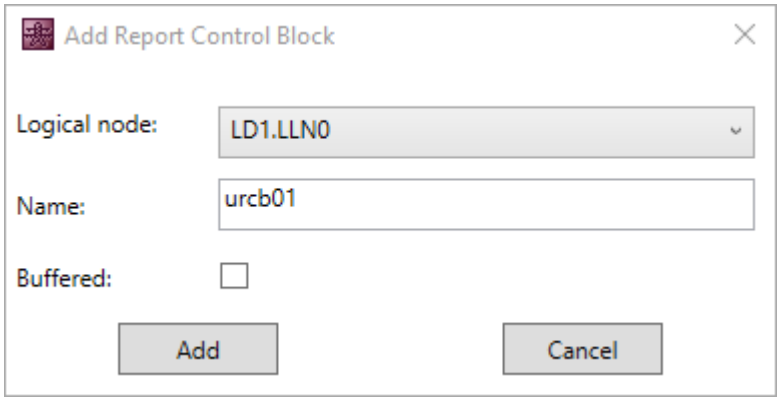


5.7 Reporting

Report Control Blocks can be generated via a button in the toolbar (**a**), or alternatively from the context menu in the data model tree under ReportControlBlocks (**b**).

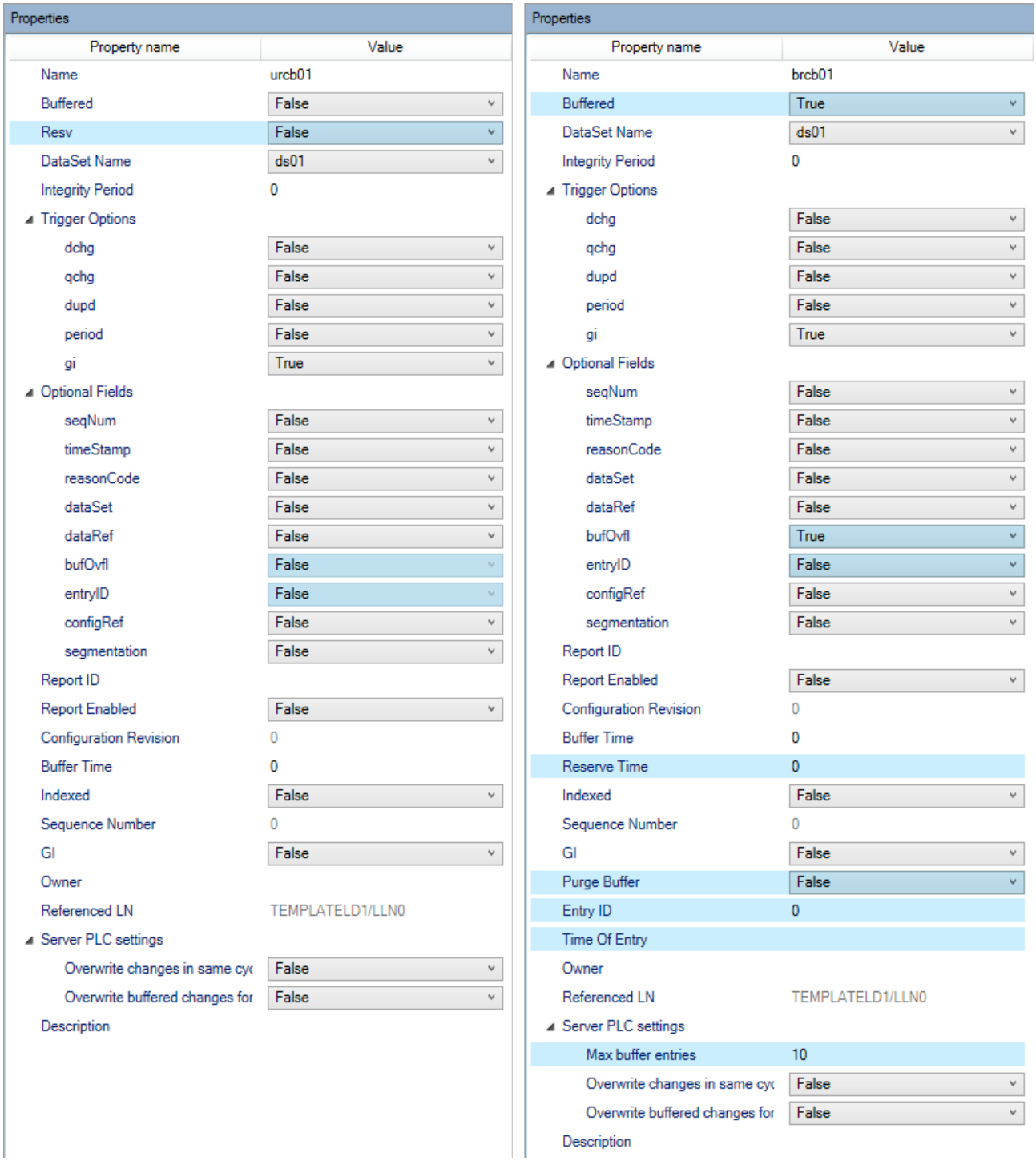


It is possible to create **Buffered** or **Unbuffered** report control blocks. The assignment to a Logical Node can be selected in the dialog, as can an appropriate name.



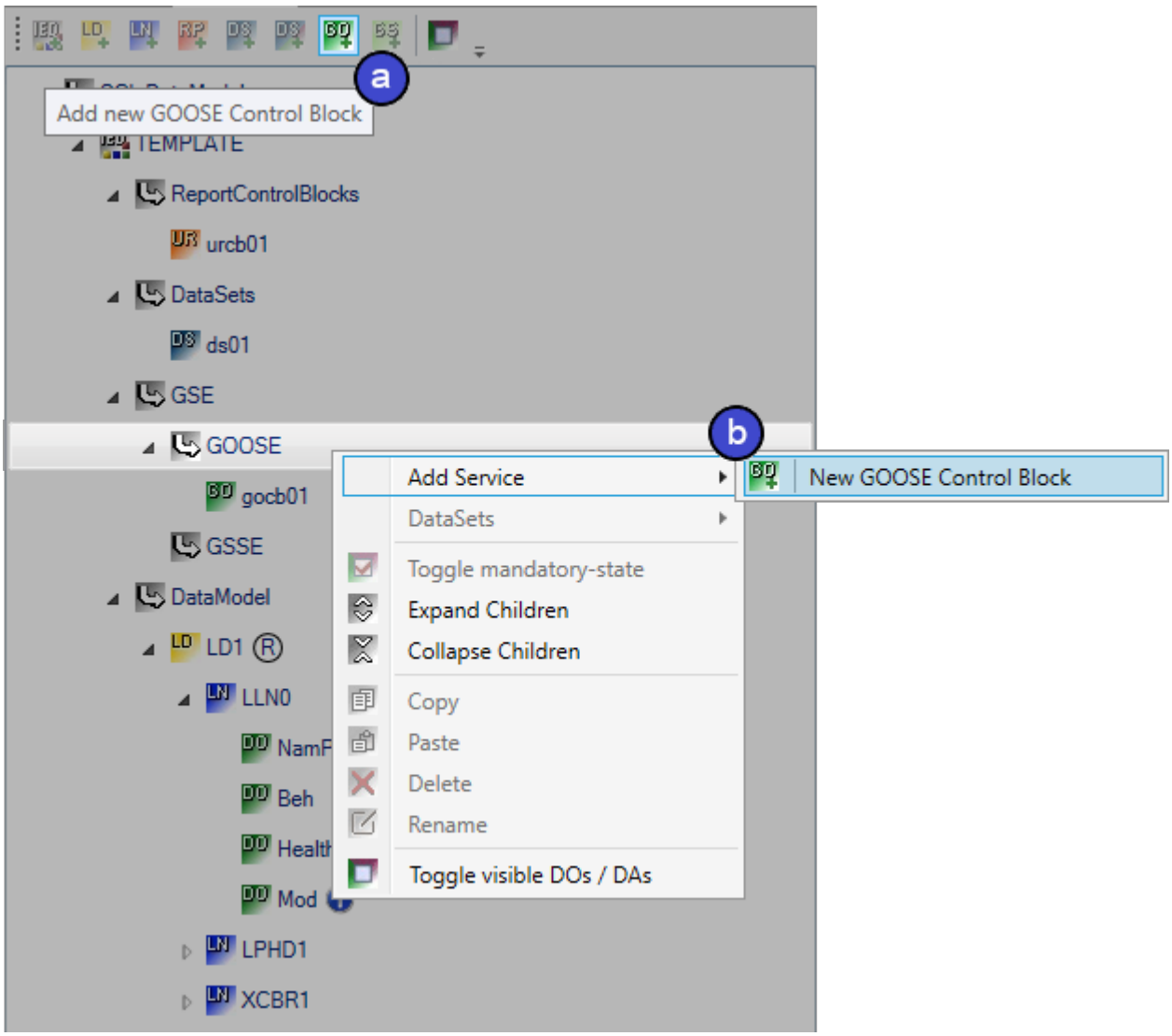
If you select a created Report Control Block in the tree, you can make settings in the properties. Depending on whether it is an unbuffered or buffered Report Control Block, different default settings are available.

In addition to the default settings, there are settings for the PLC server. Further information can be found in chapter Server - Buffered Reporting [▶ 729].



5.8 GOOSE

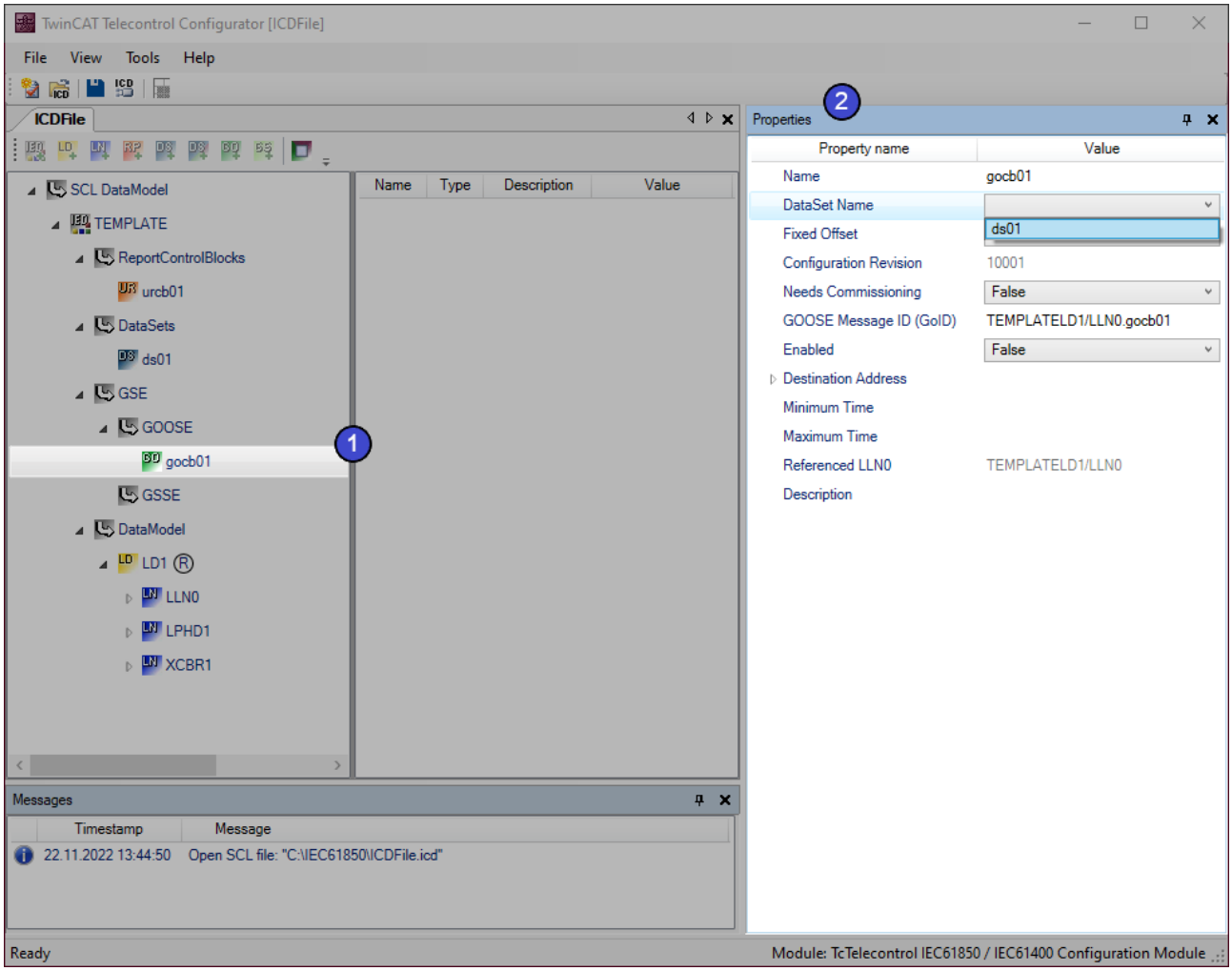
GOOSE Control Blocks can be generated via a button in the toolbar (**a**), or alternatively from the context menu in the data model tree under GOOSE (**b**).



The assignment to a Logical Node can be selected in the dialog, as can a corresponding name.

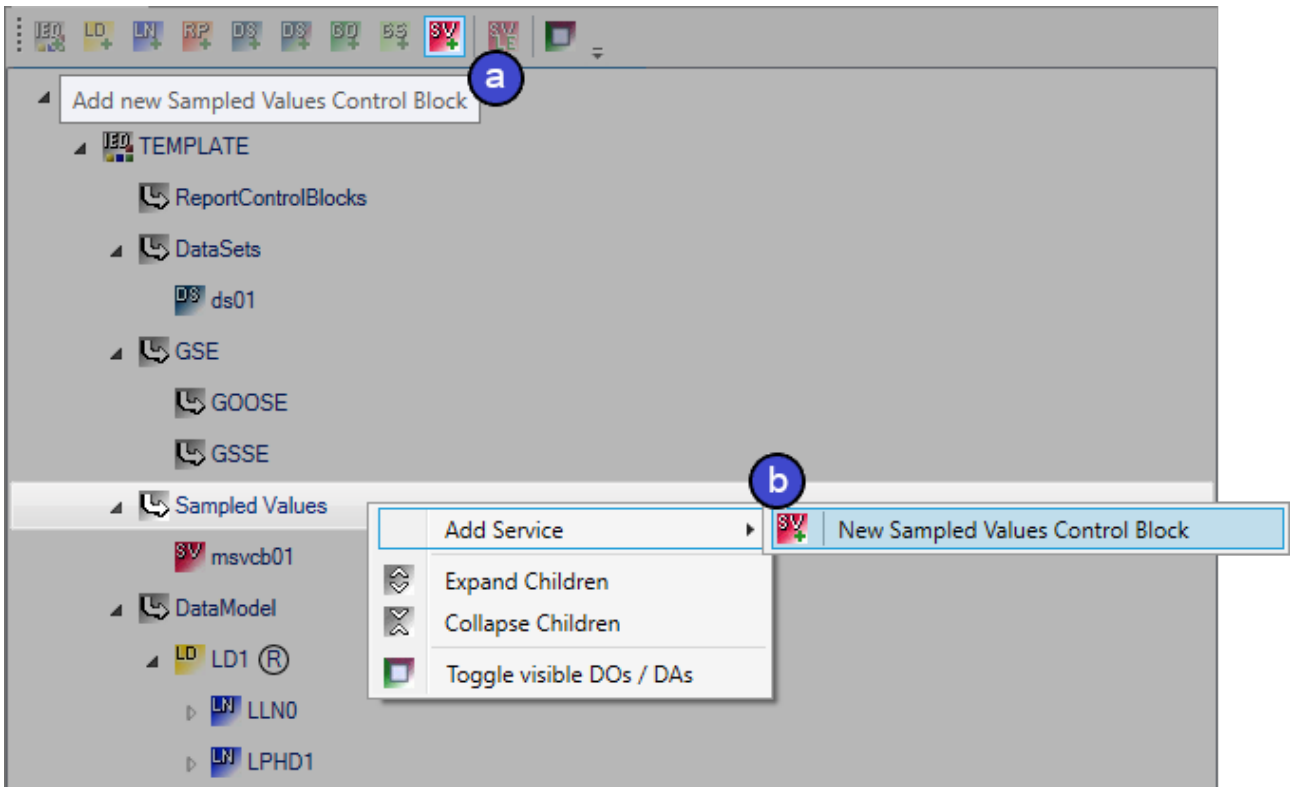


If you select the GOOSE block created in the tree (**1**), you can make GOOSE-specific settings in the Properties and thus implement the link to one of the created DataSets (**2**), for example.

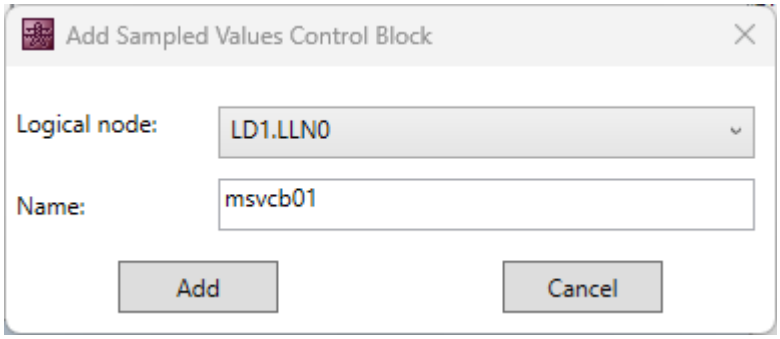


5.9 Sampled Values

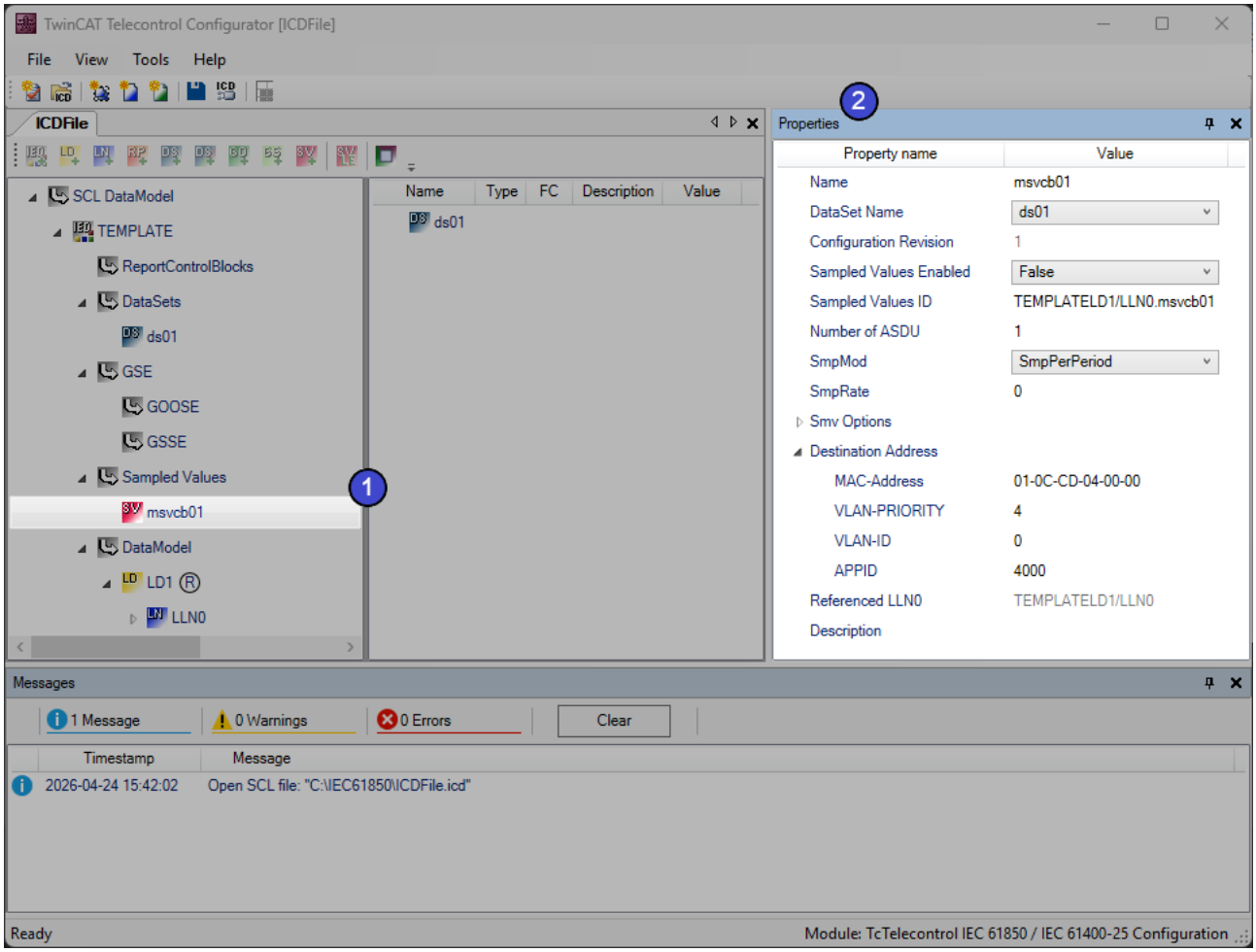
Sampled Values Control Blocks can be generated via a button in the toolbar (**a**), or alternatively from the context menu in the data model tree under Sampled Values (**b**).



The assignment to a Logical Node can be selected in the dialog, as can a corresponding name.

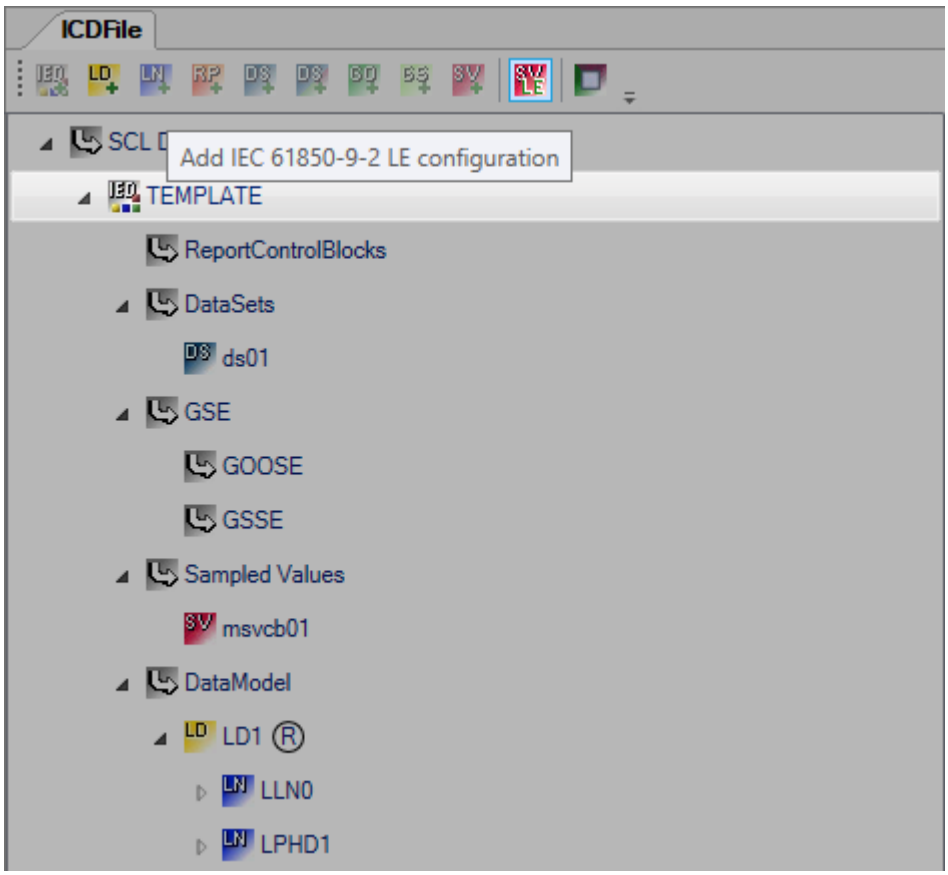


If you select the Sampled Values block created in the tree (**1**), you can make Sampled Values-specific settings in the Properties and thus implement the link to one of the created DataSets (**2**), for example.

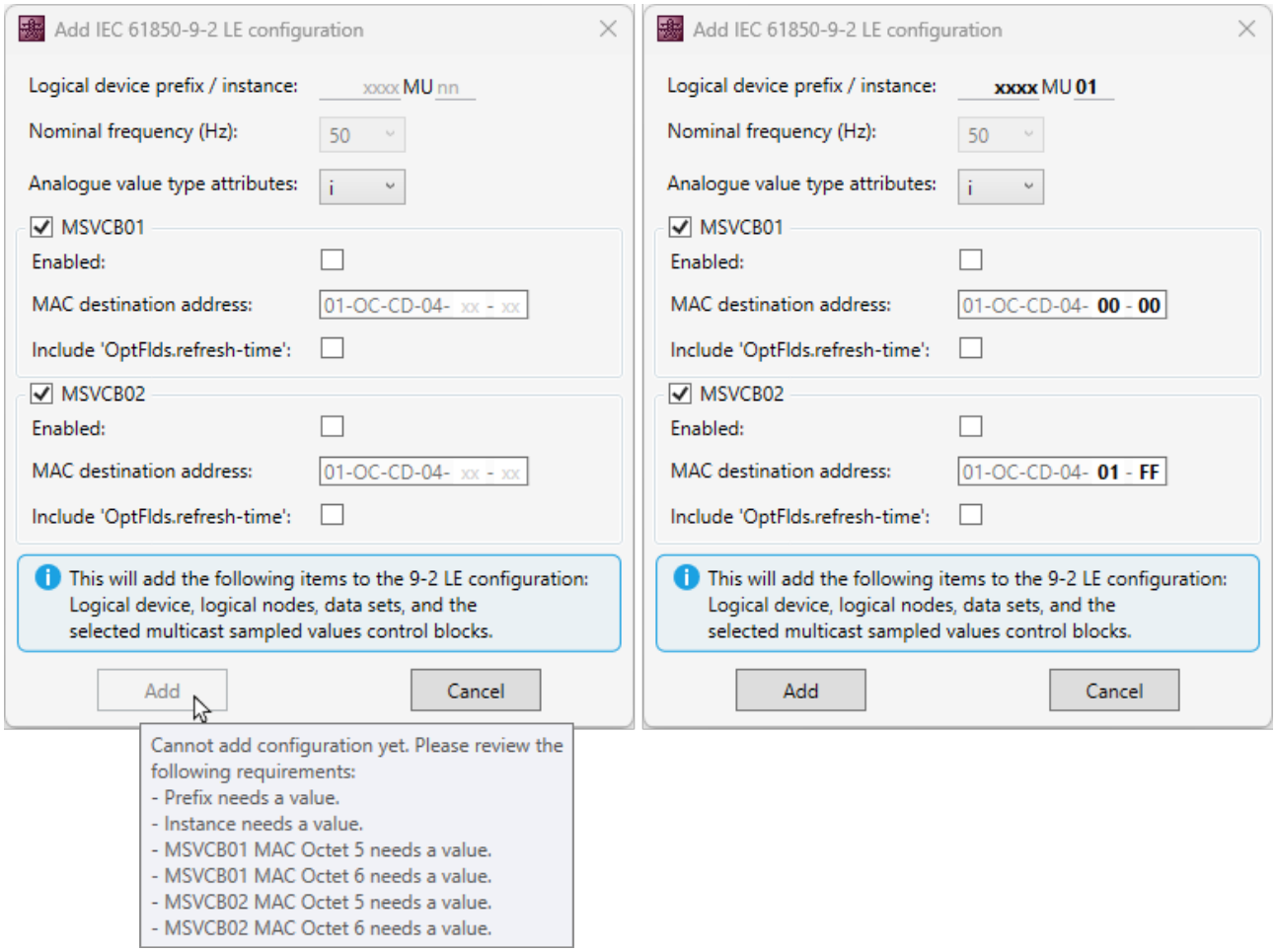


Light Edition (9-2-LE)

If you select an existing IED in the tree, you have the option of adding a finished configuration in accordance with IEC 61850-9-2-LE (Light Edition) using the button in the toolbar.

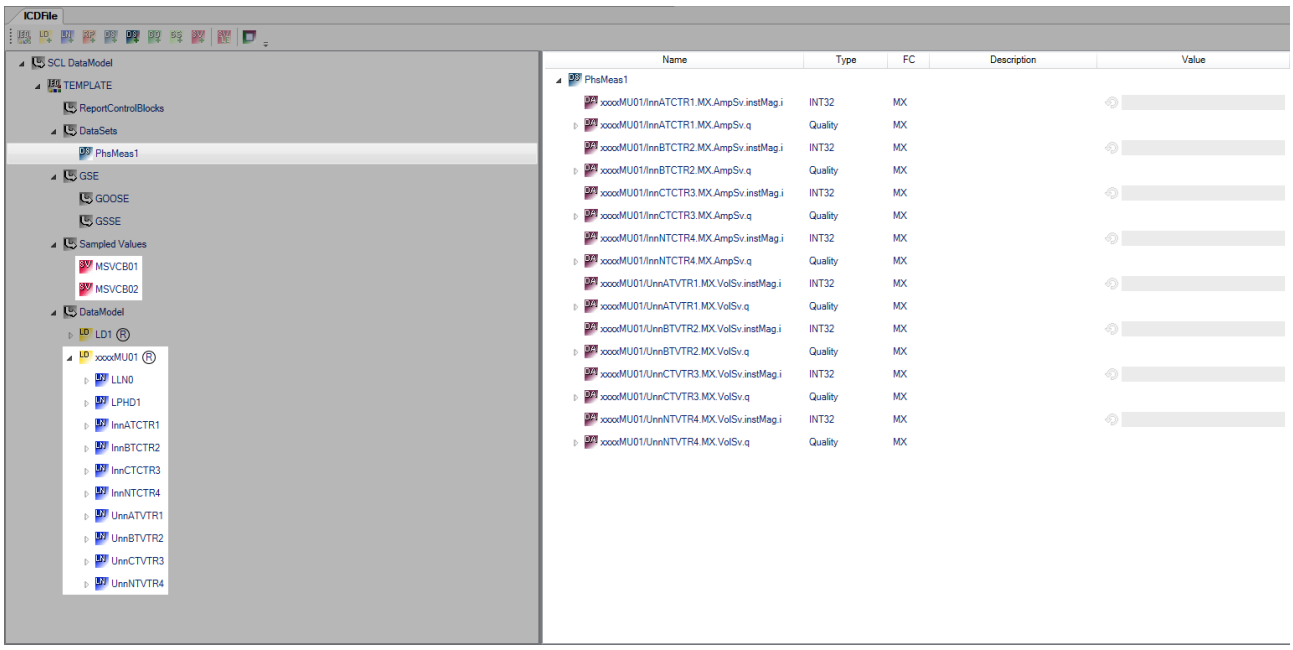


In the following dialog, you have the option of setting the standard configurable settings. The settings are validated. If a value is invalid, a note indicating the valid range of values is displayed in the corresponding tooltip. In addition, the tooltip for the “Add” button displays all the settings that still need to be configured. Dialog when starting (left) and after entering some sample values (right):



Confirm the dialog to add a complete configuration in accordance with IEC 61850-9-2-LE (Light Edition) to the current project. This contains a logical device preconfigured in accordance with the standard, a preconfigured data set, and the desired Sampled Values blocks.

If you want to remove the entire configuration again, you only need to delete the logical device.



You can find an example of the TwinCAT IEC 61850 implementation in chapter: SV Publisher/Subscriber according to UCA 61850-9-2 Light Edition - Transmission of sampled measured values [▶ 760]

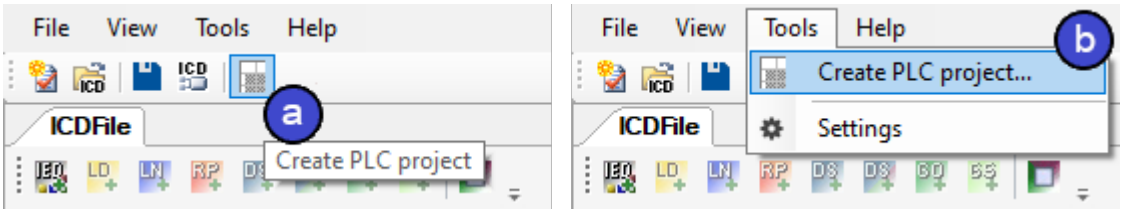
5.10 Automatic code generation

Automatic PLC code generation

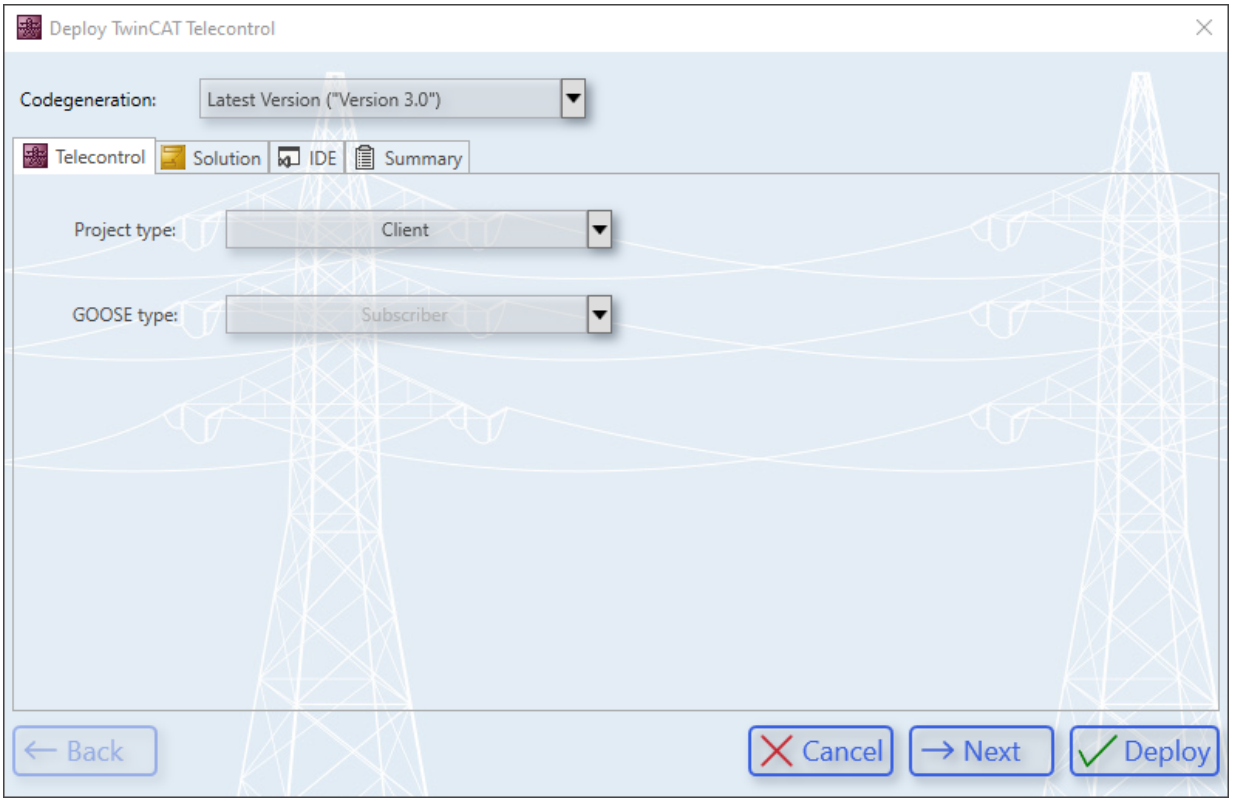
The code generated by the TwinCAT Telecontrol Configurator is merely an application example, which can also be adapted from version to version. There are several possible ways of implementing a functional application with the IEC 61850 function blocks.

✓ Automatic PLC code generation from the TwinCAT Telecontrol Configurator is carried out via the TwinCAT Automation Interface.

1. Pressing the button **Create PLC Project** in the toolbar(**a**) or in the Tools menu of the configurator(**b**) opens the Deploy TwinCAT Telecontrol dialog.

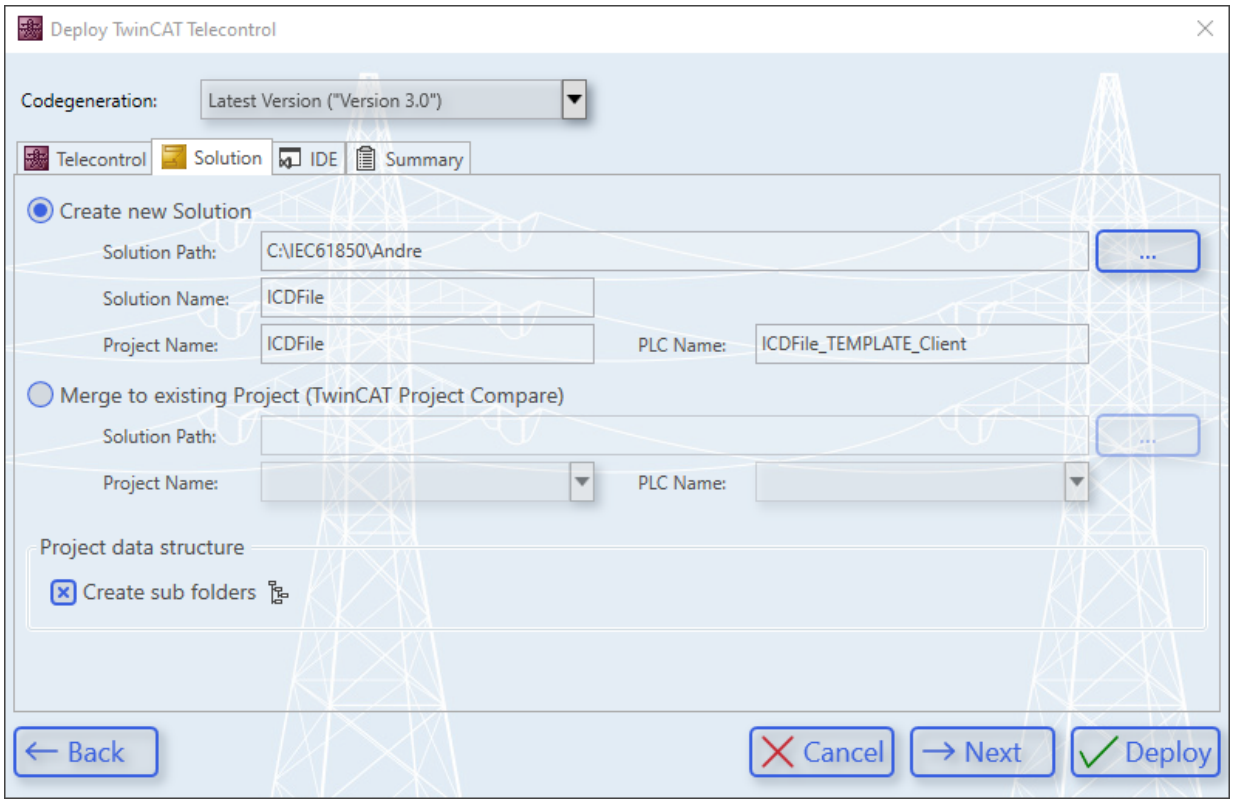


2. In the **Telecontrol** tab you have the possibility to choose between client and server configuration (**Project type**). If GOOSE Control Blocks are present in the project to be generated, the associated **GOOSE type** is also automatically selected.

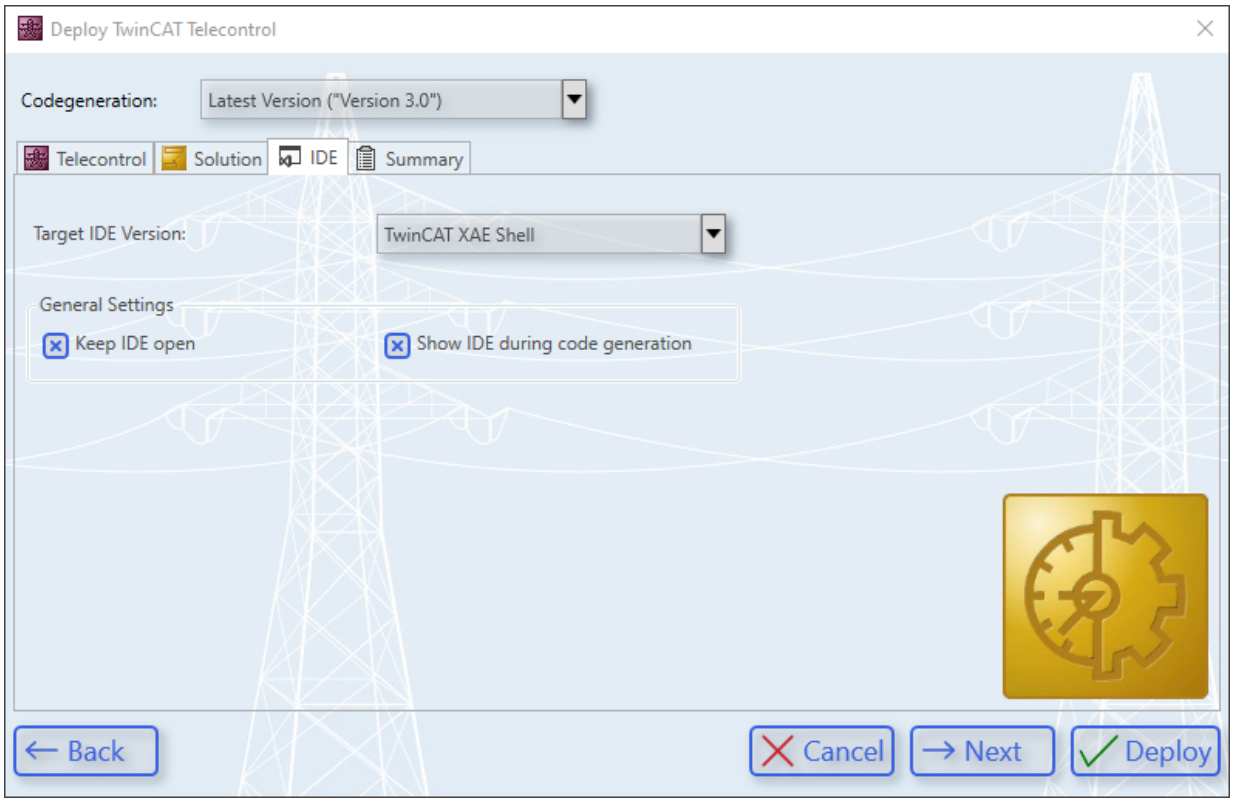


3. In the **Solution** tab, select a path (**Solution Path**) and assign the names for **Solution**, **XAE Project** and **PLC Project**. You can either create a new solution or integrate the changes into an existing solution using the TwinCAT Project Compare tool. To do this, first select the option **Merge to existing Project** and then an existing SLN file with the adjacent button "**...**".
Under **Project data structure** you can additionally select whether subfolders (**Create sub folders**) should be generated. By default, these are generated to map the configured data model in the folder hierarchy.

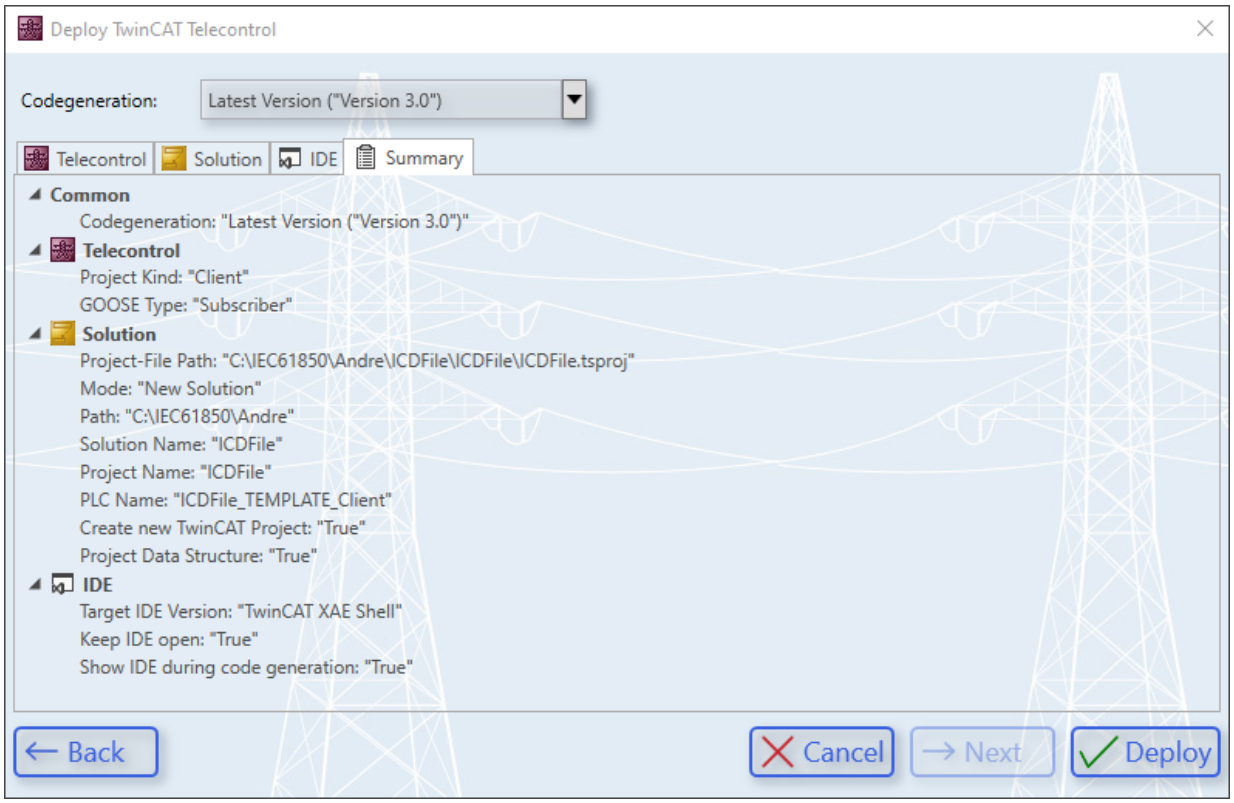
If you use longer names for the components of the data model, it may be necessary not to generate subfolders in order not to exceed the maximum path length.



4. In the **IDE** tab you have the possibility to use a development environment of your choice for PLC code generation (**Target IDE Version**). The prerequisite for this is an existing TwinCAT integration in the development environment.
   In addition, you can set whether the development environment should remain open after PLC code generation is complete (**Keep IDE open)** and whether the development environment should be visible during PLC code generation (**Show IDE during code generation**).

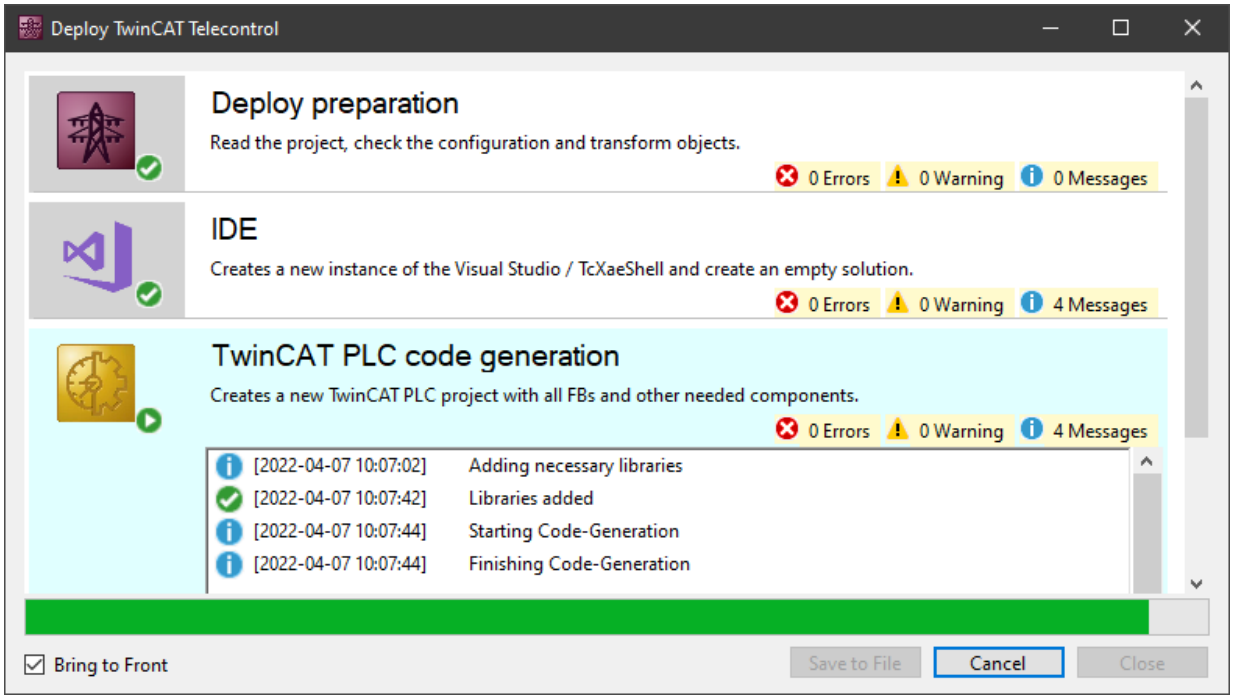


5. The last, concluding tab **Summary** provides a brief overview of the settings made.
   The PLC code generation is started as soon as you press the **Deploy** button.

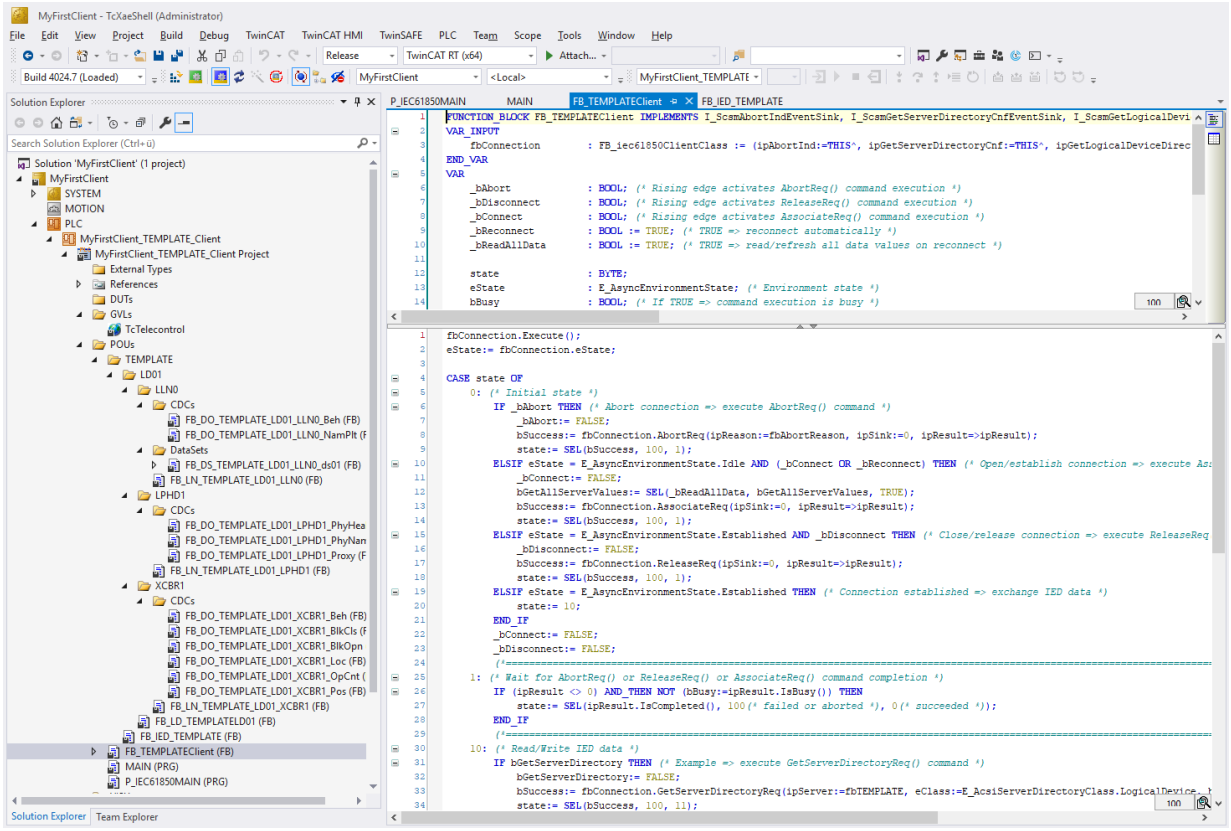


⇨ During PLC code generation, the progress is displayed in the automatically opening log window.

6. To get more detailed information about the individual categories, such as IDE or TwinCAT PLC Code Generation, you can click on the respective category.
   With the Bring to Front option the log window remains permanently in the foreground.



⇨ After the generation is completed, a new instance of the selected development environment is opened. The PLC code is completely transparent so that the application can be written in the best possible way.



⇨ Regardless of whether PLC code is generated for an IEC 61850 client or server, the Global Variable List TcTelecontrol forms the entry point.

```
VAR_GLOBAL
    ipCreator          : I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=2).SetGui
Ver(major:=1, minor:=0, build:=93, revision:=10);
    fb[IEDName]        : FB_IED_[IEDName];
    fb[IEDName]Client  : FB_[IEDName]Client := (fbConnection:=(ipIED:=fb[IEDName],settings:=(sRemoteH
ost:='127.0.0.1')));
    fb[IEDName]Gse     : FB_[IEDName]Gse := (fbAdapter:=(ipIED:=fb[IEDName], settings:=(sMulticastAdd
r:='01-0C-CD-01-00-00', eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
END_VAR
```

In addition to the data model, you will find the FB_[IEDName]Client for the client, for example. This is an application function block that encapsulates the FB_iec61850ClientClass and the complete communication. Here it is particularly important to specify the IP address of the server, which you can also do in the configurator.

If you generate a server, the Global Variable List looks slightly different.

```
VAR_GLOBAL
    ipCreator                 : I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=2
).SetGuiVer(major:=1, minor:=1, build:=94, revision:=1);
    fb[IEDName]               : FB_IED_[IEDName];
    fb[IEDName]Server         : FB_iec61850ServerClass := (ipIED:=fb[IEDName], settings:=(bEnable:=T
RUE, sLocalHost:='127.0.0.1'));
    fb[IEDName]ServerSession1 : FB_[IEDName]ServerSession := (fbConnection:=(ipServer:=fb[IEDName]Se
rver, settings:=(bEnable:=TRUE)));
    fb[IEDName]Gse            : FB_[IEDName]Gse := (fbAdapter:=(ipIED:=fb[IEDName], settings:=(sMult
icastAddr:='01-0C-CD-01-00-00', eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
END_VAR
```

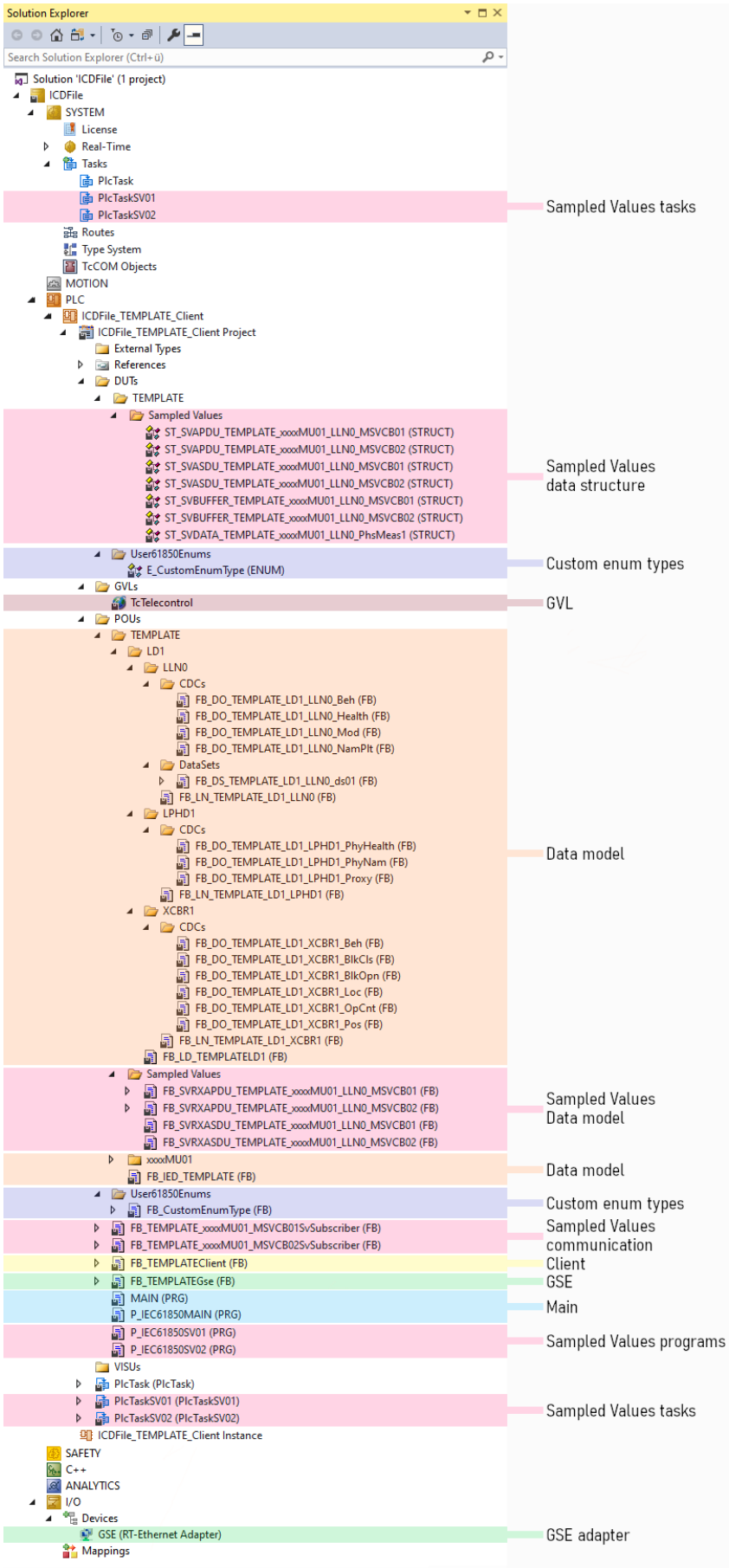
The FB_iec61850ServerClass and the sessions required for communication are not encapsulated in an application function block, since multiple sessions can also be created here in parallel for different clients.

5.10.1 Code version 2.0

The code version has been supplied since release v1.1.94.2. It is also always listed in the generated PLC code in the global variables TcTelecontrol.

```
ipCreator :I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=2).SetGuiVer(major:=1,
 minor:=1, build:=94, revision:=2);
```

5.10.1.1 Client



Custom enum types:

User-defined enum types that do not exist in the standard system.

GVL:

The global variable list instantiates and initializes the client connection instance, the IED data model function block instance, and, optionally, one or more Sampled Values and GOOSE communication block instances.

The code version that is used during code generation is stored there. The version of the TwinCAT Telecontrol Configurator is also stored there.

Datamodel:

The entire hierarchical structure of the IED data model as function blocks. Also includes the IED block that is instantiated in the Global Variable List.

Client:

This block contains a state machine, the basic states of which are used in every Client example.

The block instance is required for connection management and data exchange.

GSE / GSE Adapter:

This function block connects a network adapter of the TwinCAT control computer with the GOOSE control blocks in the data model and controls the execution of these control blocks.

To be able to use GOOSE communication, the real-time network adapter must be commissioned and linked. This is explained in RT Ethernet adapter configuration [▶ 35].

The GSE adapter is a network adapter for GOOSE and Sampled Values communication whose network status is constantly queried by the corresponding function block.

Main:

The TwinCAT PLC project contains a "MAIN" program as standard. This is called cyclically by a TwinCAT task and in turn calls the program "P_IEC61850MAIN".

The program "P_IEC61850MAIN" encapsulates the call of the Client block and of the optional GSE block, separates the IEC 61850 communication from the remainder of the PLC machine program and helps, for example, with the implementation of further Clients.

Sampled Values

The control blocks in a Sampled Values configuration each require structures, function blocks, tasks, and programs.

Each program is assigned to a single task and calls all subscriber function blocks at the same cycle time.

The Subscriber function block establishes a connection between the network adapter of the TwinCAT control computer and the corresponding Sampled Values control block in the data model and controls its execution.

To be able to use GOOSE communication, the real-time network adapter must be commissioned and linked. This is explained in RT Ethernet adapter configuration [▶ 35].

5.10.1.1.1 E_[EnumName]

User-defined enum types that do not exist in the standard system.

The maximum indices are -32768 and 32767, because it is a 16-bit integer. The naming of the enum values from the ICD file is applied only as a comment.

Syntax

Example Definition:

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_[EnumName] :
(
    e_1 := 1, (*Ok*)
    e_2 := 2, (*Warning*)
    e_3 := 3, (*Alarm*)
    ...

) SINT;
END_TYPE
```

Values

Name	Description
e_1	User-defined enum value, corresponds to "OK" in the ICD file.
e_2	User-defined enum value, corresponds to "Warning" in the ICD file.
e_3	User-defined enum value, corresponds to "Alarm" in the ICD file.
…	Any other user-defined enum values in the index range between -32768 and 32767.

5.10.1.1.2 TcTelecontrol (GVL)

The global variable list instantiates and initializes the client connection instance, the IED data model function block instance, and, optionally, one or more Sampled Values and GOOSE communication block instances.

The code version that is used during code generation is stored there. The version of the TwinCAT Telecontrol Configurator is also stored there.

Full description [▶ 27]

Syntax

Definition:

```
VAR_GLOBAL
    ipCreator         : I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=2).SetGui
Ver(major:=1, minor:=0, build:=93, revision:=10);
    fb[IEDName]       : FB_IED_[IEDName];
    fb[IEDName]Client : FB_[IEDName]Client := (fbConnection:=(ipIED:=fb[IEDName],settings:=(sRemoteH
ost:='127.0.0.1')));
    fb[IEDName]Gse    : FB_[IEDName]Gse := (fbAdapter:=(ipIED:=fb[IEDName], settings:=(sMulticastAdd
r:='01-0C-CD-01-00-00', eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
END_VAR
```

5.10.1.1.3 FB_[EnumName]

User-defined enum types that do not exist in the standard system.

Full description [▶ 103]

Syntax

Definition:

```
{attribute 'no_explicit_call' := 'Direct call not allowed for type FB_[EnumName]'}
{attribute 'call_after_init'}
FUNCTION_BLOCK FB_[EnumName] EXTENDS FB_AcsiBATypeENUMERATED
VAR
    _eValue: E_CustomEnumType;
    _eConfig: E_CustomEnumType;
    {attribute 'hide'}
    ipInit: I_AcsiCommonAttributeClass := THIS^.AddDAValue(eP:=E_AcsiP.Actual, in:=_eValue).AddDAVal
```

```
ue(eP:=E_AcsiP.Config, in:=_eConfig).AddDAType(in:=THIS^.T(in:=E_AcsiBasicType.Enum));
END_VAR
```

5.10.1.1.4 FB_[IEDName]Client

This block contains a state machine, the basic states of which are used in every Client example.

The block instance is required for connection management and data exchange.

Full description [▶ 27]

Syntax

Declaration:

```
FUNCTION_BLOCK FB_[IEDName]Client
VAR_INPUT
    fbConnection              : FB_iec61850ClientClass;
END_VAR
VAR
    _bAbort                   : BOOL;
    _bDisconnect              : BOOL;
    _bConnect                 : BOOL;
    _bReconnect               : BOOL := TRUE;
    _bReadAllData             : BOOL := TRUE;
    state                     : BYTE;
    eState                    : E_AsyncEnvironmentState;
    bBusy                     : BOOL;
    bSuccess                  : BOOL;
    ipResult                  : I_AsyncServiceResultClass;
    sLastErrorResult          : T_MaxString;
    fbAbortReason             : FB_ServiceErrorClass := (stError:=SUCCESS_EVENT);
    sLastAbortReason          : T_MaxString;
    nInvokeID                 : UDINT;
    eServiceError             : E_AcsiServiceError;
    nServiceError             : UDINT;
    nCmdError                 : UDINT;
    sObjReference             : T_AcsiObjectReference;
    sCtrlReference            : T_AcsiObjectReference;

    bGetServerDirectory       : BOOL := TRUE;
    bGetLogicalDeviceDirectory: BOOL := TRUE;
    bGetLogicalNodeDirectory  : BOOL := TRUE;
    bGetAllServerValues       : BOOL := TRUE;
    bGetAllServerValues       : BOOL := TRUE;
    bGetAllDataValues_LLN0_ST : BOOL := TRUE;
    bGetDataValues_LLN0_ST_Beh: BOOL := TRUE;
END_VAR
```

```
fbConnection.Execute();
eState:= fbConnection.eState;

CASE state OF
    0:
        IF _bAbort THEN
            _bAbort:= FALSE;
            bSuccess:= fbConnection.AbortReq(ipReason:=fbAbortReason, ipSink:=0, ipResult=>ipResult)
;
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Idle AND (_bConnect OR _bReconnect) THEN
            _bConnect:= FALSE;
            bGetAllServerValues:= SEL(_bReadAllData, bGetAllServerValues, TRUE);
            bSuccess:= fbConnection.AssociateReq(ipSink:=0, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Established AND _bDisconnect THEN
            _bDisconnect:= FALSE;
            bSuccess:= fbConnection.ReleaseReq(ipSink:=0, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Established THEN
            state:= 10;
        END_IF
        _bConnect:= FALSE;
        _bDisconnect:= FALSE;
    1:
        IF ipResult <> 0 THEN
            ipResult.Execute();
```

```
            IF NOT (bBusy:=ipResult.IsBusy()) THEN
                state:= SEL(ipResult.IsCompleted(), 100, 0);
            END_IF
        END_IF
    10:
        IF bGetServerDirectory THEN
            bGetServerDirectory:= FALSE;
            bSuccess:= fbConnection.GetServerDirectoryReq(ipServer:=fb[IEDName], eClass:=E_AcsiServe
rDirectoryClass.LogicalDevice, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetLogicalDeviceDirectory THEN
            bGetLogicalDeviceDirectory:= FALSE;
            bSuccess:= fbConnection.GetLogicalDeviceDirectoryReq(ipLogicalDevice:=fb[IEDName].IEDLD1
, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetLogicalNodeDirectory THEN
            bGetLogicalNodeDirectory:= FALSE;
            bSuccess:= fbConnection.GetLogicalNodeDirectoryReq(ipLogicalNode:=fb[IEDName].IEDLD1.LLN
0, eClass:=E_AcsiLogicalNodeClass.DataSet, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipRe
sult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetAllServerValues THEN
            bGetAllServerValues:= FALSE;
            bSuccess:= fbConnection.GetAllServerValuesReq(ipServer:=fb[IEDName], hUser:=0, ipSink:=0
, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetAllDataValues_LLN0_ST THEN
            bGetAllDataValues_LLN0_ST:= FALSE;
            bSuccess:= fbConnection.GetAllDataValuesReq(ipLogicalNode:=fb[IEDName].IEDLD1.LLN0, eFc:
=E_AcsiFc.ST_, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetDataValues_LLN0_ST_Beh THEN
            bGetDataValues_LLN0_ST_Beh:= FALSE;
            bSuccess:= fbConnection.GetDataValuesReq(ipData:=fb[IEDName].IEDLD1.LLN0.Beh, eFc:=E_Acs
iFc.ST_, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSE
            state:= 0;
        END_IF
    11:
        IF ipResult <> 0 THEN
            ipResult.Execute();
            IF NOT (bBusy:=ipResult.IsBusy()) THEN
                state:= SEL(ipResult.IsCompleted(), 100, 0);
            END_IF
        END_IF
    100:
        state:= 0;
        IF ipResult <> 0 THEN
            nCmdError:= nCmdError + 1;
            sLastErrorResult:= ipResult.Dump();
        END_IF
END_CASE
```

5.10.1.1.5 FB_[IEDName]Gse

This function block connects a network adapter of the TwinCAT control computer with the GOOSE control blocks in the data model and controls the execution of these control blocks.

To be able to use GOOSE communication, the real-time network adapter must be commissioned and linked. This is explained in RT Ethernet adapter configuration [▶ 35].

Full description [▶ 27]

Syntax

Definition:

```
FUNCTION_BLOCK FB_[IEDName]Gse IMPLEMENTS I_GseLinkStatusEventSink
VAR_INPUT
    fbAdapter    : FB_GseAdapterClass := (ipLinkStatus:=THIS^);
END_VAR
VAR
    eLinkStatus  : E_GseLinkStatus;
    bSuccess     : BOOL;
    ipError      : I_ServiceErrorClass;
```

```
    bSubscribe   : BOOL := TRUE;
    bUnsubscribe : BOOL;
END_VAR
```

```
bSuccess:= fbAdapter.Execute(ipError=>ipError);
IF bSubscribe THEN
    bSubscribe:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Subscriber.Enable(ipAdapter:=fbAdapter, ipError=>ipErr
or);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Subscriber.Enable(ipAdapter:=fbAdapter, ipError=>ipErr
or);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Subscriber.Enable(ipAdapter:=fbAdapter, ipError=>ipErr
or);
ELSIF bUnsubscribe THEN
    bUnsubscribe:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Subscriber.Disable(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Subscriber.Disable(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Subscriber.Disable(ipError=>ipError);
ELSE
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Subscriber.Execute(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Subscriber.Execute(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Subscriber.Execute(ipError=>ipError);
END_IF
```

5.10.1.1.6 MAIN

The TwinCAT PLC project contains a "MAIN" program as standard. This is called cyclically by a TwinCAT task and in turn calls the program "P_IEC61850MAIN".

Full description [▶ 27]

Syntax

Definition:

```
PROGRAM MAIN
VAR
END_VAR
```

```
P_IEC61850MAIN();
```

5.10.1.1.7 P_IEC61850MAIN

The program "P_IEC61850MAIN" encapsulates the call of the Client block and of the optional GSE block, separates the IEC 61850 communication from the remainder of the PLC machine program and helps, for example, with the implementation of further Clients.

Full description [▶ 27]

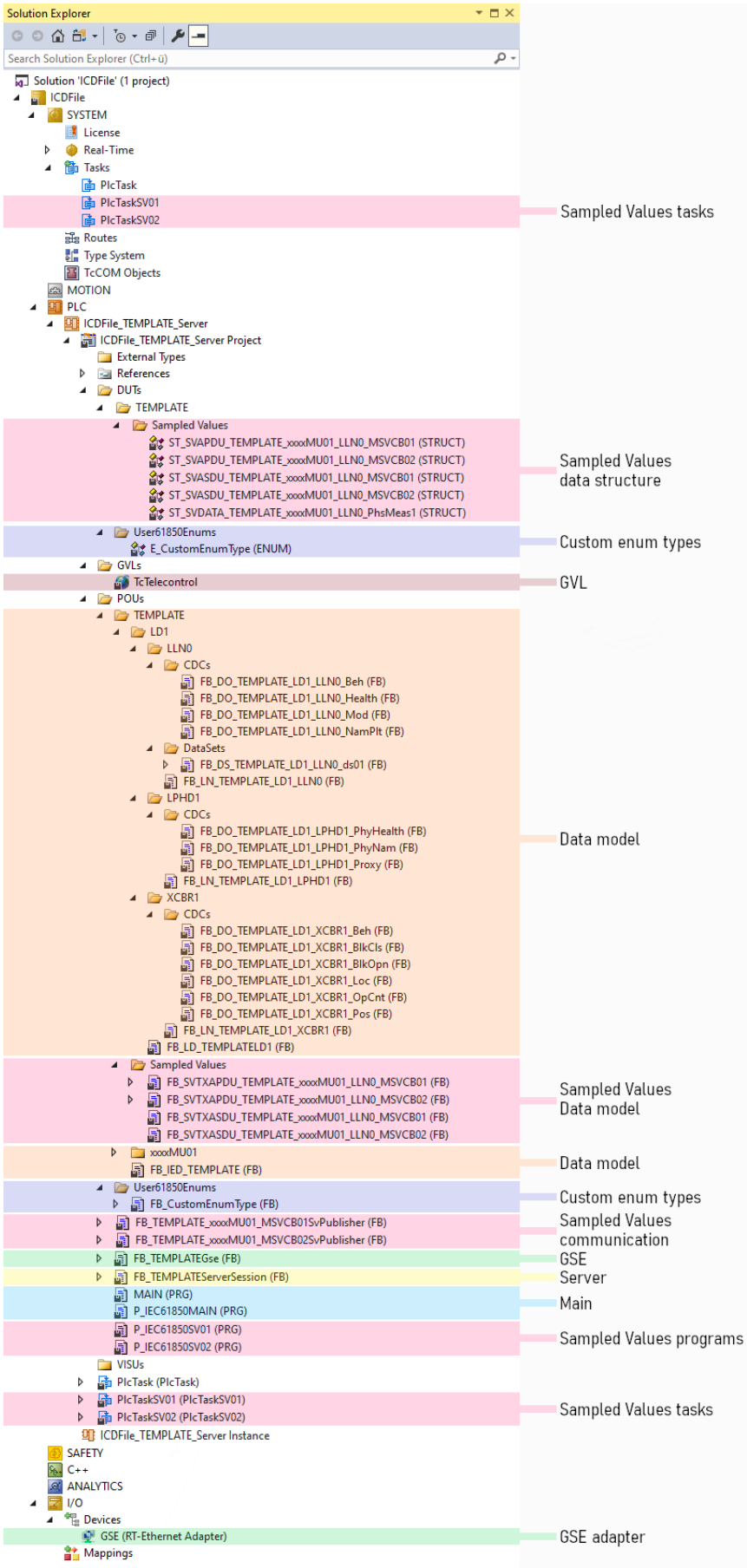
Syntax

Definition:

```
PROGRAM P_IEC61850MAIN
VAR
END_VAR
```

```
fb[IEDName]Client();
fb[IEDName]Gse();
```

5.10.1.2 Server



Custom enum types:

User-defined enum types that do not exist in the standard system.

GVL:

In the Global Variable List, at least one instance of the server session function block, the IED data model function block instance, and optionally one or more Sampled Values and GOOSE communication function block instances are instantiated and initialized.

In addition, the code version used during code generation and the version of the TwinCAT Telecontrol Configurator used are also stored there.

Datamodel:

The entire hierarchical structure of the IED data model as function blocks. Also includes the IED block that is instantiated in the Global Variable List.

GSE / GSE adapter:

This function block connects a network adapter of the TwinCAT control computer with the GOOSE control blocks in the data model and controls the execution of these control blocks.

To be able to use GOOSE communication, the real-time network adapter must be commissioned and linked. This is explained in RT Ethernet adapter configuration [▶ 35].

The GSE adapter is a network adapter for GOOSE and Sampled Values communication whose network status is constantly queried by the corresponding function block.

Server:

This block contains a state machine, the basic states of which are used in every Server example.

The Server session block is responsible for establishing connections and exchanging data with a single Client. If there are multiple simultaneous Client connections, multiple instances of this block are required and instantiated.

Main:

The TwinCAT PLC project contains a "MAIN" program as standard. This is called cyclically by a TwinCAT task and in turn calls the program "P_IEC61850MAIN".

The program "P_IEC61850MAIN" encapsulates the call of the Server session block and the optional GSE block. It also separates the IEC 61850 communication from the rest of the PLC machine program.

Sampled Values

The control blocks in a Sampled Values configuration each require structures, function blocks, tasks, and programs.

Each program is assigned to a single task and calls all publisher function blocks at the same cycle time.

The Publisher function block establishes a connection between the network adapter of the TwinCAT control computer and the corresponding Sampled Values control block in the data model and controls its execution.

To be able to use GOOSE communication, the real-time network adapter must be commissioned and linked. This is explained in RT Ethernet adapter configuration [▶ 35].

5.10.1.2.1 E_[EnumName]

User-defined enum types that do not exist in the standard system.

The maximum indices are -32768 and 32767, because it is a 16-bit integer. The naming of the enum values from the ICD file is applied only as a comment.

Syntax

Example Definition:

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_[EnumName] :
(
    e_1 := 1, (*Ok*)
    e_2 := 2, (*Warning*)
    e_3 := 3, (*Alarm*)
    ...

) SINT;
END_TYPE
```

Values

Name	Description
e_1	User-defined enum value, corresponds to "OK" in the ICD file.
e_2	User-defined enum value, corresponds to "Warning" in the ICD file.
e_3	User-defined enum value, corresponds to "Alarm" in the ICD file.
…	Any other user-defined enum values in the index range between -32768 and 32767.

5.10.1.2.2 TcTelecontrol (GVL)

In the Global Variable List, at least one instance of the server session function block, the IED data model function block instance, and optionally one or more Sampled Values and GOOSE communication function block instances are instantiated and initialized.

In addition, the code version used during code generation and the version of the TwinCAT Telecontrol Configurator used are also stored there.

Full description [▶ 31]

Syntax

Definition:

```
VAR_GLOBAL
    ipCreator                 : I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=2
).SetGuiVer(major:=1, minor:=1, build:=94, revision:=1);
    fb[IEDName]               : FB_IED_[IEDName];
    fb[IEDName]Server         : FB_iec61850ServerClass := (ipIED:=fb[IEDName], settings:=(bEnable:=T
RUE, sLocalHost:='127.0.0.1'));
    fb[IEDName]ServerSession1 : FB_[IEDName]ServerSession := (fbConnection:=(ipServer:=fb[IEDName]Se
rver, settings:=(bEnable:=TRUE)));
    fb[IEDName]Gse            : FB_[IEDName]Gse := (fbAdapter:=(ipIED:=fb[IEDName], settings:=(sMult
icastAddr:='01-0C-CD-01-00-00', eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
END_VAR
```

5.10.1.2.3 FB_[EnumName]

User-defined enum types that do not exist in the standard system.

Full description [▶ 103]

Syntax

Definition:

```
{attribute 'no_explicit_call' := 'Direct call not allowed for type FB_[EnumName]'}
{attribute 'call_after_init'}
FUNCTION_BLOCK FB_[EnumName] EXTENDS FB_AcsiBATypeENUMERATED
VAR
    _eValue: E_CustomEnumType;
    _eConfig: E_CustomEnumType;
    {attribute 'hide'}
```

```
    ipInit: I_AcsiCommonAttributeClass := THIS^.AddDAValue(eP:=E_AcsiP.Actual, in:=_eValue).AddDAVal
ue(eP:=E_AcsiP.Config, in:=_eConfig).AddDAType(in:=THIS^.T(in:=E_AcsiBasicType.Enum));
END_VAR
```

5.10.1.2.4 FB_[IEDName]Gse

This function block connects a network adapter of the TwinCAT control computer with the GOOSE control blocks in the data model and controls the execution of these control blocks.

To be able to use GOOSE communication, the real-time network adapter must be commissioned and linked. This is explained in RT Ethernet adapter configuration [▶ 35].

Full description [▶ 31]

Syntax

Definition:

```
FUNCTION_BLOCK FB_[IEDName]Gse IMPLEMENTS I_GseLinkStatusEventSink
VAR_INPUT
    fbAdapter   : FB_GseAdapterClass := (ipLinkStatus:=THIS^);
END_VAR
VAR
    eLinkStatus : E_GseLinkStatus;
    bSuccess    : BOOL;
    ipError     : I_ServiceErrorClass;
    bStart      : BOOL := TRUE;
    bStop       : BOOL;
    bUpdate     : BOOL;
END_VAR
```

```
bSuccess:= fbAdapter.Execute(ipError=>ipError);

IF bStart THEN
    bStart:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Publisher.Start(ipAdapter:=fbAdapter, ipError=>ipError
);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Publisher.Start(ipAdapter:=fbAdapter, ipError=>ipError
);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Publisher.Start(ipAdapter:=fbAdapter, ipError=>ipError
);
ELSIF bStop THEN
    bStop:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Publisher.Stop(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Publisher.Stop(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Publisher.Stop(ipError=>ipError);
ELSIF bUpdate THEN
    bUpdate:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Publisher.Update(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Publisher.Update(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Publisher.Update(ipError=>ipError);
ELSE
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Publisher.Execute(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Publisher.Execute(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Publisher.Execute(ipError=>ipError);
END_IF
```

The GSE block implements the "I_GseLinkStatusEventSink" interface. The method: "OnLinkStatusChange" belongs to this interface implementation and is called whenever the status of the network connection (at the network adapter) changes. The PLC application can, for example, query or check the network connection status via "eLinkStatus" variable.

```
METHOD OnLinkStatusChange
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    eStatus   : E_GseLinkStatus;
END_VAR
VAR
END_VAR
```

```
eLinkStatus:= eStatus;
```

5.10.1.2.5 FB_[IEDName]ServerSession

This block contains a state machine, the basic states of which are used in every Server example.

The Server session block is responsible for establishing connections and exchanging data with a single Client. If there are multiple simultaneous Client connections, multiple instances of this block are required and instantiated.

Full description [▶ 31]

Syntax

Definition:

```
FUNCTION_BLOCK FB_[IEDName]ServerSession
VAR_INPUT
    fbConnection     : FB_iec61850ConnectionClass := (ipAbortInd:=THIS^, ipAssociateInd:=THIS^, ipRe
leaseInd:=THIS^);
END_VAR
VAR
    _bAbort          : BOOL;
    _bDisconnect     : BOOL;
    state            : BYTE;
    eState           : E_AsyncEnvironmentState;
    bBusy            : BOOL;
    bSuccess         : BOOL;
    ipResult         : I_AsyncServiceResultClass;
    sLastErrorResult : T_MaxString;
    fbAbortReason    : FB_ServiceErrorClass := (stError:=SUCCESS_EVENT);
    sLastAbortReason : T_MaxString;
END_VAR
```

```
fbConnection.Execute();
eState:= fbConnection.eState;

CASE state OF
    0:
        IF _bAbort THEN
            _bAbort:= FALSE;
            bSuccess:= fbConnection.AbortReq(ipReason:=fbAbortReason, ipSink:=0, ipResult=>ipResult)
;
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Established AND _bDisconnect THEN
            _bDisconnect:= FALSE;
            bSuccess:= fbConnection.ReleaseReq(ipSink:=0, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Established THEN
            state:= 10;
        END_IF
        _bDisconnect:= FALSE;
    1:
        IF ipResult <> 0 THEN
            ipResult.Execute();
            IF NOT (bBusy:=ipResult.IsBusy()) THEN
                state:= SEL(ipResult.IsCompleted(), 100, 0);
            END_IF
        END_IF
    10:
        state:= 0;
    100:
        state:= 0;
        IF ipResult <> 0 THEN
            sLastErrorResult:= ipResult.Dump();
        END_IF
END_CASE
```

5.10.1.2.6 MAIN

The TwinCAT PLC project contains a "MAIN" program as standard. This is called cyclically by a TwinCAT task and in turn calls the program "P_IEC61850MAIN".

Full description [▶ 31]

Syntax

Definition:

```
PROGRAM MAIN
VAR
END_VAR
```

```
P_IEC61850MAIN();
```

5.10.1.2.7 P_IEC61850MAIN

The program "P_IEC61850MAIN" encapsulates the call of the Server session block and the optional GSE block. It also separates the IEC 61850 communication from the rest of the PLC machine program.

Full description [▶ 31]

Syntax

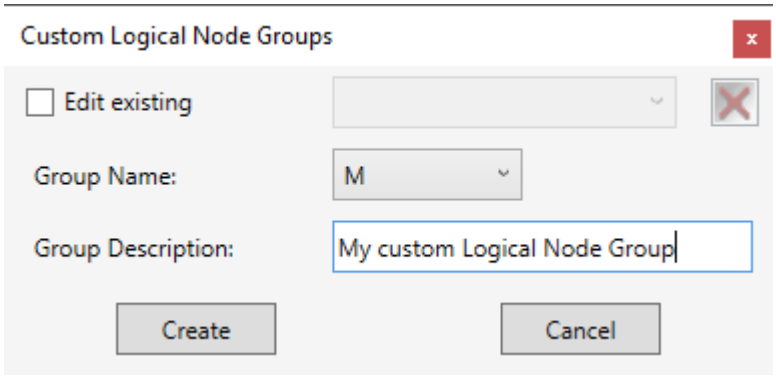
Definition:

```
PROGRAM P_IEC61850MAIN
VAR
END_VAR
```

```
fb[IEDName]Server.Execute();
fb[IEDName]ServerSession1();
fb[IEDName]Gse();
```

5.11 Custom Logical Node Groups

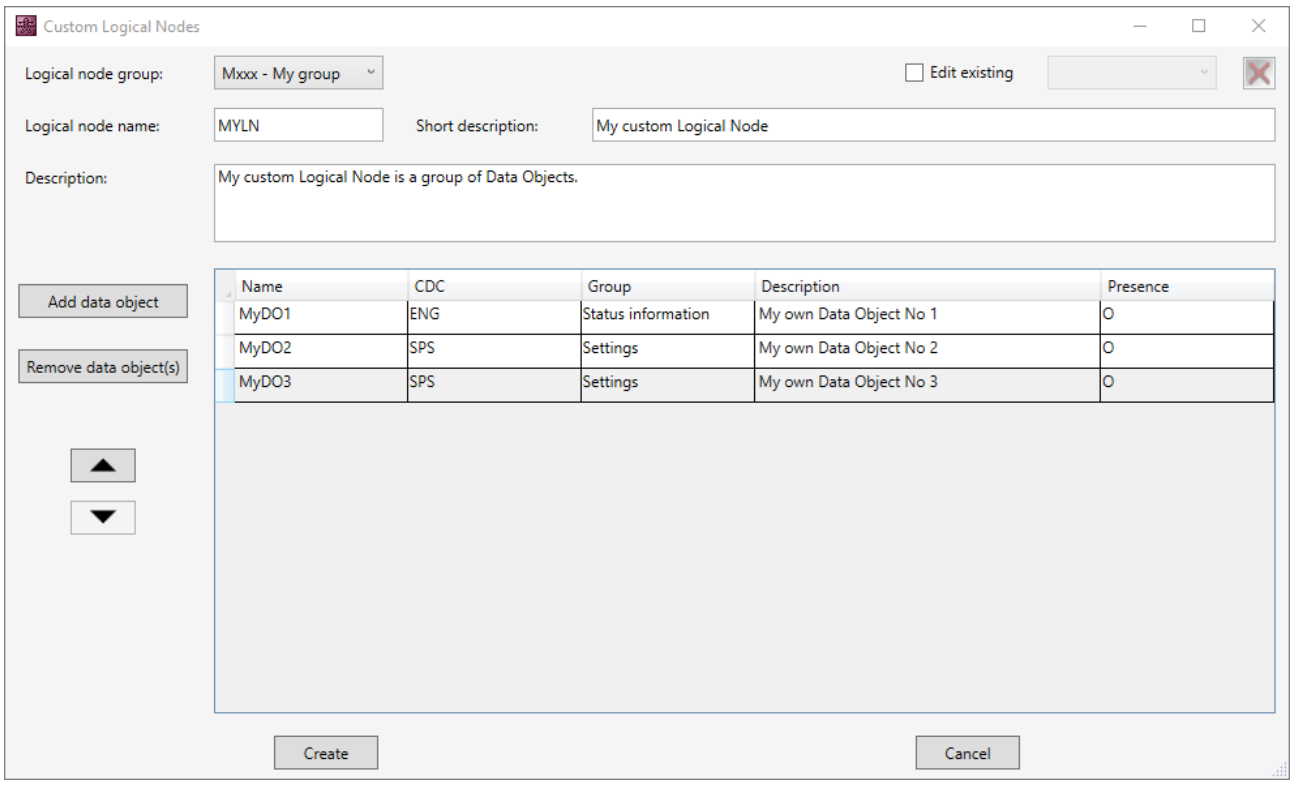
With the function *Custom Logical Node Groups* it is possible to define own Logical Node Groups. The new elements are written to a private database. This way, they are also available again for each new configuration.



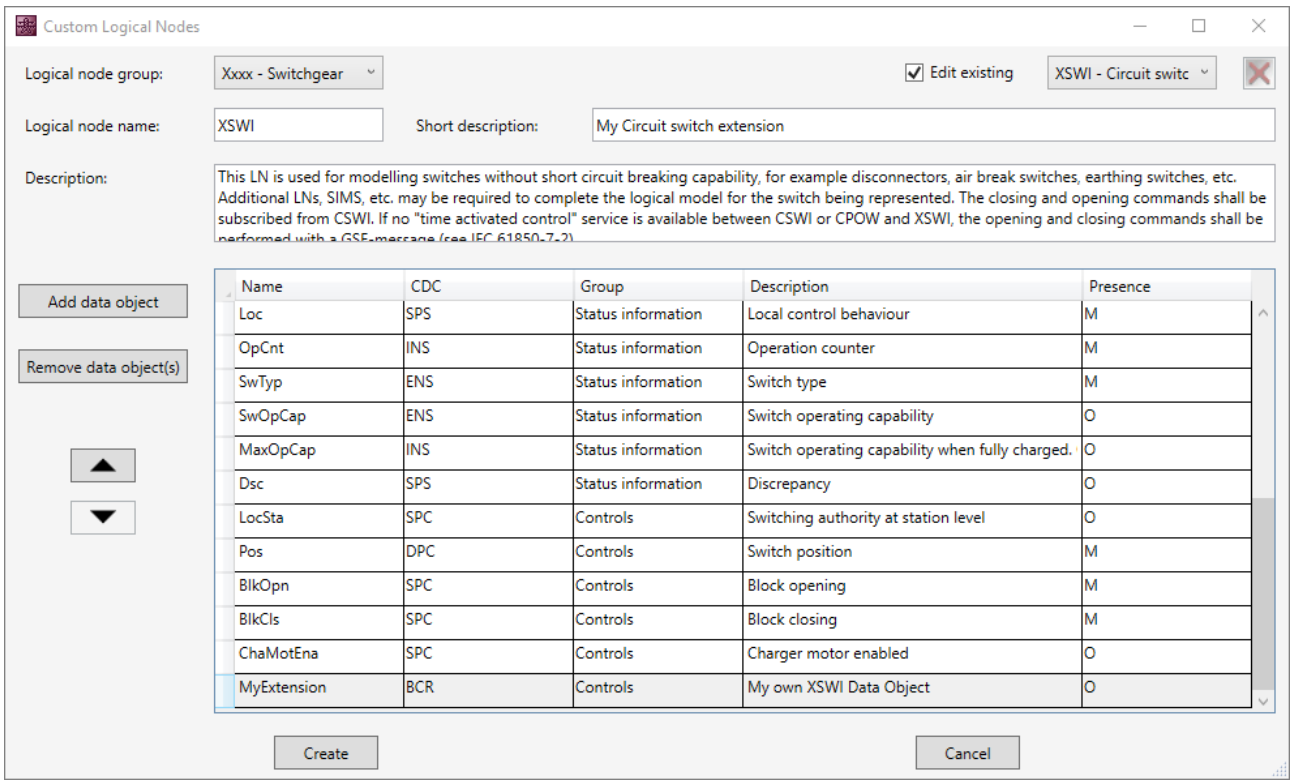
In the shown window **Name** and **Description** of the Logical Node Group can be adjusted. By confirming with the button **Create** the new Logical Node Group is written into the private database and is available for configuration from now on. At the top of the window, you can also select an already created Logical Node Group to edit or delete it.

5.12 Custom Logical Nodes

With the function *Custom Logical Nodes* it is possible to define own Logical Nodes or to extend existing Logical Nodes. The new elements are written to a private database. This way, they are also available again for each new configuration.



In the window shown, you can define your own Logical Nodes. For this purpose, for each Data Object Name and a Common Data Class can be selected. Furthermore, the data object can be assigned to a group, so that a categorization in the configurator leads to an increase in clarity. In the Presence column you can set whether the element should be added by default or whether it should be optional. By using the arrows on the left side of the window, the order of the Data Objects can be adjusted. In addition, new Data Objects can be added or existing ones deleted.

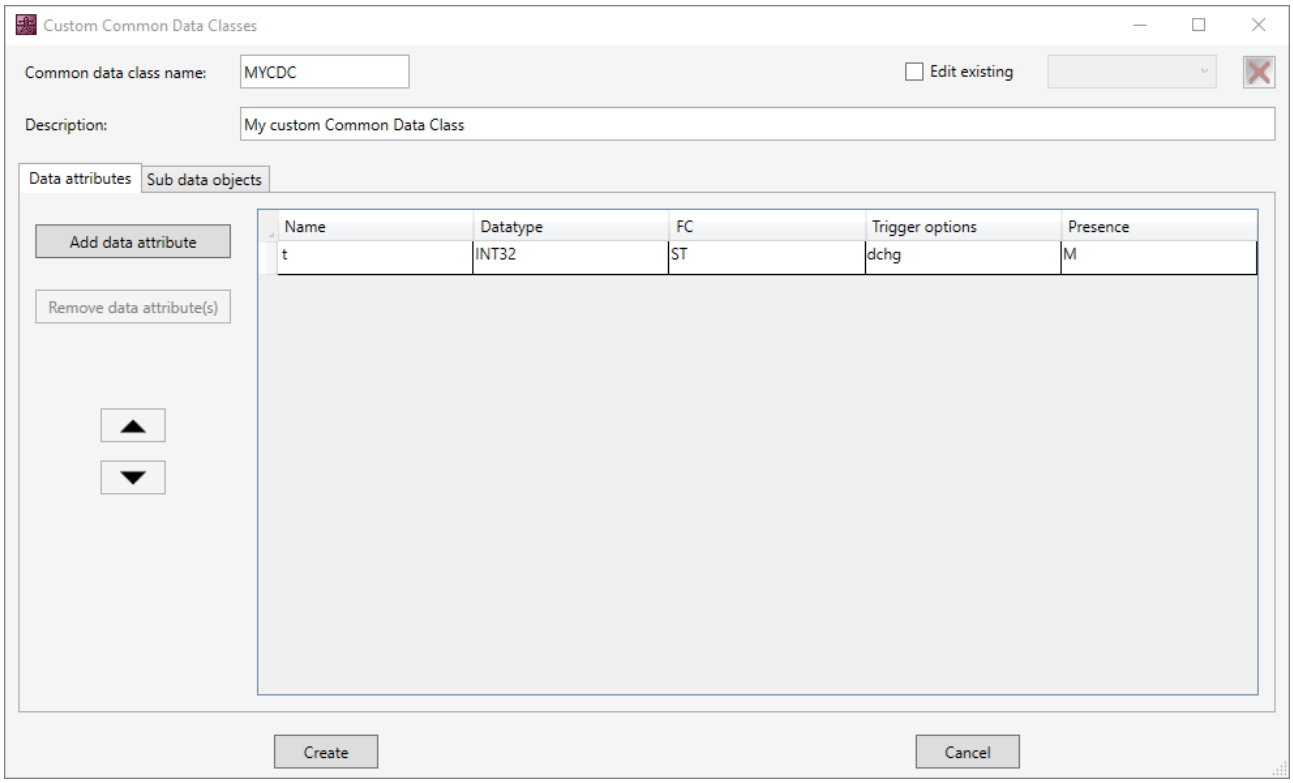


In the figure shown here, the logical node defined in the standard **XSWI (Circuit Switch)** is extended in a user-defined way. For this purpose, a **Logical Node** can be selected for editing at the top right of the window. One custom Logical Node with the same name can be created per Logical Node from the default. Additional custom Logical Nodes must be given individual names.

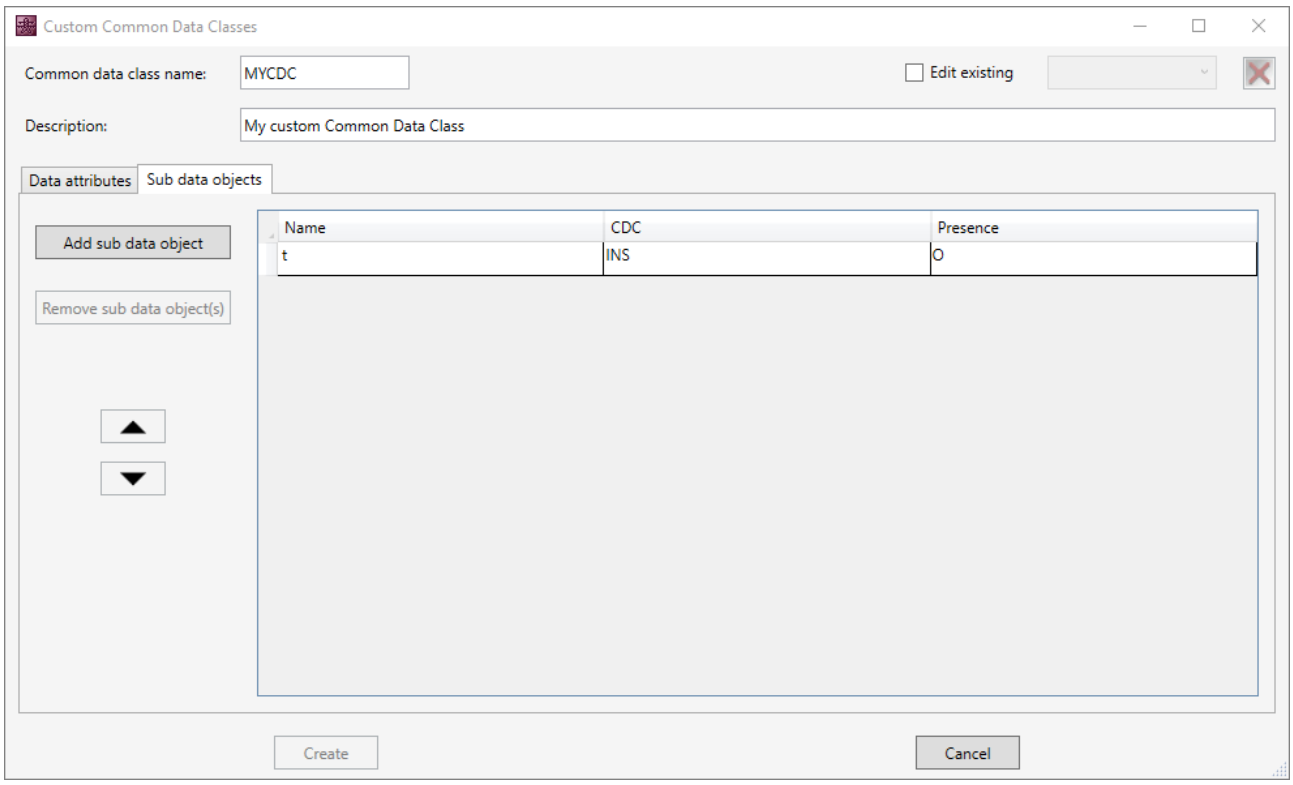
By confirming with the button **Create** the new Logical Node is written into the private database and is available for configuration from now on.

5.13 Custom Common Data Classes

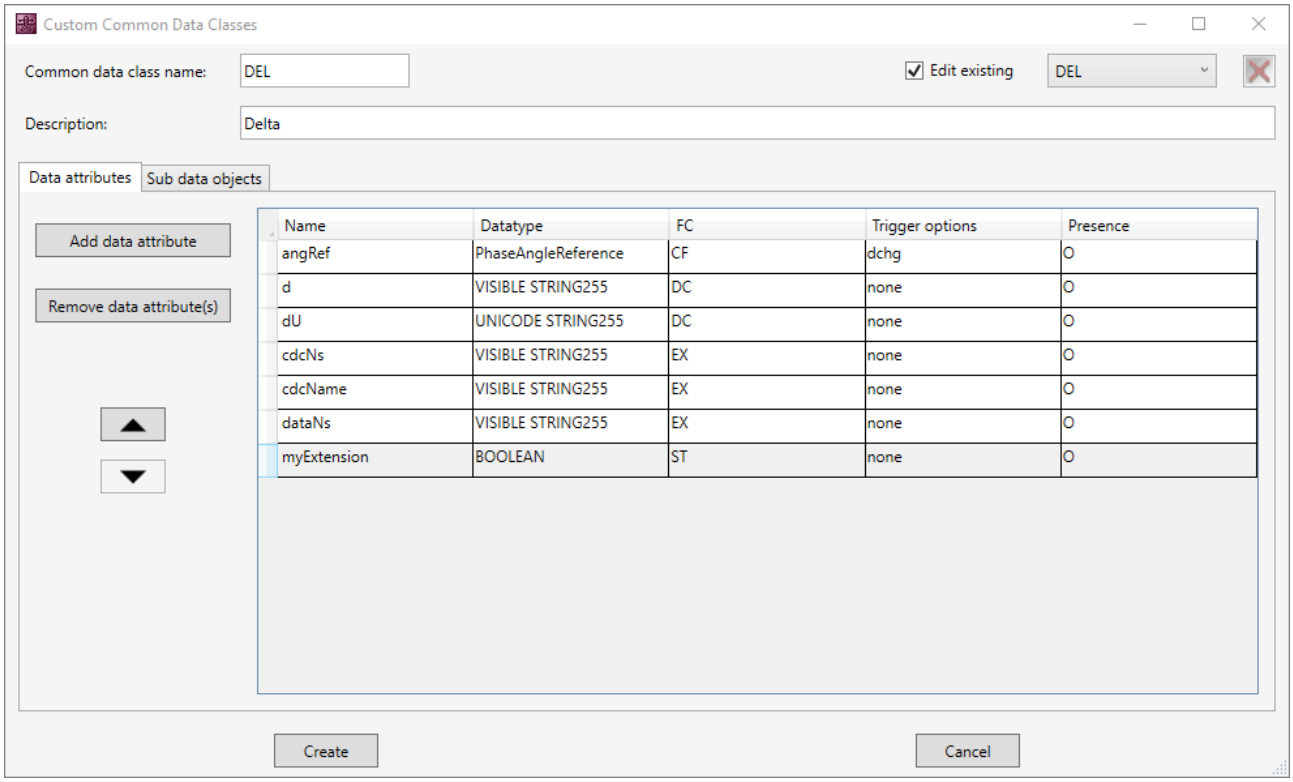
With the function *Custom Common Data Classes* it is possible to define own Common Data Classes or to extend existing Common Data Classes. The new elements are written to a private database. This way, they are also available again for each new configuration.



In the window shown, you can define your own Common Data Classes. For this purpose, a name, data type, a functional constraint and the trigger options can be set for each Data Attribute. In the Presence column you can set whether the element should be added by default or whether it should be optional. Using the arrows on the left side of the window, the order of the Data Attributes can be adjusted. In addition, new Data Attributes can be added or existing ones deleted.



In addition, the Sub Data Objects of the Common Data Class can be configured. Name and Common Data Class can be selected for the individual Sub Data Objects. In the Presence column you can set whether the element should be added by default or whether it should be optional. By using the arrows on the left side of the window, the order of the Sub Data Objects can be adjusted. In addition, new Sub Data Objects can be added or existing ones deleted.



In the figure shown here, the Common Data Class **DEL (Delta)** defined in the standard is extended in a user-defined way. For this purpose, a **Common Data Class** can be selected for editing at the top right of the window. One user-defined Common Data Class with the same name can be created per Common Data Class from the standard. Additional user-defined Common Data Classes must be given individual names.

By confirming with the button **Create** the new Common Data Class is written into the private database and is available for configuration from now on.

6 PLC API

6.1 Libraries

TwinCAT IEC 61131 PLC libraries.

- Tc3_Acsi (Tc3_Acsi.compiled-library)
- Tc3_Ber (Tc3_Ber.compiled-library)
- Tc3_Collections (Tc3_Collections.compiled-library)
- Tc3_Gse (Tc3_Gse.compiled-library)
- Tc3_iec61850 (Tc3_iec61850.compiled-library)
- Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)
- Tc3_Mms (Tc3_Mms.compiled-library)
- Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)
- Tc3_Sockets (Tc3_Sockets.compiled-library)
- Tc3_Tpkt (Tc3_Tpkt.compiled-library)
- Tc3_Ulosi (Tc3_Ulosi.compiled-library)

6.2 Namespaces

6.2.1 TwinCAT TypeSystem namespace

Type: TwinCAT system resource
Library: Not required

Global Variable Lists

- TC_EVENTS [▶ 570]
- TC_EVENT_CLASSES [▶ 656]

6.2.2 Tc3_Acsi namespace

Type: IEC 61131 PLC library
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Global Variable Lists

- GVL_Acsi [▶ 659]

Parameter lists

- Param_Acsi [▶ 563]

6.2.3 Tc3_Ber namespace

Type: IEC 61131 PLC library
Library: Tc3_Ber (Tc3_Ber.compiled-library)

6.2.4 Tc3_Collections namespace

Type: IEC 61131 PLC library
Library: Tc3_Collections (Tc3_Collections.compiled-library)

6.2.5 Tc3_Gse namespace

Type: IEC 61131 PLC library
Library: Tc3_Gse (Tc3_Gse.compiled-library)

6.2.6 Tc3_iec61850_8_1 namespace

Type: IEC 61131 PLC library
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

6.2.7 Tc3_iec61850 namespace

Type: IEC 61131 PLC library
Library: Tc3_iec61850 (Tc3_iec61850.compiled-library)

6.2.8 Tc3_Mms namespace

Type: IEC 61131 PLC library
Library: Tc3_Mms (Tc3_Mms.compiled-library)

6.2.9 Tc3_Rfc1006 namespace

Type: IEC 61131 PLC library
Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

6.2.10 Tc3_Sockets namespace

Type: IEC 61131 PLC library
Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

6.2.11 Tc3_Tpkt namespace

Type: IEC 61131 PLC library
Library: Tc3_Tpkt (Tc3_Tpkt.compiled-library)

6.2.12 Tc3_Ulosi namespace

Type: IEC 61131 PLC library
Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

6.2.13 Tc2_TcpIp namespace

Type: IEC 61131 PLC library
Library: Tc2_TcpIp (Tc2_TcpIp.compiled-library)

6.2.14 Tc2_System namespace

Type: IEC 61131 PLC library
Library: Tc2_System (Tc2_System.compiled-library)

6.3 Function blocks

6.3.1 Data model

6.3.1.1 Data attributes

6.3.1.1.1 FB_AcsiBATypeBOOLEAN

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeBOOLEAN

```
FUNCTION_BLOCK FB_AcsiBATypeBOOLEAN EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
bValue	BOOL	Get,Set	Attribute value.

6.3.1.1.2 FB_AcsiBATypeCODEDENUM

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeCODEDENUM

```
FUNCTION_BLOCK FB_AcsiBATypeCODEDENUM EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
bFixLen	BOOL	Get,Set	Attribute data length (FALSE:= variable length (default value), TRUE:= fixed length).
sAny	T_MaxString	Get, Set	Attribute value as formatted binary string.

6.3.1.1.3 FB_AcsiBATypeENUMERATED

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED

```
FUNCTION_BLOCK FB_AcsiBATypeENUMERATED EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
iAny	SINT	Get,Set	Attribute value as 8-bit signed integer.
i16Any	INT	Get, Set	Attribute value as 16-bit signed integer.

6.3.1.1.4 FB_AcsiBATypeFLOAT32

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeFLOAT32

```
FUNCTION_BLOCK FB_AcsiBATypeFLOAT32 EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
fValue	REAL	Get,Set	Attribute value, 32-bit floating point number with simple accuracy according to IEEE 754.

6.3.1.1.5 FB_AcsiBATypeFLOAT64

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeFLOAT64

```
FUNCTION_BLOCK FB_AcsiBATypeFLOAT64 EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
fValue	LREAL	Get,Set	Attribute value, 64-bit floating point number with double accuracy according to IEEE 754.

6.3.1.1.6 FB_AcsiBATypeINT8

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeINT8

```
FUNCTION_BLOCK FB_AcsiBATypeINT8 EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
iValue	SINT	Get,Set	Attribute value.

6.3.1.1.7 FB_AcsiBATypeINT8U

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeINT8U

```
FUNCTION_BLOCK FB_AcsiBATypeINT8U EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
nValue	BYTE	Get,Set	Attribute value.

6.3.1.1.8 FB_AcsiBATypeINT16

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeINT16

```
FUNCTION_BLOCK FB_AcsiBATypeINT16 EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
iValue	INT	Get,Set	Attribute value.

6.3.1.1.9 FB_AcsiBATypeINT16U

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeINT16U

```
FUNCTION_BLOCK FB_AcsiBATypeINT16U EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
nValue	WORD	Get,Set	Attribute value.

6.3.1.1.10 FB_AcsiBATypeINT24

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeINT24

```
FUNCTION_BLOCK FB_AcsiBATypeINT24 EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
iValue	T_INT24 [▶ 557]	Get,Set	Attribute value.

6.3.1.1.11 FB_AcsiBATypeINT24U

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeINT24U

```
FUNCTION_BLOCK FB_AcsiBATypeINT24U EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
nValue	T_UINT24 [▶ 559]	Get,Set	Attribute value.

6.3.1.1.12 FB_AcsiBATypeINT32

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeINT32

```
FUNCTION_BLOCK FB_AcsiBATypeINT32 EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
iValue	DINT	Get,Set	Attribute value.

6.3.1.1.13 FB_AcsiBATypeINT32U

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeINT32U

```
FUNCTION_BLOCK FB_AcsiBATypeINT32U EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
nValue	DWORD	Get,Set	Attribute value.

6.3.1.1.14 FB_AcsiBATypeINT64

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeINT64

```
FUNCTION_BLOCK FB_AcsiBATypeINT64 EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
iValue	LINT	Get,Set	Attribute value.

6.3.1.1.15 FB_AcsiBATypeINT64U

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeINT64U

```
FUNCTION_BLOCK FB_AcsiBATypeINT64U EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
nValue	LWORD	Get,Set	Attribute value.

6.3.1.1.16 FB_AcsiBATypeINT128

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeINT128

```
FUNCTION_BLOCK FB_AcsiBATypeINT128 EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
iValue	T_INT28 [▶ 557]	Get,Set	Attribute value.

6.3.1.1.17 FB_AcsiBATypeINT128U

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeINT128U

```
FUNCTION_BLOCK FB_AcsiBATypeINT128U EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
nValue	T_UINT128 [▶ 559]	Get,Set	Attribute value.

6.3.1.1.18 FB_AcsiBATypeOCTETSTRING

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeOCTETSTRING

```
FUNCTION_BLOCK FB_AcsiBATypeOCTETSTRING EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
bFixLen	BOOL	Get,Set	Attribute data length (FALSE:= variable length (default value), TRUE:= fixed length).
sAny	T_MaxString	Get, Set	Attribute value as formatted hexadecimal string. Samples: "AA BB CC DD 01 02" or "00 12 34".

6.3.1.1.19 FB_AcsiBATypeUNICODESTRING

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeUNICODESTRING

```
FUNCTION_BLOCK FB_AcsiBATypeUNICODESTRING EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
bFixLen	BOOL	Get,Set	Attribute data length (FALSE:= variable length (default value), TRUE:= fixed length).
sAny	WSTRING(255)	Get, Set	Attribute value.

6.3.1.1.20 FB_AcsiBATypeVISIBLESTRING

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeVISIBLESTRING

```
FUNCTION_BLOCK FB_AcsiBATypeVISIBLESTRING EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
bFixLen	BOOL	Get,Set	Attribute data length (FALSE:= variable length (default value), TRUE:= fixed length).
sAny	T_MaxString	Get, Set	Attribute value.

6.3.1.1.21 FB_AcsiDATypeAddr

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeOCTETSTRING [▶ 108] -> FB_AcsiDATypeAddr

```
FUNCTION_BLOCK FB_AcsiDATypeAddr EXTENDS FB_AcsiBATypeOCTETSTRING
```

Properties

Name	Type	Access	Description
oValue	T_OCTET6 [▶ 558]	Get,Set	Attribute value.

6.3.1.1.22 FB_AcsiDATypeAdjSt

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeAdjSt

```
FUNCTION_BLOCK FB_AcsiDATypeAdjSt EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiAdjSt [▶ 475]	Get,Set	Attribute value.

6.3.1.1.23 FB_AcsiDATypeAlmLev

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeAlmLev

```
FUNCTION_BLOCK FB_AcsiDATypeAlmLev EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiAlmLev [▶ 475]	Get,Set	Attribute value.

6.3.1.1.24 FB_AcsiDATypeAnalogueValue

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeAnalogueValue

```
FUNCTION_BLOCK FB_AcsiDATypeAnalogueValue EXTENDS FB_AcsiCommonAttributeClass
```

Inputs

Name	Type	Description
i	FB_AcsiBATypeINT32 [▶ 106]	Attribute for displaying the analog value as an integer.
f	FB_AcsiBATypeFLOAT32 [▶ 104]	Attribute for displaying the analog value as a floating point number.

Methods

Name	Description
GetFloatPVal [▶ 110]	Reads the floating point process value.
GetIntPVal [▶ 111]	Reads the integer process value.
SetFloatPVal [▶ 111]	Writes the floating point process value.
SetIntPVal [▶ 111]	Writes the integer process value.
SetPVal [▶ 111]	Writes the floating point and integer process value.

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility.
Fc [▶ 366]	DWORD	Get,Set	Functional group.
fValue	REAL	Get,Set	Value of the data attribute: f.
iValue	DINT	Get,Set	Value of the data attribute: i.

6.3.1.1.24.1 GetFloatPVal

Reads the attribute "f" of "AnalogValue" (floating-point representation of the analog value) and scales the attribute value into a TwinCAT process value of type: LREAL.

```
METHOD GetFloatPVal : BOOL
VAR_INPUT
    ipUnit   : I_AcsiDATypeUnit;
END_VAR
VAR_OUTPUT
    fProcess : LREAL;
END_VAR
```

6.3.1.1.24.2 GetIntPVal

Reads the attribute "i" of "AnalogValue" (integer representation of the analog value) and scales the attribute value into a TwinCAT process value of type: LREAL.

```
METHOD GetIntPVal : BOOL
VAR_INPUT
    ipSVC    : I_AcsiDATypeScaledValueConfig;
END_VAR
VAR_OUTPUT
    fProcess : LREAL;
END_VAR
```

6.3.1.1.24.3 SetFloatPVal

Scales a TwinCAT process value of type: LREAL and writes the attribute value to the attribute "f" of "AnalogValue" (floating-point representation of the analog value).

```
METHOD SetFloatPVal : BOOL
VAR_INPUT
    fProcess : LREAL;
    ipUnit   : I_AcsiDATypeUnit;
END_VAR
```

6.3.1.1.24.4 SetIntPVal

Scales a TwinCAT process value of type: LREAL and writes the attribute value to the attribute "i" of "AnalogValue" (integer representation of the analog value).

```
METHOD SetIntPVal : BOOL
VAR_INPUT
    fProcess : LREAL;
    ipSVC    : I_AcsiDATypeScaledValueConfig;
END_VAR
```

6.3.1.1.24.5 SetPVal

Scales a TwinCAT process value of type: LREAL and writes the attribute value to the attribute of "i" or "f" of "AnalogValue" (analog value as integer or floating point number).

```
METHOD SetPVal : BOOL
VAR_INPUT
    fProcess : LREAL;
    ipUnit   : I_AcsiDATypeUnit;
    ipSVC    : I_AcsiDATypeScaledValueConfig;
END_VAR
```

6.3.1.1.25 FB_AcsiDATypeAngRef

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeAngRef

```
FUNCTION_BLOCK FB_AcsiDATypeAngRef EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiAngRef [▶ 476]	Get,Set	Attribute value.

6.3.1.1.26 FB_AcsiDATypeApplAddCause

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeApplAddCause

```
FUNCTION_BLOCK FB_AcsiDATypeApplAddCause EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiApplAddCause [▶ 476]	Get,Set	Attribute value.

6.3.1.1.27 FB_AcsiDATypeApplError

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeApplError

```
FUNCTION_BLOCK FB_AcsiDATypeApplError EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiApplError [▶ 478]	Get,Set	Attribute value.

6.3.1.1.28 FB_AcsiDATypeAutoRecSt

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeAutoRecSt

```
FUNCTION_BLOCK FB_AcsiDATypeAutoRecSt EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiAutoRecSt [▶ 478]	Get,Set	Attribute value.

6.3.1.1.29 FB_AcsiDATypeBeh

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeBeh

```
FUNCTION_BLOCK FB_AcsiDATypeBeh EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiBeh [▶ 479]	Get,Set	Attribute value.

6.3.1.1.30 FB_AcsiDATypeCalendarTime

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeCalendarTime

```
FUNCTION_BLOCK FB_AcsiDATypeCalendarTime EXTENDS FB_AcsiCommonAttributeClass
```

Inputs

Name	Type	Description
occ	FB_AcsiBATypeINT16U [▶ 105]	Data attribute.
occType	FB_AcsiDATypeOccType [▶ 126]	Data attribute.
occPer	FB_AcsiDATypeOccPer [▶ 126]	Data attribute.
weekDay	FB_AcsiDATypeWeekDay [▶ 151]	Data attribute.
month	FB_AcsiDATypeMonth [▶ 125]	Data attribute.
day	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
hr	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
mn	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility.
Fc [▶ 366]	DWORD	Get,Set	Functional group.
nOcc	WORD	Get,Set	Value of the data attribute.
eOccType	E_AcsiOccType [▶ 495]	Get,Set	Value of the data attribute.
eOccPer	E_AcsiOccPer [▶ 494]	Get,Set	Value of the data attribute.
eWeekDay	E_AcsiWeekDay [▶ 518]	Get,Set	Value of the data attribute.
eMonth	E_AcsiMonth [▶ 493]	Get,Set	Value of the data attribute.
nDay	BYTE	Get,Set	Value of the data attribute.
nHr	BYTE	Get,Set	Value of the data attribute.
nMn	BYTE	Get,Set	Value of the data attribute.

6.3.1.1.31 FB_AcsiDATypeCBOpCap

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeCBOpCap

```
FUNCTION_BLOCK FB_AcsiDATypeCBOpCap EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiCBOpCap [▶ 479]	Get,Set	Attribute value.

6.3.1.1.32 FB_AcsiDATypeCell

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeCell

```
FUNCTION_BLOCK FB_AcsiDATypeCell EXTENDS FB_AcsiCommonAttributeClass
```

Inputs

Name	Type	Description
xStart	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
xEnd	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
yStart	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
yEnd	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility.
Fc [▶ 366]	DWORD	Get,Set	Functional group.
fXStart	REAL	Get,Set	Value of the data attribute.
fXEnd	REAL	Get,Set	Value of the data attribute.
fYStart	REAL	Get,Set	Value of the data attribute.
fYEnd	REAL	Get,Set	Value of the data attribute.

6.3.1.1.33 FB_AcsiDATypeCheck

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeCODEDENUM [▶ 103] -> FB_AcsiDATypeCheck

```
FUNCTION_BLOCK FB_AcsiDATypeCheck EXTENDS FB_AcsiBATypeCODEDENUM
```

Properties

Name	Type	Access	Description
eValue	E_AcsiCheck [▶ 479]	Get,Set	Attribute value as enumeration type.
b0	BOOL	Get,Set	Attribute value.
b1	BOOL	Get,Set	Attribute value.
nValue	BYTE(0..3)	Get,Set	Attribute value in TwinCAT PLC BYTE format.

6.3.1.1.34 FB_AcsiDATypeClcIntvTyp

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeClcIntvTyp

```
FUNCTION_BLOCK FB_AcsiDATypeClcIntvTyp EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiClcIntvTyp [▶ 480]	Get,Set	Attribute value.

6.3.1.1.35 FB_AcsiDATypeClcMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeClcMod

```
FUNCTION_BLOCK FB_AcsiDATypeClcMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiClcMod [▶ 480]	Get,Set	Attribute value.

6.3.1.1.36 FB_AcsiDATypeClcMth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeClcMth

```
FUNCTION_BLOCK FB_AcsiDATypeClcMth EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiClcMth [▶ 481]	Get,Set	Attribute value.

6.3.1.1.37 FB_AcsiDATypeClcRfTyp

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeClcRfTyp

```
FUNCTION_BLOCK FB_AcsiDATypeClcRfTyp EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiClcRfTyp [▶ 481]	Get,Set	Attribute value.

6.3.1.1.38 FB_AcsiDATypeClcTotVA

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeClcTotVA

```
FUNCTION_BLOCK FB_AcsiDATypeClcTotVA EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiClcTotVA [▶ 482]	Get,Set	Attribute value.

6.3.1.1.39 FB_AcsiDATypeCmdQual

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeCmdQual

```
FUNCTION_BLOCK FB_AcsiDATypeCmdQual EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiCmdQual [▶ 482]	Get,Set	Attribute value.

6.3.1.1.40 FB_AcsiDATypeCtlModels

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeCtlModels

```
FUNCTION_BLOCK FB_AcsiDATypeCtlModels EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiCtlModels [▶ 482]	Get,Set	Attribute value.

6.3.1.1.41 FB_AcsiDATypeCurrency

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeVISIBLESTRING [▶ 108] -> FB_AcsiDATypeCurrency

```
FUNCTION_BLOCK FB_AcsiDATypeCurrency EXTENDS FB_AcsiBATypeVISIBLESTRING
```

Properties

Name	Type	Access	Description
sValue	STRING(3)	Get,Set	Attribute value.

6.3.1.1.42 FB_AcsiDATypeCycTrMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeCycTrMod

```
FUNCTION_BLOCK FB_AcsiDATypeCycTrMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiCycTrMod [▶ 484]	Get,Set	Attribute value.

6.3.1.1.43 FB_AcsiDATypeDatSetRef

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeVISIBLESTRING [▶ 108] -> FB_AcsiDATypeObjRef [▶ 126] -
> FB_AcsiDATypeDatSetRef

```
FUNCTION_BLOCK FB_AcsiDATypeDatSetRef EXTENDS FB_AcsiDATypeObjRef
```

Properties

Name	Type	Access	Description
ipDatSet	I_AcsiCommonDataSetClass [▶ 366]	Get	The referenced data set.

6.3.1.1.44 FB_AcsiDATypeDbpos

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeCODEDENUM [▶ 103] -> FB_AcsiDATypeDbpos

```
FUNCTION_BLOCK FB_AcsiDATypeDbpos EXTENDS FB_AcsiBATypeCODEDENUM
```

Properties

Name	Type	Access	Description
eValue	E_AcsiDbpos [▶ 484]	Get,Set	Attribute value as enumeration type.
b0	BOOL	Get,Set	Attribute value.
b1	BOOL	Get,Set	Attribute value.
nValue	BYTE(0..3)	Get,Set	Attribute value in TwinCAT PLC BYTE format.

6.3.1.1.45 FB_AcsiDATypeDir

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeDir

```
FUNCTION_BLOCK FB_AcsiDATypeDir EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiDir [▶ 484]	Get,Set	Attribute value.

6.3.1.1.46 FB_AcsiDATypeDirMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeDirMod

```
FUNCTION_BLOCK FB_AcsiDATypeDirMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiDirMod [▶ 485]	Get,Set	Attribute value.

6.3.1.1.47 FB_AcsiDATypeEEHealth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeEEHealth

```
FUNCTION_BLOCK FB_AcsiDATypeEEHealth EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiEEHealth [▶ 485]	Get,Set	Attribute value.

6.3.1.1.48 FB_AcsiDATypeEntryID

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeOCTETSTRING [▶ 108] -> FB_AcsiDATypeEntryID

```
FUNCTION_BLOCK FB_AcsiDATypeEntryID EXTENDS FB_AcsiBATypeOCTETSTRING
```

Properties

Name	Type	Access	Description
nValue	LWORD	Get,Set	Attribute value in TwinCAT PLC LWORD format. The 8 bytes of the octet string can be set or queried as a hexadecimal value. Samples: LWORD#16#1122334455667788 corresponds to OctetString: „11 22 33 44 55 66 77 88“. LWORD#16#000000000000AABB corresponds to OctetString: „00 00 00 00 00 00 AA BB“. LWORD#16#AABB000000000000 corresponds to OctetString: „AA BB 00 00 00 00 00 00“.
oValue	T_OCTET8 [▶ 558]	Get,Set	Attribute value.

6.3.1.1.49 FB_AcsiDATypeEntryTime

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeEntryTime

```
FUNCTION_BLOCK FB_AcsiDATypeEntryTime EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
tValue	T_BinaryTime [▶ 556]	Get,Set	Attribute value as a structured type.
TimeOfDay	TOD	Get,Set	Attribute value: Number of milliseconds since midnight of the current day in TwinCAT PLC TOD format.
Day	WORD	Get,Set	Attribute value: Number of days since 1 January 1984 in TwinCAT PLC WORD format.
D	DATE	Get,Set	Attribute value: Number of days since 1 January 1984 in TwinCAT PLC DATE format.
sValue	STRING(26)	Get,Set	Attribute value: Binary time information as a string with the following format: „BT#1984-01-01-00:00:00.000“.
nValue	LWORD(0..16#FFFFFFFFFF)	Get,Set	Attribute value in TwinCAT PLC LWORD format. (only the lowest 48 bits are used).

6.3.1.1.50 FB_AcsiDATypeFailMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeFailMod

```
FUNCTION_BLOCK FB_AcsiDATypeFailMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiFailMod [▶ 486]	Get,Set	Attribute value.

6.3.1.1.51 FB_AcsiDATypeFanCtl

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeFanCtl

```
FUNCTION_BLOCK FB_AcsiDATypeFanCtl EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiFanCtl [▶ 486]	Get,Set	Attribute value.

6.3.1.1.52 FB_AcsiDATypeFanCtlGen

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeFanCtlGen

```
FUNCTION_BLOCK FB_AcsiDATypeFanCtlGen EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiFanCtlGen [▶ 486]	Get,Set	Attribute value.

6.3.1.1.53 FB_AcsiDATypeFilTyp

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeFilTyp

```
FUNCTION_BLOCK FB_AcsiDATypeFilTyp EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiFilTyp [▶ 488]	Get,Set	Attribute value.

6.3.1.1.54 FB_AcsiDATypeFltLoop

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeFltLoop

```
FUNCTION_BLOCK FB_AcsiDATypeFltLoop EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiFltLoop [▶ 488]	Get,Set	Attribute value.

6.3.1.1.55 FB_AcsiDATypeGnSt

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeGnSt

```
FUNCTION_BLOCK FB_AcsiDATypeGnSt EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiGnSt [▶ 489]	Get,Set	Attribute value.

6.3.1.1.56 FB_AcsiDATypeHealth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeHealth

```
FUNCTION_BLOCK FB_AcsiDATypeHealth EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiHealth [▶ 489]	Get,Set	Attribute value.

6.3.1.1.57 FB_AcsiDATypeHvRef

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeHvRef

```
FUNCTION_BLOCK FB_AcsiDATypeHvRef EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiHvRef [▶ 490]	Get,Set	Attribute value.

6.3.1.1.58 FB_AcsiDATypeIntrDetMth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeIntrDetMth

```
FUNCTION_BLOCK FB_AcsiDATypeIntrDetMth EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiIntrDetMth [▶ 490]	Get,Set	Attribute value.

6.3.1.1.59 FB_AcsiDATypeLastApplError

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeApplError

```
FUNCTION_BLOCK FB_AcsiDATypeLastApplError EXTENDS FB_AcsiCommonAttributeClass IMPLEMENTS I_AcsiDATyp
eLastApplError
```

Interfaces

Type	Description
I_AcsiDATypeLastApplError [▶ 374]	Interface pointer of the LastApplError object instance.

Inputs

Name	Type	Description
CntrlObj	FB_AcsiDATypeVisString129 [▶ 151]	Data attribute.
Error	FB_AcsiDATypeApplError [▶ 112]	Data attribute.
Origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
AddCause	FB_AcsiDATypeApplAddCause [▶ 112]	Data attribute.

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility.
Fc [▶ 366]	DWORD	Get,Set	Functional group.
sCntrlObj	STRING(129)	Get,Set	Attribute value.
eError	E_AcsiApplError [▶ 478]	Get,Set	Attribute value.
stOrigin	ST_AcsiOriginator [▶ 538]	Get,Set	Attribute value.
nCtlNum	BYTE	Get,Set	Attribute value.
eAddCause	E_AcsiApplAddCause [▶ 476]	Get,Set	Attribute value.

6.3.1.1.60 FB_AcsiDATypeLevMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeLevMod

```
FUNCTION_BLOCK FB_AcsiDATypeLevMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiLevMod [▶ 491]	Get,Set	Attribute value.

6.3.1.1.61 FB_AcsiDATypeLivDeaMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeLivDeaMod

```
FUNCTION_BLOCK FB_AcsiDATypeLivDeaMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiLivDeaMod [▶ 491]	Get,Set	Attribute value.

6.3.1.1.62 FB_AcsiDATypeMechHealth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeMechHealth

```
FUNCTION_BLOCK FB_AcsiDATypeMechHealth EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiMechHealth [▶ 492]	Get,Set	Attribute value.

6.3.1.1.63 FB_AcsiDATypeMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154]-> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeMod

```
FUNCTION_BLOCK FB_AcsiDATypeMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiMod [▶ 492]	Get,Set	Attribute value.

6.3.1.1.64 FB_AcsiDATypeMonth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeMonth

```
FUNCTION_BLOCK FB_AcsiDATypeMonth EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiMonth [▶ 493]	Get,Set	Attribute value.

6.3.1.1.65 FB_AcsiDATypeMultiplier

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeMultiplier

```
FUNCTION_BLOCK FB_AcsiDATypeMultiplier EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiMultiplier [▶ 493]	Get,Set	Attribute value.

6.3.1.1.66 FB_AcsiDATypeObjectName

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeVISIBLESTRING [▶ 108] -> FB_AcsiDATypeObjectName

```
FUNCTION_BLOCK FB_AcsiDATypeObjectName EXTENDS FB_AcsiBATypeVISIBLESTRING
```

Properties

Name	Type	Access	Description
sValue	T_AcsiObjectName [▶ 555]	Get,Set	Attribute value.

6.3.1.1.67 FB_AcsiDATypeObjRef

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeVISIBLESTRING [▶ 108] -> FB_AcsiDATypeObjRef

```
FUNCTION_BLOCK FB_AcsiDATypeObjRef EXTENDS FB_AcsiBATypeVISIBLESTRING
```

Properties

Name	Type	Access	Description
sValue	T_AcsiObjectReference [▶ 555]	Get,Set	Attribute value.

6.3.1.1.68 FB_AcsiDATypeOccPer

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeOccPer

```
FUNCTION_BLOCK FB_AcsiDATypeOccPer EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiOccPer [▶ 494]	Get,Set	Attribute value.

6.3.1.1.69 FB_AcsiDATypeOccType

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeOccType

```
FUNCTION_BLOCK FB_AcsiDATypeOccType EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiOccType [▶ 495]	Get,Set	Attribute value.

6.3.1.1.70 FB_AcsiDATypeOctet8

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeOCTETSTRING [▶ 108] -> FB_AcsiDATypeOctet8

```
FUNCTION_BLOCK FB_AcsiDATypeOctet8 EXTENDS FB_AcsiBATypeOCTETSTRING
```

Properties

Name	Type	Access	Description
oValue	T_OCTET8 [▶ 558]	Get,Set	Attribute value.

6.3.1.1.71 FB_AcsiDATypeOctet16

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeOCTETSTRING [▶ 108] -> FB_AcsiDATypeOctet16

```
FUNCTION_BLOCK FB_AcsiDATypeOctet16 EXTENDS FB_AcsiBATypeOCTETSTRING
```

Properties

Name	Type	Access	Description
oValue	T_OCTET16 [▶ 558]	Get,Set	Attribute value.

6.3.1.1.72 FB_AcsiDATypeOctet32

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeOCTETSTRING [▶ 108] -> FB_AcsiDATypeOctet32

```
FUNCTION_BLOCK FB_AcsiDATypeOctet32 EXTENDS FB_AcsiBATypeOCTETSTRING
```

Properties

Name	Type	Access	Description
oValue	T_OCTET32 [▶ 559]	Get,Set	Attribute value.

6.3.1.1.73 FB_AcsiDATypeOctet64

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeOCTETSTRING [▶ 108] -> FB_AcsiDATypeOctet64

```
FUNCTION_BLOCK FB_AcsiDATypeOctet64 EXTENDS FB_AcsiBATypeOCTETSTRING
```

Properties

Name	Type	Access	Description
oValue	T_OCTET64 [▶ 559]	Get,Set	Attribute value.

6.3.1.1.74 FB_AcsiDATypeOpModRect

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeOpModRect

```
FUNCTION_BLOCK FB_AcsiDATypeOpModRect EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiOpModRect [▶ 495]	Get,Set	Attribute value.

6.3.1.1.75 FB_AcsiDATypeOpModSyn

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeOpModSyn

```
FUNCTION_BLOCK FB_AcsiDATypeOpModSyn EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiOpModSyn [▶ 495]	Get,Set	Attribute value.

6.3.1.1.76 FB_AcsiDATypeOptFlds

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeCODEDENUM [▶ 103] -> FB_AcsiDATypeOptFlds

```
FUNCTION_BLOCK FB_AcsiDATypeOptFlds EXTENDS FB_AcsiBATypeCODEDENUM IMPLEMENTS I_AcsiDATypeOptFlds
```

Properties

Name	Type	Access	Description
cValue	ST_AcsiOptionalFields [▶ 536]	Get,Set	Attribute value as a structured type.
SequenceNumber	BOOL	Get,Set	Attribute value.
ReportTimeStamp	BOOL	Get,Set	Attribute value.
ReasonForInclusion	BOOL	Get,Set	Attribute value.
DataSetName	BOOL	Get,Set	Attribute value.
DataReference	BOOL	Get,Set	Attribute value.
BufferOverflow	BOOL	Get,Set	Attribute value.
EntryID	BOOL	Get,Set	Attribute value.
ConfRevision	BOOL	Get,Set	Attribute value.
Segmentation	BOOL	Get,Set	Attribute value.
nValue	WORD(0..1023)	Get,Set	Information about the optional fields transferred in the report in TwinCAT PLC WORD format (only the lower 10 bits are used).

6.3.1.1.77 FB_AcsiDATypeOrCategory

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeOrCategory

```
FUNCTION_BLOCK FB_AcsiDATypeOrCategory EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiOrCategory [▶ 496]	Get,Set	Attribute value.

6.3.1.1.78 FB_AcsiDATypeOrIdent

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeOCTETSTRING [▶ 108] -> FB_AcsiDATypeOrIdent

```
FUNCTION_BLOCK FB_AcsiDATypeOrIdent EXTENDS FB_AcsiBATypeOCTETSTRING
```

Properties

Name	Type	Access	Description
oValue	T_OCTET64 [▶ 559]	Get,Set	Attribute value.

6.3.1.1.79 FB_AcsiDATypeOriginator

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeOriginator

```
FUNCTION_BLOCK FB_AcsiDATypeOriginator EXTENDS FB_AcsiCommonAttributeClass
```

Inputs

Name	Type	Description
orCat	FB_AcsiDATypeOrCategory [▶ 129]	Data attribute.
orIdent	FB_AcsiDATypeOrIdent [▶ 129]	Data attribute.

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility.
Fc [▶ 366]	DWORD	Get,Set	Functional group.
stOrigin	ST_AcsiOriginator [▶ 538]	Get,Set	Attribute value as a structured type.
eOrCat	E_AcsiOrCategory [▶ 496]	Get,Set	Attribute value.
eOrIdent	T_OCTET64 [▶ 559]	Get,Set	Attribute value.

6.3.1.1.80 FB_AcsiDATypeParColMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeParColMod

```
FUNCTION_BLOCK FB_AcsiDATypeParColMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiParColMod [▶ 496]	Get,Set	Attribute value.

6.3.1.1.81 FB_AcsiDATypeParMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeParMod

```
FUNCTION_BLOCK FB_AcsiDATypeParMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiParMod [▶ 497]	Get,Set	Attribute value.

6.3.1.1.82 FB_AcsiDATypeParTraMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeParTraMod

```
FUNCTION_BLOCK FB_AcsiDATypeParTraMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiParTraMod [▶ 497]	Get,Set	Attribute value.

6.3.1.1.83 FB_AcsiDATypePFSign

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypePFSign

```
FUNCTION_BLOCK FB_AcsiDATypePFSign EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiPFSign [▶ 498]	Get,Set	Attribute value.

6.3.1.1.84 FB_AcsiDATypePhsRef

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypePhsRef

```
FUNCTION_BLOCK FB_AcsiDATypePhsRef EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiPhsRef [▶ 498]	Get,Set	Attribute value.

6.3.1.1.85 FB_AcsiDATypePhyComAddr

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypePhyComAddr

```
FUNCTION_BLOCK FB_AcsiDATypePhyComAddr EXTENDS FB_AcsiCommonAttributeClass
```

Inputs

Name	Type	Description
Addr	FB_AcsiDATypeAddr [▶ 109]	Data attribute.
PRIORITY	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
VID	FB_AcsiBATypeINT16U [▶ 105]	Data attribute.
APPID	FB_AcsiBATypeINT16U [▶ 105]	Data attribute.

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility.
Fc [▶ 366]	DWORD	Get,Set	Functional group.
oAddr	T_OCTET6 [▶ 558]	Get,Set	Attribute value.
nPRIORITY	BYTE(0..7)	Get,Set	Attribute value.
nVID	WORD(0..4095)	Get,Set	Attribute value.
nAPPID	WORD	Get,Set	Attribute value.

6.3.1.1.86 FB_AcsiDATypePhyHealth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypePhyHealth

```
FUNCTION_BLOCK FB_AcsiDATypePhyHealth EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiPhyHealth [▶ 498]	Get,Set	Attribute value.

6.3.1.1.87 FB_AcsiDATypePIDAlg

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypePIDAlg

```
FUNCTION_BLOCK FB_AcsiDATypePIDAlg EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiPIDAlg [▶ 499]	Get,Set	Attribute value.

6.3.1.1.88 FB_AcsiDATypePmpCtl

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypePmpCtl

```
FUNCTION_BLOCK FB_AcsiDATypePmpCtl EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiPmpCtl [▶ 499]	Get,Set	Attribute value.

6.3.1.1.89 FB_AcsiDATypePmpCtlGen

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypePmpCtlGen

```
FUNCTION_BLOCK FB_AcsiDATypePmpCtlGen EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiPmpCtlGen [▶ 499]	Get,Set	Attribute value.

6.3.1.1.90 FB_AcsiDATypePoint

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160]-> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypePoint

```
FUNCTION_BLOCK FB_AcsiDATypePoint EXTENDS FB_AcsiCommonAttributeClass
```

Inputs

Name	Type	Description
xVal	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
yVal	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
zVal	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility.
Fc [▶ 366]	DWORD	Get,Set	Functional group.
fXVal	REAL	Get,Set	Attribute value.
fYVal	REAL	Get,Set	Attribute value.
fZVal	REAL	Get,Set	Attribute value.

6.3.1.1.91 FB_AcsiDATypePolQty

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypePolQty

```
FUNCTION_BLOCK FB_AcsiDATypePolQty EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiPolQty [▶ 500]	Get,Set	Attribute value.

6.3.1.1.92 FB_AcsiDATypePOWCap

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypePOWCap

```
FUNCTION_BLOCK FB_AcsiDATypePOWCap EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiPOWCap [▶ 500]	Get,Set	Attribute value.

6.3.1.1.93 FB_AcsiDATypePulseConfig

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypePulseConfig

```
FUNCTION_BLOCK FB_AcsiDATypePulseConfig EXTENDS FB_AcsiCommonAttributeClass
```

Inputs

Name	Type	Description
cmdQual	FB_AcsiDATypeCmdQual [▶ 116]	Data attribute.
onDur	FB_AcsiBATypeINT32U [▶ 106]	Data attribute.
offDur	FB_AcsiBATypeINT32U [▶ 106]	Data attribute.
numPls	FB_AcsiBATypeINT32U [▶ 106]	Data attribute.

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility.
Fc [▶ 366]	DWORD	Get,Set	Functional group.
eCmdQual	E_AcsiCmdQual [▶ 482]	Get,Set	Attribute value.
nOnDur	DWORD	Get,Set	Attribute value.
nOffDur	DWORD	Get,Set	Attribute value.
nNumPls	DWORD	Get,Set	Attribute value.

6.3.1.1.94 FB_AcsiDATypeQuality

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeQuality

```
FUNCTION_BLOCK FB_AcsiDATypeQuality EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
qValue	ST_AcsiQuality [▶ 541]	Get,Set	Attribute value as a structured type.
eValidity	E_AcsiQualityValidity [▶ 501]	Get,Set	Attribute value.
Overflow	BOOL	Get,Set	Attribute value.
OutOfRange	BOOL	Get,Set	Attribute value.
BadReference	BOOL	Get,Set	Attribute value.
Oscillatory	BOOL	Get,Set	Attribute value.
Failure	BOOL	Get,Set	Attribute value.
OldData	BOOL	Get,Set	Attribute value.
Inconsistent	BOOL	Get,Set	Attribute value.
Inaccurate	BOOL	Get,Set	Attribute value.
Test	BOOL	Get,Set	Attribute value.
eSource	E_AcsiQualitySource [▶ 501]	Get,Set	Attribute value.
OperatorBlocked	BOOL	Get,Set	Attribute value.
nValue	WORD(0..8191)	Get,Set	Attribute value in TwinCAT PLC WORD format.

6.3.1.1.95 FB_AcsiDATypeRange

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeRange

```
FUNCTION_BLOCK FB_AcsiDATypeRange EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiRange [▶ 501]	Get,Set	Attribute value.

6.3.1.1.96 FB_AcsiDATypeRangeConfig

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeRangeConfig

```
FUNCTION_BLOCK FB_AcsiDATypeRangeConfig EXTENDS FB_AcsiCommonAttributeClass
```

Inputs

Name	Type	Description
hhLim	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
hLim	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
lLim	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
llLim	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
min_	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
max_	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
limDb	FB_AcsiBATypeINT32U [▶ 106]	Data attribute.

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility.
Fc [▶ 366]	DWORD	Get,Set	Functional group.
fHHLim	REAL	Get,Set	Attribute value.
fHLim	REAL	Get,Set	Attribute value.
fLLim	REAL	Get,Set	Attribute value.
fLLLim	REAL	Get,Set	Attribute value.
fMax	REAL	Get,Set	Attribute value.
fMin	REAL	Get,Set	Attribute value.
nLimDb	DWORD(0..100000)	Get,Set	Attribute value.

6.3.1.1.97 FB_AcsiDATypeRcdMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeRcdMod

```
FUNCTION_BLOCK FB_AcsiDATypeRcdMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiRcdMod [▶ 502]	Get,Set	Attribute value.

6.3.1.1.98 FB_AcsiDATypeReasonCode

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160]-> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeReasonCode

```
FUNCTION_BLOCK FB_AcsiDATypeReasonCode EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
cValue	ST_AcsiReasonCode [▶ 542]	Get,Set	Attribute value as a structured type.
DataChange	BOOL	Get,Set	Attribute value.
QualityChange	BOOL	Get,Set	Attribute value.
DataUpdate	BOOL	Get,Set	Attribute value.
Integrity	BOOL	Get,Set	Attribute value.
GeneralInterrogation	BOOL	Get,Set	Attribute value.
ApplicationTrigger	BOOL	Get,Set	Attribute value.
nValue	BYTE(0..127)	Get,Set	Attribute value in TwinCAT PLC BYTE format.

6.3.1.1.99 FB_AcsiDATypeReTrMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeReTrMod

```
FUNCTION_BLOCK FB_AcsiDATypeReTrMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiReTrMod [▶ 502]	Get,Set	Attribute value.

6.3.1.1.100 FB_AcsiDATypeRotDir

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeRotDir

```
FUNCTION_BLOCK FB_AcsiDATypeRotDir EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiRotDir [▶ 503]	Get,Set	Attribute value.

6.3.1.1.101 FB_AcsiDATypeRstMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeRstMod

```
FUNCTION_BLOCK FB_AcsiDATypeRstMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiRstMod [▶ 503]	Get,Set	Attribute value.

6.3.1.1.102 FB_AcsiDATypeSamplingMode

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeSamplingMode

```
FUNCTION_BLOCK FB_AcsiDATypeSamplingMode EXTENDS FB_AcsiBATypeENUMERATED IMPLEMENTS I_AcsiDATypeSamp
lingMode
```

Properties

Name	Type	Access	Description
eValue	E_AcsiSamplingMode [▶ 503]	Get,Set	Attribute value.

6.3.1.1.103 FB_AcsiDATypeSboClasses

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeSboClasses

```
FUNCTION_BLOCK FB_AcsiDATypeSboClasses EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiSboClasses [▶ 504]	Get,Set	Attribute value.

6.3.1.1.104 FB_AcsiDATypeScaledValueConfig

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152]-> FB_AcsiDATypeScaledValueConfig

```
FUNCTION_BLOCK FB_AcsiDATypeScaledValueConfig EXTENDS FB_AcsiCommonAttributeClass IMPLEMENTS I_AcsiD
ATypeScaledValueConfig
```

Interfaces

Type	Description
I_AcsiDATypeScaledValueConfig [▶ 361]	Interface pointer of the ScaledValueConfig object instance.

Inputs

Name	Type	Description
scaleFactor	FB_AcsiBATypeFLOAT32 [▶ 104]	Data attribute.
offset	FB_AcsiBATypeFLOAT32 [▶ 104]	Data attribute.

Methods

Name	Description
ScaleAToP [▶ 361]	Scales an attribute value into a process value.
ScalePToA [▶ 361]	Scales a process value into an attribute value.

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility.
Fc [▶ 366]	DWORD	Get,Set	Functional group.
fOffset	REAL	Get,Set	Attribute value: Scaling offset.
fScaleFactor	REAL	Get,Set	Attribute value: Scaling factor.

6.3.1.1.105 FB_AcsiDATypeSeqT

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeSeqT

```
FUNCTION_BLOCK FB_AcsiDATypeSeqT EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiSeqT [▶ 504]	Get,Set	Attribute value.

6.3.1.1.106 FB_AcsiDATypeServiceError

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeServiceError

```
FUNCTION_BLOCK FB_AcsiDATypeServiceError EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiServiceError [▶ 505]	Get,Set	Attribute value.

6.3.1.1.107 FB_AcsiDATypeSetCharact

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154]-> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeSetCharact

```
FUNCTION_BLOCK FB_AcsiDATypeSetCharact EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiSetCharact [▶ 506]	Get,Set	Attribute value.

6.3.1.1.108 FB_AcsiDATypeSev

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeSev

```
FUNCTION_BLOCK FB_AcsiDATypeSev EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiSev [▶ 507]	Get,Set	Attribute value.

6.3.1.1.109 FB_AcsiDATypeShOpCap

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeShOpCap

```
FUNCTION_BLOCK FB_AcsiDATypeShOpCap EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiShOpCap [▶ 508]	Get,Set	Attribute value.

6.3.1.1.110 FB_AcsiDATypeSIUnit

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeSIUnit

```
FUNCTION_BLOCK FB_AcsiDATypeSIUnit EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiSIUnit [▶ 508]	Get,Set	Attribute value.

6.3.1.1.111 FB_AcsiDATypeSptEndSt

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeSptEndSt

```
FUNCTION_BLOCK FB_AcsiDATypeSptEndSt EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiSptEndSt [▶ 510]	Get,Set	Attribute value.

6.3.1.1.112 FB_AcsiDATypeStClcTun

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeStClcTun

```
FUNCTION_BLOCK FB_AcsiDATypeStClcTun EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiStClcTun [▶ 511]	Get,Set	Attribute value.

6.3.1.1.113 FB_AcsiDATypeStrWeekDay

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeStrWeekDay

```
FUNCTION_BLOCK FB_AcsiDATypeStrWeekDay EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiStrWeekDay [▶ 512]	Get,Set	Attribute value.

6.3.1.1.114 FB_AcsiDATypeSvOptFlds

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeCODEDENUM [▶ 103] -> FB_AcsiDATypeSvOptFlds

```
FUNCTION_BLOCK FB_AcsiDATypeSvOptFlds EXTENDS FB_AcsiBATypeCODEDENUM IMPLEMENTS I_AcsiDATypeSvOptFld
s
```

Properties

Name	Type	Access	Description
cValue	ST_AcsiOptionalSvFields [▶ 537]	Get,Set	Attribute value as a structured type.
RefreshTime	BOOL	Get,Set	Attribute value.
SampleSynchronized	BOOL	Get,Set	Attribute value.
SampleRate	BOOL	Get,Set	Attribute value.
DataSetName	BOOL	Get,Set	Attribute value.
Security	BOOL	Get,Set	Attribute value.
SampleMode	BOOL	Get,Set	Attribute value.
SynchSourceIdentity	BOOL	Get,Set	Attribute value.
ConfRevision	BOOL	Get,Set	Attribute value.

6.3.1.1.115 FB_AcsiDATypeSwOpCap

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeSwOpCap

```
FUNCTION_BLOCK FB_AcsiDATypeSwOpCap EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiSwOpCap [▶ 512]	Get,Set	Attribute value.

6.3.1.1.116 FB_AcsiDATypeSwTyp

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeSwTyp

```
FUNCTION_BLOCK FB_AcsiDATypeSwTyp EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiSwTyp [▶ 513]	Get,Set	Attribute value.

6.3.1.1.117 FB_AcsiDATypeTcmd

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeCODEDENUM [▶ 103] -> FB_AcsiDATypeTcmd

```
FUNCTION_BLOCK FB_AcsiDATypeTcmd EXTENDS FB_AcsiBATypeCODEDENUM
```

Properties

Name	Type	Access	Description
eValue	E_AcsiTcmd [▶ 513]	Get,Set	Attribute value as enumeration type.
b0	BOOL	Get,Set	Attribute value.
b1	BOOL	Get,Set	Attribute value.
nValue	BYTE(0..3)	Get,Set	Attribute value in TwinCAT PLC BYTE format.

6.3.1.1.118 FB_AcsiDATypeTimeStamp

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeTimeStamp

```
FUNCTION_BLOCK FB_AcsiDATypeTimeStamp EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
tValue	T_UtcTime [▶ 560]	Get,Set	Attribute value: Time information as a structured type.
SecondSinceEpoch	DT	Get,Set	Attribute value.
FractionOfSecond	T_UINT24 [▶ 559]	Get,Set	Attribute value.
Quality	T_UtcTimeQuality [▶ 560]	Get,Set	Attribute value: Quality as a structured type.
LeapSecondsKnown	BOOL	Get,Set	Attribute value.
ClockFailure	BOOL	Get,Set	Attribute value.
ClockNotSynchronized	BOOL	Get,Set	Attribute value.
eAccuracy	E_UtcTimeAccuracy [▶ 527]	Get,Set	Attribute value.
sValue	STRING(39)	Get,Set	Attribute value: Time information as a string with the following format: "UT#1970-01-01-00:00:00.000000000|000|0".
nValue	LWORD	Get,Set	Attribute value: Time information in TwinCAT PLC LWORD format.

6.3.1.1.119 FB_AcsiDATypeTmSyn

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeTmSyn

```
FUNCTION_BLOCK FB_AcsiDATypeTmSyn EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiTmSyn [▶ 513]	Get,Set	Attribute value.

6.3.1.1.120 FB_AcsiDATypeTnkTyp

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeTnkTyp

```
FUNCTION_BLOCK FB_AcsiDATypeTnkTyp EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiTnkTyp [▶ 514]	Get,Set	Attribute value.

6.3.1.1.121 FB_AcsiDATypeTpcRxMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeTpcRxMod

```
FUNCTION_BLOCK FB_AcsiDATypeTpcRxMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiTpcRxMod [▶ 514]	Get,Set	Attribute value.

6.3.1.1.122 FB_AcsiDATypeTpcTxMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeTpcTxMod

```
FUNCTION_BLOCK FB_AcsiDATypeTpcTxMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiTpcTxMod [▶ 515]	Get,Set	Attribute value.

6.3.1.1.123 FB_AcsiDATypeTrBeh

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeTrBeh

```
FUNCTION_BLOCK FB_AcsiDATypeTrBeh EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiTrBeh [▶ 515]	Get,Set	Attribute value.

6.3.1.1.124 FB_AcsiDATypeTrgMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeTrgMod

```
FUNCTION_BLOCK FB_AcsiDATypeTrgMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiTrgMod [▶ 516]	Get,Set	Attribute value.

6.3.1.1.125 FB_AcsiDATypeTriggerConditions

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeTriggerConditions

```
FUNCTION_BLOCK FB_AcsiDATypeTriggerConditions EXTENDS FB_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
cValue	ST_AcsiTriggerConditions [▶ 544]	Get,Set	Attribute value as a structured type.
DataChange	BOOL	Get,Set	Attribute value.
QualityChange	BOOL	Get,Set	Attribute value.
DataUpdate	BOOL	Get,Set	Attribute value.
Integrity	BOOL	Get,Set	Attribute value.
GeneralInterrogation	BOOL	Get,Set	Attribute value.
nValue	BYTE(0..63)	Get,Set	Attribute value in TwinCAT PLC BYTE format.

6.3.1.1.126 FB_AcsiDATypeTrMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeTrMod

```
FUNCTION_BLOCK FB_AcsiDATypeTrMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiTrMod [▶ 516]	Get,Set	Attribute value.

6.3.1.1.127 FB_AcsiDATypeTypRsCrv

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeTypRsCrv

```
FUNCTION_BLOCK FB_AcsiDATypeTypRsCrv EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiTypRsCrv [▶ 516]	Get,Set	Attribute value.

6.3.1.1.128 FB_AcsiDATypeUnbDetMth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeUnbDetMth

```
FUNCTION_BLOCK FB_AcsiDATypeUnbDetMth EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiUnbDetMth [▶ 517]	Get,Set	Attribute value.

6.3.1.1.129 FB_AcsiDATypeUnBlkMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeUnBlkMod

```
FUNCTION_BLOCK FB_AcsiDATypeUnBlkMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiUnBlkMod [▶ 517]	Get,Set	Attribute value.

6.3.1.1.130 FB_AcsiDATypeUnicode255

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeUNICODESTRING [▶ 108] -> FB_AcsiDATypeUnicode255

```
FUNCTION_BLOCK FB_AcsiBATypeUnicode255 EXTENDS FB_AcsiBATypeUNICODESTRING
```

Properties

Name	Type	Access	Description
sValue	WSTRING(255)	Get,Set	Attribute value.

6.3.1.1.131 FB_AcsiDATypeUnit

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeUnit

```
FUNCTION_BLOCK FB_AcsiDATypeUnit EXTENDS FB_AcsiCommonAttributeClass IMPLEMENTS I_AcsiDATypeUnit
```

Interfaces

Type	Description
I_AcsiDATypeUnit [▶ 362]	Interface pointer of the unit object instance.

Inputs

Name	Type	Description
SIUnit	FB_AcsiDATypeSIUnit [▶ 142]	Data attribute.
multiplier	FB_AcsiDATypeMultiplier [▶ 125]	Data attribute.

Methods

Name	Description
ScaleAToP [▶ 362]	Scales an attribute value into a process value.
ScalePToA [▶ 362]	Scales a process value into an attribute value.

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility.
Fc [▶ 366]	DWORD	Get,Set	Functional group.
eMultiplier	E_AcsiMultiplier [▶ 493]	Get,Set	Attribute value: Multiplier.
eSIUnit	E_AcsiSIUnit [▶ 508]	Get,Set	Attribute value: SI unit.

6.3.1.1.132 FB_AcsiDATypeValWithTrans

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeValWithTrans

```
FUNCTION_BLOCK FB_AcsiDATypeValWithTrans EXTENDS FB_AcsiCommonAttributeClass
```

Inputs

Name	Type	Description
posVal	FB_AcsiBATypeINT8 [▶ 104]	Data attribute.
transInd	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility.
Fc [▶ 366]	DWORD	Get,Set	Functional group.
iPosVal	SINT(-64..63)	Get,Set	Attribute value.
bTransInd	BOOL	Get,Set	Attribute value.

6.3.1.1.133 FB_AcsiDATypeVector

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiDATypeVector

```
FUNCTION_BLOCK FB_AcsiDATypeVector EXTENDS FB_AcsiCommonAttributeClass
```

Inputs

Name	Type	Description
mag	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
ang	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility.
Fc [▶ 366]	DWORD	Get,Set	Functional group.
fMag	REAL	Get,Set	Attribute value.
iMag	DINT	Get,Set	Attribute value.
fAng	REAL	Get,Set	Attribute value.
iAng	DINT	Get,Set	Attribute value.

6.3.1.1.134 FB_AcsiDATypeVisString64

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeVISIBLESTRING [▶ 108] -> FB_AcsiDATypeVisString64

```
FUNCTION_BLOCK FB_AcsiDATypeVisString64 EXTENDS FB_AcsiBATypeVISIBLESTRING
```

Properties

Name	Type	Access	Description
sValue	STRING(64)	Get,Set	Attribute value.

6.3.1.1.135 FB_AcsiDATypeVisString129

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeVISIBLESTRING [▶ 108] -> FB_AcsiDATypeVisString129

```
FUNCTION_BLOCK FB_AcsiDATypeVisString129 EXTENDS FB_AcsiBATypeVISIBLESTRING
```

Properties

Name	Type	Access	Description
sValue	STRING(129)	Get,Set	Attribute value.

6.3.1.1.136 FB_AcsiDATypeVisString255

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeVISIBLESTRING [▶ 108] -> FB_AcsiDATypeVisString255

```
FUNCTION_BLOCK FB_AcsiDATypeVisString255 EXTENDS FB_AcsiBATypeVISIBLESTRING
```

Properties

Name	Type	Access	Description
sValue	STRING(255)	Get,Set	Attribute value.

6.3.1.1.137 FB_AcsiDATypeWeekDay

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeWeekDay

```
FUNCTION_BLOCK FB_AcsiDATypeWeekDay EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiWeekDay [▶ 518]	Get,Set	Attribute value as enumeration type.

6.3.1.1.138 FB_AcsiDATypeWeiMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiDATypeWeiMod

```
FUNCTION_BLOCK FB_AcsiDATypeWeiMod EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

Name	Type	Access	Description
eValue	E_AcsiWeiMod [▶ 519]	Get,Set	Attribute value.

6.3.1.2 FB_AcsiCommonAttributeClass

The function block FB_AcsiCommonAttributeClass is one of the basic function blocks of the TwinCAT IEC 61850 data model. Each data attribute instance (data attribute, DA) generated by the TwinCAT Telecontrol Configurator is derived/extended from FB_AcsiCommonAttributeClass.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass

```
FUNCTION_BLOCK FB_AcsiCommonAttributeClass EXTENDS FB_AcsiCommonDataClass IMPLEMENTS I_AcsiCommonAtt
ributeClass
```

Interfaces

Type	Description
I_AcsiCommonAttributeClass [▶ 362]	Interface of the data attribute instance.

Methods

Name	Description
AddAttributeToContainer [▶ 363]	Adds a data attribute object to the container of another data attribute object

Properties

-

6.3.1.3 FB_AcsiCommonBufferedReportControlBlockClass

The function block FB_AcsiCommonBufferedReportControlClass is one of the basic function blocks of the TwinCAT IEC 61850 data model. Each report control block instance for buffered reports (buffered report control block, BRCB) generated by the TwinCAT Telecontrol Configurator is derived/extended from FB_AcsiCommonBufferedReportControlBlockClass.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonBufferedReportControlBlockClass

```
FUNCTION_BLOCK FB_AcsiCommonBufferedReportControlBlockClass EXTENDS FB_AcsiCommonControlBlockClass I
MPLEMENTS I_AcsiCommonBufferedReportControlBlockClass
```

Interfaces

Type	Description
I_AcsiCommonBufferedReportControlClass [▶ 363]	Interface of the report control block instance.

Methods

-

Properties

-

6.3.1.4 FB_AcsiCommonControlBlockClass

The function block FB_AcsiCommonControlBlockClass belongs to the basic function blocks of the TwinCAT IEC 61850 data model. The function block serves as the basis for all control block function blocks.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass

```
FUNCTION_BLOCK FB_AcsiCommonControlBlockClass EXTENDS FB_AcsiCommonDataClass IMPLEMENTS I_AcsiCommon
ControlBlockClass
```

Interfaces

Type	Description
I_AcsiCommonControlBlockClass [▶ 363]	Interface of the general control block.

6.3.1.5 FB_AcsiCommonControllableDataClass

The function block FB_AcsiCommonControllableDataClass is one of the basic function blocks of the TwinCAT IEC 61850 data model. Each switch control instances generated by the TwinCAT Telecontrol Configurator (common data class, CDC: SPC, DPC, INC,...) is derived/extended from FB_AcsiCommonControllableDataClass.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControllableDataClass

```
FUNCTION_BLOCK FB_AcsiCommonControllableDataClass EXTENDS FB_AcsiCommonDataClass IMPLEMENTS I_AcsiCo
mmonControllableDataClass
```

Interfaces

Type	Description
I_AcsiCommonControllableDataClass [▶ 364]	Interface of the object instance for the switch control.

Methods

-

Properties

-

6.3.1.6 FB_AcsiCommonDataClass

The function block FB_AcsiCommonDataClass is one of the basic function blocks of the TwinCAT IEC 61850 data model. Each data object instance (DO) generated by the TwinCAT Telecontrol Configurator is derived/extended by FB_AcsiCommonDataClass.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass

```
FUNCTION_BLOCK FB_AcsiCommonDataClass EXTENDS FB_AcsiCommonNodeClass IMPLEMENTS I_AcsiCommonDataClas
s
```

Interfaces

Type	Description
I_AcsiCommonDataClass [▶ 364]	Interface of the data object instance.

Methods

Name	Description
AddAttributeToContainer [▶ 364]	Adds a data attribute object to the container of another data object
AddDataToContainer [▶ 365]	Adds a data object to the container of another data object
AddToDataSet [▶ 365]	Adds a data object to the member container of the data set
DeleteFromDataSet [▶ 365]	Removes a data object from the member container of the data set

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility
Fc [▶ 366]	DWORD	Get,Set	Functional group
TrgOp [▶ 366]	DWORD	Get,Set	Report trigger

6.3.1.7 FB_AcsiCommonDataSetClass

The function block FB_AcsiCommonDataSetClass is one of the basic function blocks of the TwinCAT IEC 61850 data model. Each dataset instance generated by the TwinCAT Telecontrol Configurator is derived/extended from FB_AcsiCommonDataSetClass.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataSetClass

```
FUNCTION_BLOCK FB_AcsiCommonDataSetClass EXTENDS FB_AcsiCommonNodeClass IMPLEMENTS I_AcsiCommonDataS
etClass
VAR_OUTPUT
    stInfo : ST_AcsiDataSetClassInfo;
END_VAR
```

Interfaces

Type	Description
I_AcsiCommonDataSetClass [▶ 366]	Interface of the dataset instance.

Outputs

Name	Type	Description
stInfo	ST_AcsiDataSetClassInfo [▶ 533]	Status and statistics data of the data set instance.

Methods

Name	Description
AddMember [▶ 367]	Adds a new data object to the member container
DeleteMember [▶ 367]	Removes a data object from the member container
DeleteAllMembers [▶ 367]	Removes all data objects from the member container

Methods for event handling

Name	Description
OnInit [▶ 367]	Initializes/configures the data objects in the member container

Properties

Name	Type	Access	Description
nMembers	UDINT	Get	Number of data objects in the member container

6.3.1.8 FB_AcsiCommonGooseControlBlockClass

The function block FB_AcsiCommonGooseControlBlockClass belongs to the basic function blocks of the TwinCAT IEC 61850 data model. Each GOOSE control block instance generated by the TwinCAT Telecontrol Configurator has been derived/extended from FB_AcsiCommonGooseControlBlockClass.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonGooseControlBlockClass

```
FUNCTION_BLOCK FB_AcsiCommonGooseControlBlockClass EXTENDS FB_AcsiCommonControlBlockClass IMPLEMENTS
 I_AcsiCommonGooseControlBlockClass
```

Interfaces

Type	Description
I_AcsiCommonGooseControlBlockClass [▶ 368]	Interface of the GOOSE control block instance.

6.3.1.9 FB_AcsiCommonIEDGroupClass

The function block FB_AcsiCommonIEDGroupClass belongs to the basic function blocks of the TwinCAT IEC 61850 data models. Objects derived/extended from FB_AcsiCommonIEDGroupClass represent a group of several IEDs. All IEDs in this group use the same "Real-Time Ethernet Adapter (Multiple Protocol Handler)" for communication.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonIEDGroupClass

```
FUNCTION_BLOCK FB_AcsiCommonIEDGroupClass IMPLEMENTS I_AcsiCommonIEDGroupClass
VAR_OUTPUT
    stInfo : ST_AcsiIEDGroupClassInfo;
END_VAR
```

Interfaces

Type	Description
I_AcsiCommonIEDGroupClass [▶ 368]	Interface of the object instance of a group of IEDs.

Outputs

Name	Type	Description
stInfo	ST_AcsiIEDGroupClassInfo [▶ 534]	Status and statistical data of the IED group.

Methods

Name	Description
AddIEDToContainer [▶ 156]	Adds a new device to the container of the IED group.

Properties

Name	Type	Access	Description
nIEDs	UDINT	Get	Number of IEDs in the container.
sObjectName	T_AcsiObjectName [▶ 555]	Get, Set	Object name.
Day	T_AcsiTag [▶ 556]	Get, Set	Additional object information.

6.3.1.9.1 AddIEDToContainer

Adds a new IED to the container of the IED group.

```
METHOD FINAL AddIEDToContainer : BOOL
VAR_INPUT
    ipIED     : I_AcsiCommonIntelligentElectronicDeviceClass;
END_VAR
VAR_OUTPUT
```

```
    bMatched  : BOOL;
    ipDirNode : I_AcsiIEDContainerClass;
END_VAR
```

ipIED: Interface pointer of type I_AcsiCommonIntelligentElectronicDeviceClass [▶ 368] of the IED device to be added to the container.

bMatched: If TRUE, the object already exists in the container.

ipDirNode: interface pointer of the new container list item.

Return parameter: Positive message (TRUE) on success or negative feedback (FALSE) on error.

6.3.1.10 FB_AcsiCommonIntelligentElectronicDeviceClass

The function block FB_AcsiCommonIntelligentElectronicDeviceClass is one of the basic function blocks of the TwinCAT IEC 61850 data model. Each server instance (IED) generated by the TwinCAT Telecontrol Configurator is derived/extended from FB_AcsiCommonIntelligentElectronicDeviceClass.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonIntelligentElectronicDeviceClass

```
FUNCTION_BLOCK FB_AcsiCommonIntelligentElectronicDeviceClass EXTENDS FB_AcsiCommonNodeClass IMPLEMEN
TS I_AcsiCommonIntelligentElectronicDeviceClass
VAR_OUTPUT
    stInfo : ST_AcsiIntelligentElectronicDeviceClassInfo;
END_VAR
```

Interfaces

Type	Description
I_AcsiCommonIntelligentElectronicDeviceClass [▶ 368]	Interface of the object instance of the top level of the server data structure.

Outputs

Name	Type	Description
stInfo	ST_AcsiIntelligentElectronicDeviceClassInfo [▶ 534]	Status and statistical data of the IED.

Methods

Name	Description
AddLogicalDeviceToContainer [▶ 369]	Adds a new logical device to the server container

Properties

Name	Type	Access	Description
nLogicalDevices	UDINT	Get	Number of logical devices in the server container

6.3.1.11 FB_AcsiCommonLogicalDeviceClass

The function block FB_AcsiCommonLogicalDeviceClass is one of the basic function blocks of the TwinCAT IEC 61850 data model. Each instance of the logical device (LD) generated by the TwinCAT Telecontrol Configurator is derived/extended from FB_AcsiCommonLogicalDeviceClass.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonLogicalDeviceClass

```
FUNCTION_BLOCK FB_AcsiCommonLogicalDeviceClass EXTENDS FB_AcsiCommonNodeClass IMPLEMENTS I_AcsiCommo
nLogicalDeviceClass
VAR_OUTPUT
    stInfo : ST_AcsiLogicalDeviceClassInfo;
END_VAR
```

Interfaces

Type	Description
I_AcsiCommonLogicalDeviceClass [▶ 369]	Interface of the object instance of the logical device.

Outputs

Name	Type	Description
stInfo	ST_AcsiLogicalDeviceClassInfo [▶ 535]	Status and statistics data of the logical device.

Methods

Name	Description
AddLogicalNodeToContainer [▶ 370]	Adds a new logical node

Properties

Name	Type	Access	Description
nLogicalNodes	UDINT	Get	Number of logical node in the container of the logical device

6.3.1.12 FB_AcsiCommonLogicalNodeClass

The function block FB_AcsiCommonLogicalNodeClass is one of the basic function blocks of the TwinCAT IEC 61850 data model. Each instance of the logical node (LN) generated by the TwinCAT Telecontrol Configurator is derived/extended from FB_AcsiCommonLogicalNodeClass.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonLogicalNodeClass

```
FUNCTION_BLOCK FB_AcsiCommonLogicalNodeClass EXTENDS FB_AcsiCommonNodeClass IMPLEMENTS I_AcsiCommonL
ogicalNodeClass
VAR_OUTPUT
    stInfo : ST_AcsiLogicalNodeClassInfo;
END_VAR
```

Interfaces

Type	Description
I_AcsiCommonLogicalNodeClass [▶ 370]	Interface of the object instance of the logical node.

Outputs

Name	Type	Description
stInfo	ST_AcsiLogicalNodeClassInfo [▶ 535]	Status and statistics data of the logical node.

Methods

Name	Description
AddDataToContainer [▶ 371]	Adds a new data object
AddDataSetToContainer [▶ 371]	Adds a new data set
AddUnbufferedReportControlBlockToContainer [▶ 371]	Adds a new report control block instance for unbuffered reports
AddBufferedReportControlBlockToContainer [▶ 372]	Adds a new report control block instance for buffered reports
AddGooseControlBlockToContainer [▶ 372]	Adds a new goose control block instance

Properties

Name	Type	Access	Description
nDataSets	UDINT	Get	Number of data sets in the container of the logical node
nBufferedReportControlBlocks	UDINT	Get	Number of report control block instances for unbuffered reports
nUnbufferedReportControlBlocks	UDINT	Get	Number of report control block instances for buffered reports
nGooseControlBlocks	UDINT	Get	Number of goose control block instances

6.3.1.13 FB_AcsiCommonMulticastSampledValueControlBlockClass

The FB_AcsiCommonMulticastSampledValueControlBlockClass function block is one of the basic function blocks of the TwinCAT IEC 61850 data model. Each SV control block instance generated by the TwinCAT Telecontrol Configurator is derived/extended from FB_AcsiCommonMulticastSampledValueControlBlockClass.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonMulticastSampledValueControlBlockClass

```
FUNCTION_BLOCK FB_AcsiCommonMulticastSampledValueControlBlockClass EXTENDS FB_AcsiCommonControlBlock
Class IMPLEMENTS I_AcsiCommonMulticastSampledValueControlBlockClass
```

Interfaces

Type	Description
I_AcsiCommonMulticastSampledValueControlBlockClass [▶ 372]	Interface of the SV control block instance.

6.3.1.14 FB_AcsiCommonNodeClass

The function block FB_AcsiCommonNodeClass is one of the basic function blocks of the TwinCAT IEC 61850 data model. Each object instance generated by the TwinCAT Telecontrol Configurator is derived/extended from the basic FB_AcsiCommonNodeClass.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass

```
FUNCTION_BLOCK FB_AcsiCommonNodeClass IMPLEMENTS I_AcsiCommonNodeClass
```

Interfaces

Type	Description
I_AcsiCommonNodeClass [▶ 373]	Interface of the basic node instance.

Methods

Name	Description
GetObjectReference [▶ 373]	Returns the reference path of an object instance
GetServerObject [▶ 374]	Provides the interface pointer of the object instance of the server

Properties

Name	Type	Access	Description
sObjectName [▶ 374]	T_AcsiObjectName [▶ 555]	Get,Set	Object name

6.3.1.15 FB_AcsiCommonUnbufferedReportControlBlockClass

The function block FB_AcsiCommonUnbufferedReportControlBlockClass is one of the basic function blocks of the TwinCAT IEC 61850 data model. Each report control block instance for unbuffered reports (unbuffered report control block, URCB) generated by the TwinCAT Telecontrol Configurator is derived/extended from FB_AcsiCommonUnbufferedReportControlBlockClass.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonUnbufferedReportControlBlockClass

```
FUNCTION_BLOCK FB_AcsiCommonUnbufferedReportControlBlockClass EXTENDS FB_AcsiCommonControlBlockClass
 IMPLEMENTS I_AcsiCommonUnbufferedReportControlBlockClass
```

Interfaces

Type	Description
I_AcsiCommonUnbufferedReportControlBlockClass [▶ 374]	Interface of the report control block instance.

Methods

-

Properties

-

6.3.1.16 Switch control

6.3.1.16.1 FB_AcsiCDC_Cancel_AnalogValue

Use: APC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCDC_Cancel_AnalogueValue

```
FUNCTION_BLOCK FB_AcsiCDC_Cancel_AnalogueValue EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.

6.3.1.16.2 FB_AcsiCDC_Cancel_BOOLEAN

Use: SPC, DPC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCDC_Cancel_BOOLEAN

```
FUNCTION_BLOCK FB_AcsiCDC_Cancel_BOOLEAN EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute: Control value (FALSE:= Off, TRUE:= On).
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.

6.3.1.16.3 FB_AcsiCDC_Cancel_CODEDENUM

Use: BSC, BAC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCDC_Cancel_CODEDENUM

```
FUNCTION_BLOCK FB_AcsiCDC_Cancel_CODEDENUM EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiDATypeTcmd [▶ 144]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.

6.3.1.16.4 FB_AcsiCDC_Cancel_ENUMERATED

Use: ENC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCDC_Cancel_ENUMERATED

```
FUNCTION_BLOCK FB_AcsiCDC_Cancel_ENUMERATED EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiCDC_ENUMERATED_ctlVal [▶ 164]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.

6.3.1.16.5 FB_AcsiCDC_Cancel_INT32

Use: INC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCDC_Cancel_INT32

```
FUNCTION_BLOCK FB_AcsiCDC_Cancel_INT32 EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiBATypeINT32 [▶ 106]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.

6.3.1.16.6 FB_AcsiCDC_Cancel_INT8

Use: ISC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCDC_Cancel_INT8

```
FUNCTION_BLOCK FB_AcsiCDC_Cancel_INT8 EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiBATypeINT8 [▶ 104]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.

6.3.1.16.7 FB_AcsiCDC_ENUMERATED_ctlVal

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeENUMERATED [▶ 103] -> FB_AcsiCDC_ENUMERATED_ctlVal

```
FUNCTION_BLOCK FB_AcsiCDC_ENUMERATED_ctlVal EXTENDS FB_AcsiBATypeENUMERATED
```

Properties

-

6.3.1.16.8 FB_AcsiCDC_Oper_AnalogValue

Use: APC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCDC_Oper_AnalogueValue

```
FUNCTION_BLOCK FB_AcsiCDC_Oper_AnalogueValue EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
Check	FB_AcsiDATypeCheck [▶ 114]	Data attribute.

6.3.1.16.9 FB_AcsiCDC_Oper_BOOLEAN

Use: SPC, DPC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCDC_Oper_BOOLEAN

```
FUNCTION_BLOCK FB_AcsiCDC_Oper_BOOLEAN EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute: Control value (FALSE:= Off, TRUE:= On).
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
Check	FB_AcsiDATypeCheck [▶ 114]	Data attribute.

6.3.1.16.10 FB_AcsiCDC_Oper_CODEDENUM

Use: BSC, BAC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCDC_Oper_CODEDENUM

```
FUNCTION_BLOCK FB_AcsiCDC_Oper_CODEDENUM EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiDATypeTcmd [▶ 144]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
Check	FB_AcsiDATypeCheck [▶ 114]	Data attribute.

6.3.1.16.11 FB_AcsiCDC_Oper_ENUMERATED

Use: ENC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCDC_Oper_ENUMERATED

```
FUNCTION_BLOCK FB_AcsiCDC_Oper_ENUMERATED EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiCDC_ENUMERATED_ctlVal [▶ 164]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
Check	FB_AcsiDATypeCheck [▶ 114]	Data attribute.

6.3.1.16.12 FB_AcsiCDC_Oper_INT32

Use: INC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCDC_Oper_INT32

```
FUNCTION_BLOCK FB_AcsiCDC_Oper_INT32 EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiBATypeINT32 [▶ 106]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
Check	FB_AcsiDATypeCheck [▶ 114]	Data attribute.

6.3.1.16.13 FB_AcsiCDC_Oper_INT8

Use: ISC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCDC_Oper_INT8

```
FUNCTION_BLOCK FB_AcsiCDC_Oper_INT8 EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiBATypeINT8 [▶ 104]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
Check	FB_AcsiDATypeCheck [▶ 114]	Data attribute.

6.3.1.16.14 FB_AcsiCDC_SBO

Select before operate.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCommonAttributeClass [▶ 152] -> FB_AcsiBATypeVISIBLESTRING [▶ 108] -> FB_AcsiCDC_SBO

```
FUNCTION_BLOCK FB_AcsiCDC_SBO EXTENDS FB_AcsiBATypeVISIBLESTRING
```

Properties

Name	Type	Access	Description
sValue	STRING(129)	Get,Set	Attribute value.

6.3.1.16.15 FB_AcsiCDC_SBOw_AnalogValue

Use: APC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCDC_SBOw_AnalogueValue

```
FUNCTION_BLOCK FB_AcsiCDC_SBOw_AnalogueValue EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiDATypeAnalogueValue [▶ 110]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
Check	FB_AcsiDATypeCheck [▶ 114]	Data attribute.

6.3.1.16.16 FB_AcsiCDC_SBOw_BOOLEAN

Use: SPC, DPC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCDC_SBOw_BOOLEAN

```
FUNCTION_BLOCK FB_AcsiCDC_SBOw_BOOLEAN EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute: Control value (FALSE:= Off, TRUE:= On).
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute
Check	FB_AcsiDATypeCheck [▶ 114]	Data attribute

6.3.1.16.17 FB_AcsiCDC_SBOw_CODEDENUM

Use: BSC, BAC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCDC_SBOw_CODEDENUM

```
FUNCTION_BLOCK FB_AcsiCDC_SBOw_CODEDENUM EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiDATypeTcmd [▶ 144]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
Check	FB_AcsiDATypeCheck [▶ 114]	Data attribute.

6.3.1.16.18 FB_AcsiCDC_SBOw_ENUMERATED

Use: ENC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCDC_SBOw_ENUMERATED

```
FUNCTION_BLOCK FB_AcsiCDC_SBOw_ENUMERATED EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiCDC_ENUMERATED_ctlVal [▶ 164]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
Check	FB_AcsiDATypeCheck [▶ 114]	Data attribute.

6.3.1.16.19 FB_AcsiCDC_SBOw_INT32

Use: INC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCDC_SBOw_INT32

```
FUNCTION_BLOCK FB_AcsiCDC_SBOw_INT32 EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiBATypeINT32 [▶ 106]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
Check	FB_AcsiDATypeCheck [▶ 114]	Data attribute.

6.3.1.16.20 FB_AcsiCDC_SBOw_INT8

Use: ISC.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -> FB_AcsiCDC_SBOw_INT8

```
FUNCTION_BLOCK FB_AcsiCDC_SBOw_INT8 EXTENDS FB_AcsiCommonDataClass
```

Inputs

Name	Type	Description
ctlVal	FB_AcsiBATypeINT8 [▶ 104]	Data attribute.
operTm	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
origin	FB_AcsiDATypeOriginator [▶ 130]	Data attribute.
ctlNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute.
T	FB_AcsiDATypeTimeStamp [▶ 144]	Data attribute.
Test	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
Check	FB_AcsiDATypeCheck [▶ 114]	Data attribute.

6.3.1.17 Control blocks

6.3.1.17.1 FB_AcsiBufferedReportControlBlock

The function block FB_AcsiBufferedReportControlBlock contains the default data attributes of a report control block for buffered reports.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonBufferedReportControlBlockClass [▶ 152] -
> FB_AcsiBufferedReportControlBlock

```
FUNCTION_BLOCK FB_AcsiBufferedReportControlBlock EXTENDS FB_AcsiCommonBufferedReportControlBlockClas
s
```

Inputs

Name	Type	Description
RptID	FB_AcsiDATypeVisString129 [▶ 151]	Data attribute, Ed1=>RptID:VisStr[65]!
RptEna	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
DatSet	FB_AcsiDATypeDatSetRef [▶ 117]	Data attribute.
ConfRev	FB_AcsiBATypeINT32U [▶ 106]	Data attribute.
OptFlds	FB_AcsiDATypeOptFlds [▶ 128]	Data attribute.
BufTm	FB_AcsiBATypeINT32U [▶ 106]	Data attribute.
SqNum	FB_AcsiBATypeINT16U [▶ 105]	Data attribute.
TrgOps	FB_AcsiDATypeTriggerConditions [▶ 147]	Data attribute.
IntgPd	FB_AcsiBATypeINT32U [▶ 106]	Data attribute.
GI	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
PurgeBuf	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute.
EntryID	FB_AcsiDATypeEntryID [▶ 119]	Data attribute.
TimeOfEntry	FB_AcsiDATypeEntryTime [▶ 119]	Data attribute.
ResvTms	FB_AcsiBATypeINT16 [▶ 105]	Data attribute, Ed2.
Owner	FB_AcsiDATypeOctet64 [▶ 127]	Data attribute, Ed2.

6.3.1.17.2 FB_AcsiGooseControlBlock

The function block FB_AcsiGooseControlBlock contains the standard data attributes of a GOOSE control block.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonGooseControlBlockClass [▶ 155] -
> FB_AcsiGooseControlBlock

```
FUNCTION_BLOCK FB_AcsiGooseControlBlock EXTENDS FB_AcsiCommonGooseControlBlockClass
```

Inputs

Name	Type	Description
GoEna	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute, enables/disables the GOOSE control block.
GoID	FB_AcsiDATypeVisString129 [▶ 151]	Data attribute, the ID used for the GOOSE messages.
DatSet	FB_AcsiDATypeDatSetRef [▶ 117]	Data attribute, determines the data set used for the GOOSE messages.
ConfRev	FB_AcsiBATypeINT32U [▶ 106]	Data attribute, the revision number of the GOOSE control block configuration.
NdsCom	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute, indicates whether the GOOSE control block is not yet fully configured.
DstAddress	FB_AcsiDATypePhyComAddr [▶ 132]	Data attribute, the destination address of the GOOSE messages.
MinTime	FB_AcsiBATypeINT32U [▶ 106]	Data attribute, time between direct send after data change and first retry in milliseconds.
MaxTime	FB_AcsiBATypeINT32U [▶ 106]	Data attribute, maximum time between repeated transmissions of a GOOSE message in milliseconds.
FixedOffs	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute, enables/disables coding with fixed instead of dynamic lengths.

6.3.1.17.3 FB_AcsiMulticastSampledValueControlBlock

The FB_AcsiMulticastSampledValueControlBlock function block contains the standard data attributes of an SV control block.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonMulticastSampledValueControlBlockClass [▶ 159] -> FB_AcsiMulticastSampledValueControlBlock

```
FUNCTION_BLOCK FB_AcsiMulticastSampledValueControlBlock EXTENDS FB_AcsiCommonMulticastSampledValueCo
ntrolBlockClass
```

Inputs

Name	Type	Description
SvEna	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute, enables/disables the SV control block.
MsvID	FB_AcsiDATypeVisString129 [▶ 151]	Data attribute, the ID used for the SV messages.
DatSet	FB_AcsiDATypeDatSetRef [▶ 117]	Data attribute, determines the data set used for the SV messages.
ConfRev	FB_AcsiBATypeINT32U [▶ 106]	Data attribute, the revision number of the SV control block configuration.
SmpRate	FB_AcsiBATypeINT16U [▶ 105]	Data attribute: the sampling rate for SV messages.
OptFlds	FB_AcsiDATypeSvOptFlds [▶ 143]	Data attribute, configures the transmission of the optional parameter values in the SV message.
SmpMod	FB_AcsiDATypeSamplingMode [▶ 139]	Data attribute, sampling mode for SV measured values.
DstAddress	FB_AcsiDATypePhyComAddr [▶ 132]	Data attribute, the destination address of the SV messages.
noASDU	FB_AcsiBATypeINT16U [▶ 105]	Data attribute, the number of ASDUs in an SV message.

6.3.1.17.4 FB_AcsiUnbufferedReportControlBlock

The function block FB_AcsiUnbufferedReportControlBlock contains the default data attributes of a report control block for unbuffered reports.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonUnbufferedReportControlBlockClass [▶ 160] -
> FB_AcsiUnbufferedReportControlBlock

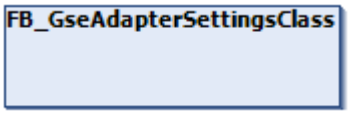
```
FUNCTION_BLOCK FB_AcsiUnbufferedReportControlBlock EXTENDS FB_AcsiCommonUnbufferedReportControlBlock
Class
```

Inputs

Name	Type	Description
RptID	FB_AcsiDATypeVisString129 [▶ 151]	Data attribute, Ed1=>RptID:VisStr[65]!
RptEna	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute
Resv	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute
DatSet	FB_AcsiDATypeDatSetRef [▶ 117]	Data attribute
ConfRev	FB_AcsiBATypeINT32U [▶ 106]	Data attribute
OptFlds	FB_AcsiDATypeOptFlds [▶ 128]	Data attribute
BufTm	FB_AcsiBATypeINT32U [▶ 106]	Data attribute
SqNum	FB_AcsiBATypeINT8U [▶ 105]	Data attribute
TrgOps	FB_AcsiDATypeTriggerConditions [▶ 147]	Data attribute
IntgPd	FB_AcsiBATypeINT32U [▶ 106]	Data attribute
GI	FB_AcsiBATypeBOOLEAN [▶ 103]	Data attribute
Owner	FB_AcsiDATypeOctet64 [▶ 127]	Data attribute, Ed2.

6.3.2 Protocol settings

6.3.2.1 FB_GseAdapterSettingsClass



The function block FB_GseAdapterSettingsClass is used to configure a network adapter that can be used for GSE communication.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

FB_GseAdapterSettingsClass

```
FUNCTION_BLOCK FB_GseAdapterSettingsClass IMPLEMENTS I_GseAdapterSettingsClass
```

Interfaces

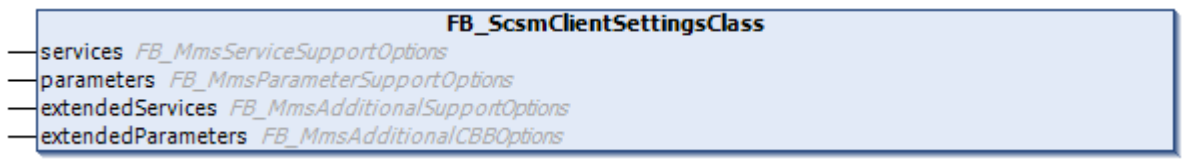
Type	Description
I_GseAdapterSettingsClass [▶ 375]	Interface for configuring the network adapter.

Properties

Name	Type	Access	Definition location	Initial value	Description
bExecGoCBPubs	BOOL	Get	I_GseAdapterSettingsClass [▶ 375]	FALSE	Enables/disables automatic calling of the "Execute" method of the Publisher GOOSE control blocks by the adapter.
		Set	Local		
bExecGoCBSubs	BOOL	Get	I_GseAdapterSettingsClass [▶ 375]	FALSE	Enables/disables automatic calling of the "Execute" method of the Subscriber GOOSE control blocks by the adapter.
		Set	Local		
bSimulation	BOOL	Get	I_GseAdapterSettingsClass [▶ 375]	FALSE	Enables/disables the GOOSE simulation bit field.
		Set	Local		
eAdapterPrio	TcEthernetAdapterPriority [▶ 530]	Get	I_GseAdapterSettingsClass [▶ 375]	TcEthernetAdapterPriority.TcEthernetAdapterPriorityHigh	Priority of the network adapter.
		Set	Local		
eDispatchMode	E_GseDispatchMode [▶ 521]	Get	I_GseAdapterSettingsClass [▶ 375]	E_GseDispatchMode.NonPromiscuous	Setting whether received Ethernet telegrams are forwarded to the operating system.
		Set	Local		
macAddr	ETHERNET_ADDRESS [▶ 528]	Get	I_GseAdapterSettingsClass [▶ 375]	[0, 0, 0, 0, 0, 0] (not set, '00-00-00-00-00-00')	MAC address of the network adapter.
		Set	I_GseAdapterSettingsClass [▶ 375]		

Name	Type	Access	Definition location	Initial value	Description
multicastAddr	ETHERNET_ADDRESS [▶ 528]	Get	I_GseAdapterSettingsClass [▶ 375]	[0, 0, 0, 0, 0, 0] (not set, '00-00-00-00-00-00')	Selected single subscriber multicast address of the network adapter. If set, the network adapter block instance receives only the subscriber messages sent to this exact address. In this case, the address is used as a filter for the subscriber messages. All other messages are forwarded either to the operating system or (if available) to further instances of the network adapter block. **Info:** **From TwinCAT TF6510 IEC 61850 Telecontrol v3.1.98.0 and higher: If this address is not set ('00-00-00-00-00-00'), then all multicast addresses and APPIDs configured on the Subscriber control block instances are used as filters for the Subscriber messages. In this case, different multicast addresses can be configured in the GOOSE control block instances, for example.** The configured Subscriber APPIDs (if used > 0), must match the APPIDs on the Publisher page. All received messages that cannot be assigned to a control block are forwarded to the operating system or to further instances of the network adapter block.
		Set	Local		
oid	OTCID [▶ 529]	Get	I_GseAdapterSettingsClass [▶ 375]	0	TwinCAT object ID of the network adapter. To configure the parameter, see: RT Ethernet adapter configuration [▶ 35].
		Set	Local		
sMacAddr	STRING (17)	Get	I_GseAdapterSettingsClass [▶ 375]	'00-00-00-00-00-00' (not set)	Text representation of the MAC address of the network adapter.
sMulticastAddr	STRING (17)	Get	I_GseAdapterSettingsClass [▶ 375]	'00-00-00-00-00-00' (not set)	Text representation of the multicast address of the network adapter.
		Set	I_GseAdapterSettingsClass [▶ 375]		

6.3.2.2 FB_ScsmClientSettingsClass



The function block FB_ScsmClientSettingsClass is used to configure the SCSM communication layer.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

FB_SocketClientSettingsClass [▶ 236] -> FB_TpktClientSettingsClass [▶ 229] -> FB_Rfc1006ClientSettingsClass [▶ 221] -> FB_UlosiClientSettingsClass [▶ 208] -> FB_MmsClientSerttingsClass [▶ 191] -
> FB_ScsmClientSettingsClass

```
FUNCTION_BLOCK FB_ScsmClientSettingsClass EXTENDS FB_MmsClientSettingsClass IMPLEMENTS I_ScsmClientS
ettingsClass
```

Interfaces

Type	Description
I_ScsmClientSettingsClass [▶ 376]	Interface for configuring the SCSM communication layer

Methods

Name	Description
Defaults [▶ 377]	Resets the configuration settings to the default values.

Inputs

Name	Type	Definition location	Description
services	FB_MmsServiceSupportOptions [▶ 198]	Inherited from FB_MmsClientSettingsClass [▶ 191]	Configuration of the supported MMS protocol services.
parameters	FB_MmsParameterSupportOptions [▶ 199]	-//-	Configuration of the supported MMS protocol parameters.
extendedServices	FB_MmsAdditionalSupportOptions [▶ 199]	-//-	Configuration of the additionally supported MMS protocol services.
extendedParameters	FB_MmsAdditionalCBBOptions [▶ 200]	-//-	Configuration of the supported MMS-CBB protocol parameters.

Properties

Name	Type	Access	Definition location	Initial value	Description
bAutoCleanup	BOOL	Get,Set	Inherited from FB_SocketClientSettingsClass [▶ 236]	TRUE	Enables/disables the automatic clean-up/ closure of the unused sockets (e.g. after program download or PLC reset).
sSrvNetID	T_AmsNetID [▶ 529]	Get,Set	-//-	''	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
nRemotePort	UDINT	Get,Set	-//-	102	TCP/IP port number of the server.
sRemoteHost	T_Ipv4Addr [▶ 529]	Get,Set	-//-	'127.0.0.1'	IPv4 network address of the server.
tConnect	TIME	Get,Set	-//-	T#45s	Earliest time on expiry of which a new connection can be established.
tAdsTimeout	TIME	Get,Set	-//-	DEFAULT_ADS_TIMEOUT	Maximum timeout time for ADS services that may not be exceeded.
tSocketTimeout	TIME	Get,Set	-//-	T#5s	Maximum timeout time during the execution of the confirmed services that may not be exceeded.
bThrottleMode	BOOL	Get,Set	-//-	TRUE	Enables/disables the receive data throttle polling mode.
throttleTimes	T_ThrottleTimes [▶ 530]	Get,Set	-//-	Param_Sockets.cDefaultReceiverThrottleTimes	Configuration of the cycle times for the receive data throttle polling mode.
eTraceSocket	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
ipTLS	I_SocketTlsSettingsClass [▶ 394]	Get,Set	-//-	0	Configuration parameters of the TLS (Transport Layer Security) protocol layer. Value null = not configured.
bEvaLinkStatusErr	BOOL	Get,Set	-//-	FALSE	Configures the evaluation mode for the link status error of the network adapter. If "FALSE" then disabled, short connection interruptions are bridged. If "TRUE" is enabled, the connection is terminated when a connection error is detected.
tTpktTimeout	TIME	Get,Set	Inherited from FB_TpktClientSettingsClass [▶ 229]	T#10s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceTpkt	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.

Name	Type	Access	Definition location	Initial value	Description
sCalling_T_Selector	STRING	Get,Set	Inherited from FB_Rfc1006ClientSettingsClass [▶ 221]	'00 00'	RFC 1006 parameter: Calling transport address selector (source).
sCalled_T_Selector	STRING	Get,Set	-//-	'00 01'	RFC 1006 parameter: Called transport address selector (destination).
eTpduSize	E_Rfc1006TpduSize [▶ 524]	Get,Set	-//-	E_Rfc1006TpduSize.Size_1024	Maximum TPDU byte length.
tRfc1006Timeout	TIME	Get,Set	-//-	T#15s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceRfc1006	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
sCalling_S_Selector	STRING	Get,Set	Inherited from FB_UlosiClientSettingsClass [▶ 208]	'00 01‘	Calling-S-selector.
sCalled_S_Selector	STRING	Get,Set	-//-	'00 01‘	Called-S-selector.
nSessionRequirement	WORD	Get,Set	-//-	2	Session requirements. With the setting 16#FFFF, the value is optional and is not transmitted.
sCalling_P_Selector	STRING	Get,Set	-//-	'00 00 00 01‘	Calling-P-selector.
sCalled_P_Selector	STRING	Get,Set	-//-	'00 00 00 01‘	Called-P-selector.
nPresentRequirements	BYTE	Get,Set	-//-	0	Presentation requirements. With the setting: 0, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
nPresentProtocolVersion	BYTE	Get,Set	-//-	16#FF	Presentation protocol version. With the setting 1, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
nProtocol_Version	BYTE	Get,Set	-//-	16#FF	Association Control Service protocol version. With the setting 1, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
sContext_Name	STRING	Get,Set	-//-	'1.0.9506.2.3'	Application context name (object identifier, dotted presentation), MMS.
sCalled_AP_Title	STRING	Get,Set	-//-	'1.1.1.999.1'	Called-AP-title. With the empty string, the value is optional and is not transmitted.
nCalled_AE_Qualifier	UDINT	Get,Set	-//-	12	Called-AE-qualifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalled_AP_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Called-AP-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.

Name	Type	Access	Definition location	Initial value	Description
nCalled_AE_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Called-AE-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
sCalling_AP_Title	STRING	Get,Set	-//-	'1.1.1.999'	Calling-AP-title. With the empty string, the value is optional and is not transmitted.
nCalling_AE_Qualifier	UDINT	Get,Set	-//-	12	Calling-AE-qualifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalling_AP_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Calling-AP-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalling_AE_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Calling-AE-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nRequirements	BYTE	Get,Set	-//-	0	Association Control Service requirements. The value: 1 enables the authentication function and the value: 0 disables it.
sMechanism_Name	STRING	Get,Set	-//-	'2.2.3.1'	Authentication mechanism name. If there is no empty string and the authentication function is enabled, the value is transmitted.
sAuthent_Value	STRING	Get,Set	-//-	‘‘	Authentication value (password). If there is no empty string and the authentication function is enabled, the value is transmitted.
sResponding_Value	STRING	Get,Set	-//-	‘‘	Responding authentication value (password). If there is no empty string and the authentication function is enabled and the authentication mode is mutual, the value is transmitted.
eAcse_Authent	E_AcseAuthentication [▶ 474]	Get,Set	-//-	E_AcseAuthentication.Unilateral	Authentication mode.
tUlosiTimeout	TIME	Get,Set	-//-	T#20s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceUlosi	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
nLocalDetail	T_MmsInteger32 [▶ 557]	Get,Set	Inherited from FB_MmsClientSettingsClass [▶ 191]	65000	MMS Initiate-RequestPDU-Parameter: localDetailCalling.
nMaxServOutCalling	T_MmsInteger16 [▶ 557]	Get,Set	-//-	10	MMS Initiate-RequestPDU parameter: proposedMaxServOutstandingCalling
nMaxServOutCalled	T_MmsInteger16 [▶ 557]	Get,Set	-//-	10	MMS Initiate-RequestPDU parameter: proposedMaxServOutstandingCalled.

Name	Type	Access	Definition location	Initial value	Description
nNestingLevel	T_MmsInteger8 [▶ 557]	Get,Set	-//-	5	MMS Initiate-RequestPDU parameter: proposedDataStructureNestingLevel.
nMmsVersionNumber	T_MmsInteger16 [▶ 557]	Get,Set	-//-	16#0001	MMS initRequestDetail parameter: proposedVersionNumber.
ipParameters	I_MmsParameterSupportOptions [▶ 380]	Get,Set	-//-	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: proposedParameterCBB.
ipServices	I_MmsServiceSupportOptions [▶ 380]	Get,Set	-//-	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: servicesSupportedCalling.
ipExtendedServices	I_MmsAdditionalSupportOptions [▶ 381]	Get,Set	-//-	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: additionalSupportedCalling.
ipExtendedParameters	I_MmsAdditionalCBBOptions [▶ 381]	Get,Set	-//-	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: additionalCbbSupportedCalling.
sProvilegeClassIdentity	T_MmsVisibleString [▶ 558]	Get,Set	-//-	‘‘	MMS-initRequestDetail parameter: privilegeClassIdentityCalled.
tMmsTimeout	TIME	Get,Set	-//-	T#25s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceMms	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
bAutoInitDataSets	BOOL	Get	I_ScsmClientSettingsClass [▶ 376]	TRUE	Enables/disables the automatic initialization of the DataSets/DataSet member.
		Set	Local		
eScsmEdition	E_ScsmEdition [▶ 525]	Get	I_ScsmClientSettingsClass [▶ 376]	E_ScsmEdition.Ed2	IEC 61850 data model edition (1 or 2)
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
tScsmTimeout	TIME	Get	I_ScsmClientSettingsClass [▶ 376]	T#30s	Maximum time that may not be exceeded during the execution of the confirmed services.
		Set	Local		
eTraceScsm	T_TraceLevel [▶ 526]	Get	I_ScsmClientSettingsClass [▶ 376]	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
		Set	Local		

6.3.2.3 FB_ScsmConnectionSettingsClass



The function block FB_ScsmConnectionSettingsClass is used to configure the SCSM communication layer. The configuration settings only apply to a selected server session function block instance.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

FB_SocketConnectionSettingsClass [▶ 239] -> FB_TpktConnectionSettingsClass [▶ 232] -
> FB_Rfc1006ConnectionSettingsClass [▶ 224] -> FB_UlosiConnectionSettingsClass [▶ 213] -
> FB_MmsConnectionSettingsClass [▶ 200] -> FB_ScsmConnectionSettingsClass

```
FUNCTION_BLOCK FB_ScsmConnectionSettingsClass EXTENDS FB_MmsConnectionSettingsClass IMPLEMENTS I_Scs
mConnectionSettingsClass
```

Interfaces

Type	Description
I_ScsmConnectionSettingsClass [▶ 377]	Interface for configuring the SCSM communication layer.

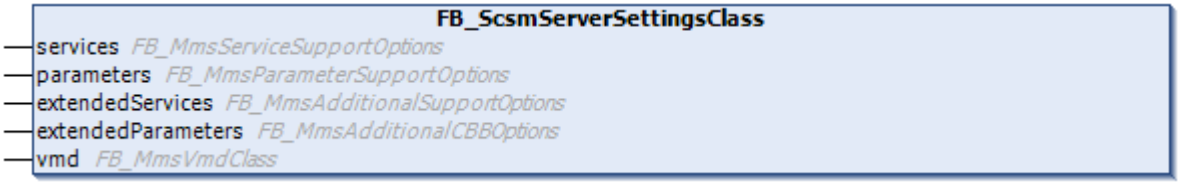
Methods

Name	Description
Defaults [▶ 377]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Definition location	Initial value	Description
bEnable	BOOL	Get,Set	Inherited from FB_SocketConnectionSettingsClass [▶ 239]	FALSE	Enables/disables the establishment of the incoming connection in a server session function block instance.
bTrace	BOOL	Get,Set	-//-	TRUE	Enables/disables the logging of error messages and log messages in a server session function block instance.
eAccept	E_SocketAcceptMode	Get,Set	-//-	E_SocketAcceptMode.eACCEPT_ALL	Configures the acceptance mode for incoming connections. The incoming connections can be accepted or rejected by a session function block instance depending on the network address and/or port number of the client. By default, connections to all clients are accepted.
nAcceptPort	UDINT	Get,Set	-//-	102	Configures the TCP/IP port number of the remote communication partner (clients) to be accepted.
sAcceptHost	T_IPv4Addr [▶ 529]	Get,Set	-//-	'127.0.0.1'	Configures the IPv4 network address of the remote communication partner (clients) to be accepted.
sAuthent_Value	STRING	Get,Set	Inherited from FB_UlosiConnectionSettingsClass [▶ 213]	''	Connection-specific authentication password. Remote communication partner (client) connected to the selected server session function block instance can successfully authenticate with the server using this password.

6.3.2.4 FB_ScsmServerSettingsClass



The function block FB_ScsmServerSettingsClass is used for the configuration of the SCSM communication layer.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

FB_SocketServerSettingsClass [▶ 240] -> FB_TpktServerSettingsClass [▶ 233] -> FB_Rfc1006ServerSettingsClass [▶ 225] -> FB_UlosiServerSettingsClass [▶ 215] -> FB_MmsServerSettingsClass [▶ 201] -> FB_ScsmServerSettingsClass

```
FUNCTION_BLOCK FB_ScsmServerSettingsClass EXTENDS FB_MmsServerSettingsClass IMPLEMENTS I_ScsmServerSettingsClass
```

Interfaces

Type	Description
I_ScsmServerSettingsClass [▶ 378]	Interface for configuring the SCSM communication layer

Methods

Name	Description
Defaults [▶ 378]	Resets the configuration settings to the default values.

Inputs

Name	Type	Definition location	Description
services	FB_MmsServiceSupportOptions [▶ 198]	Inherited from FB_MmsServerSettingsClass [▶ 201]	Configuration of the supported MMS protocol services.
parameters	FB_MmsParameterSupportOptions [▶ 199]	-//-	Configuration of the supported MMS protocol parameters.
extendedServices	FB_MmsAdditionalSupportOptions [▶ 199]	-//-	Configuration of the additionally supported MMS protocol services.
extendedParameters	FB_MmsAdditionalCBBOptions [▶ 200]	-//-	Configuration of the supported MMS-CBB protocol parameters.

Properties

Name	Type	Access	Definition location	Initial value	Description
bAutoCleanup	BOOL	Get,Set	Inherited from FB_SocketServerSettingsClass [▶ 240]	TRUE	Enables/disables the automatic clean-up/ closure of the unused sockets (e.g. after program download or PLC reset).
bEnable	BOOL	Get,Set	-//-	FALSE	Enables/disables establishing the server connections. "TRUE" enables the Listen command (listening and incoming connections are accepted) and "FALSE" disables it.
sSrvNetID	T_AmsNetID [▶ 529]	Get,Set	-//-	''	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
nLocalPort	UDINT	Get,Set	-//-	102	TCP/IP port number of the server.
sLocalHost	T_Ipv4Addr [▶ 529]	Get,Set	-//-	'127.0.0.1'	IPv4 network address of the server.
tAccept	TIME	Get,Set	-//-	T#1s	Retry cycle time for the internal TCP/IP socket Accept commands.
tListen	TIME	Get,Set	-//-	T#1s	Retry cycle time for the internal TCP/IP socket Listen command.
tAdsTimeout	TIME	Get,Set	-//-	DEFAULT_ADS_TIMEOUT	Maximum timeout time for ADS services that may not be exceeded.
tSocketTimeout	TIME	Get,Set	-//-	T#5s	Maximum timeout time during the execution of the confirmed services that may not be exceeded.
bThrottleMode	BOOL	Get,Set	-//-	TRUE	Enables/disables the receive data throttle polling mode.
throttleTimes	T_ThrottleTimes [▶ 530]	Get,Set	-//-	Param_Sockets.cDefaultReceiverThrottleTimes	Configuration of the cycle times for the receive data throttle polling mode.
eTraceSocket	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
ipTLS	I_SocketTlsSettingsClass [▶ 394]	Get,Set	-//-	0	Configuration parameters of the TLS (Transport Layer Security) protocol layer. Value null = not configured.
bEvaLinkStatusErr	BOOL	Get,Set	-//-	FALSE	Configures the evaluation mode for the link status error of the network adapter. If "FALSE" then disabled, short connection interruptions are bridged. If "TRUE" is enabled, the connection is terminated when a connection error is detected.
tTpktTimeout	TIME	Get,Set	Inherited from FB_TpktServerSettingsClass [▶ 233]	T#10s	Maximum time that may not be exceeded during the execution of the confirmed services.

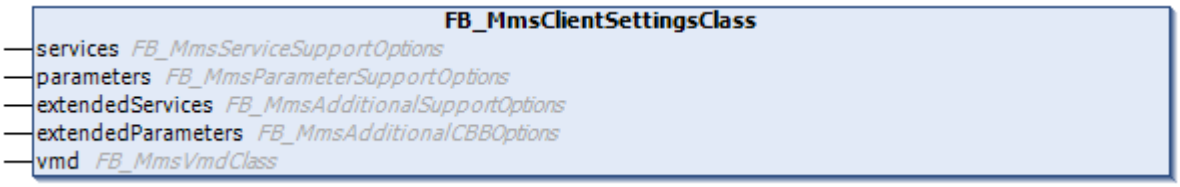
Name	Type	Access	Definition location	Initial value	Description
eTraceTpkt	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
sCalling_T_Selector	STRING	Get,Set	Inherited from FB_Rfc1006ServerSettingsClass [▶ 225]	'00 00'	RFC 1006 parameter: Calling transport address selector (source).
sCalled_T_Selector	STRING	Get,Set	-//-	'00 01'	RFC 1006 parameter: Called transport address selector (destination).
eTpduSize	E_Rfc1006TpduSize [▶ 524]	Get,Set	-//-	E_Rfc1006TpduSize.Size_1024	Maximum TPDU byte length.
tRfc1006Timeout	TIME	Get,Set	-//-	T#15s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceRfc1006	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
sCalling_S_Selector	STRING	Get,Set	Inherited from FB_UlosiServerSettingsClass [▶ 215]	'00 01‘	Calling-S-selector.
sCalled_S_Selector	STRING	Get,Set	-//-	'00 01‘	Called-S-selector.
nSessionRequirement	WORD	Get,Set	-//-	2	Session requirements. With the setting 16#FFFF, the value is optional and is not transmitted.
sCalling_P_Selector	STRING	Get,Set	-//-	'00 00 00 01‘	Calling-P-selector.
sCalled_P_Selector	STRING	Get,Set	-//-	'00 00 00 01‘	Called-P-selector.
nPresentRequirements	BYTE	Get,Set	-//-	0	Presentation requirements. With the setting: 0, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
nPresentProtocolVersion	BYTE	Get,Set	-//-	16#FF	Presentation protocol version. With the setting 1, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
nProtocol_Version	BYTE	Get,Set	-//-	16#FF	Association Control Service protocol version. With the setting 1, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
sContext_Name	STRING	Get,Set	-//-	'1.0.9506.2.3'	Applicaton context name (object identifier, dotted presentation), MMS.
sCalled_AP_Title	STRING	Get,Set	-//-	'1.1.1.999.1'	Called-AP-title. With the empty string, the value is optional and is not transmitted.

Name	Type	Access	Definition location	Initial value	Description
nCalled_AE_Qualifier	UDINT	Get,Set	-//-	12	Called-AE-qualifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalled_AP_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Called-AP-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalled_AE_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Called-AE-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
sCalling_AP_Title	STRING	Get,Set	-//-	'1.1.1.999'	Calling-AP-title. With the empty string, the value is optional and is not transmitted.
nCalling_AE_Qualifier	UDINT	Get,Set	-//-	12	Calling-AE-qualifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalling_AP_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Calling-AP-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalling_AE_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Calling-AE-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nRequirements	BYTE	Get,Set	-//-	0	Association Control Service requirements. The value: 1 enables the authentication function and the value: 0 disables it.
sMechanism_Name	STRING	Get,Set	-//-	'2.2.3.1'	Authentication mechanism name. If there is no empty string and the authentication function is enabled, the value is transmitted.
sAuthent_Value	STRING	Get,Set	-//-	‘‘	Authentication value (password). If there is no empty string and the authentication function is enabled, the value is transmitted.
sResponding_Value	STRING	Get,Set	-//-	‘‘	Responding authentication value (password). If no empty string, the authentication function is enabled and authentication mode: Mutual then the value is transmitted.
eAcse_Authent	E_AcseAuthentication [▶ 474]	Get,Set	-//-	E_AcseAuthentication.Unilateral	Authentication mode.
tUlosiTimeout	TIME	Get,Set	-//-	T#20s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceUlosi	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
nLocalDetail	T_MmsInteger32 [▶ 557]	Get,Set	Inherited from FB_MmsServerSettingsClass [▶ 201]	65000	MMS Initiate-RequestPDU-Parameter: localDetailCalling.

Name	Type	Access	Definition location	Initial value	Description
nMaxServOutCalling	T_MmsInteger16 [▶ 557]	Get,Set	-//-	10	MMS Initiate-RequestPDU parameter: proposedMaxServOutstandingCalling
nMaxServOutCalled	T_MmsInteger16 [▶ 557]	Get,Set	-//-	10	MMS Initiate-RequestPDU parameter: proposedMaxServOutstandingCalled.
nNestingLevel	T_MmsInteger8 [▶ 557]	Get,Set	-//-	5	MMS Initiate-RequestPDU parameter: proposedDataStructureNestingLevel.
nMmsVersionNumber	T_MmsInteger16 [▶ 557]	Get,Set	-//-	16#0001	MMS initRequestDetail parameter: proposedVersionNumber.
ipParameters	I_MmsParameterSupportOptions [▶ 380]	Get,Set	-//-	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: proposedParameterCBB.
ipServices	I_MmsServiceSupportOptions [▶ 380]	Get,Set	-//-	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: servicesSupportedCalling.
ipExtendedServices	I_MmsAdditionalSupportOptions [▶ 381]	Get,Set	-//-	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: additionalSupportedCalling.
ipExtendedParameters	I_MmsAdditionalCBBOptions [▶ 381]	Get,Set	-//-	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: additionalCbbSupportedCalling.
sProvilegeClassIdentity	T_MmsVisibleString [▶ 558]	Get,Set	-//-	‘‘	MMS-initRequestDetail parameter: privilegeClassIdentityCalled.
tMmsTimeout	TIME	Get,Set	-//-	T#25s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceMms	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
bAutoInitDataSets	BOOL	Get	I_ScsmServerSettingsClass [▶ 378]	TRUE	Enables/disables the automatic initialization of the DataSets/DataSet member.
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
eScsmEdition	E_ScsmEdition [▶ 525]	Get	I_ScsmServerSettingsClass [▶ 378]	E_ScsmEdition.Ed2	IEC 61850 data model edition (1 or 2)
		Set	Local		
tScsmTimeout	TIME	Get	I_ScsmServerSettingsClass [▶ 378]	T#30s	Maximum time that may not be exceeded during the execution of the confirmed services.
		Set	Local		
eTraceScsm	T_TraceLevel [▶ 526]	Get	I_ScsmServerSettingsClass [▶ 378]	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
		Set	Local		

6.3.2.5 FB_MmsClientSettingsClass



The function block FB_MmsClientSettingsClass is used for the configuration of the MMS communication layer.

Namespace: Tc3_Mms [▶ 102]
Library: Tc3_Mms (Tc3_Mms.compiled-library)

Inheritance hierarchy

FB_SocketClientSettingsClass [▶ 236] -> FB_TpktClientSettingsClass [▶ 229] -> FB_Rfc1006ClientSettingsClass [▶ 221] -> FB_UlosiClientSettingsClass [▶ 208] -> FB_MmsClientSettingsClass

```
FUNCTION_BLOCK FB_MmsClientSettingsClass EXTENDS FB_UlosiClientSettingsClass IMPLEMENTS I_MmsClientSettingsClass
```

Interfaces

Type	Description
I_MmsClientSettingsClass [▶ 378]	Interface for configuring the MMS communication layer.

Methods

Name	Description
Defaults [▶ 381]	Resets the configuration settings to the default values.

Inputs

Name	Type	Initial value	Description
services	FB_MmsServiceSupportOptions [▶ 198]	.nUsedBits := cMmsServicesSupportOptionsBitSize (* default: 85 *), .bStatus := TRUE, .bGetNameList := TRUE, .bIdentify := TRUE, .bRead := TRUE, .bWrite := TRUE, .bGetVariableAccessAttributes := TRUE, .bDefineNamedVariableList := TRUE, .bGetNamedVariableListAttributes := TRUE, .bDeleteNamedVariableList := TRUE, .bGetDomainAttributes := TRUE, .bInformationReport := TRUE, .bConclude := TRUE, .bCancel := TRUE	Object for the configuration of the supported MMS protocol services.
parameters	FB_MmsParameterSupportOptions [▶ 199]	.nUsedBits := .cMmsParameterSupportOptionsBitSize (* default := 11 *), .bStr1 := TRUE, .bStr2 := TRUE, .bVnam := TRUE, .bValt := TRUE, .bVlis := TRUE	Object for the configuration of the supported MMS protocol parameters.
extendedServices	FB_MmsAdditionalSupportOptions [▶ 199]	.nUsedBits := cMmsAdditionalSupportOptionsBitSize (* default := 0, not used *)	Object for the configuration of the additionally supported MMS protocol services.
extendedParameters	FB_MmsAdditionalCBBOptions [▶ 200]	.nUsedBits := cMmsAdditionalCbbOptionsBitSize (* default := 0, not used *)	Object for the configuration of the additionally supported MMS-CBB protocol parameters.

Properties

Name	Type	Access	Definition location	Initial value	Description
bAutoCleanup	BOOL	Get,Set	Inherited from FB_SocketClientSettingsClass [▶ 236]	TRUE	Enables/disables the automatic clean-up/ closure of the unused sockets (e.g. after program download or PLC reset).
sSrvNetID	T_AmsNetID [▶ 529]	Get,Set	-//-	‘‘	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
nRemotePort	UDINT	Get,Set	-//-	102	TCP/IP port number of the server.
sRemoteHost	T_Ipv4Addr [▶ 529]	Get,Set	-//-	‘127.0.0.1‘	IPv4 network address of the server.
tConnect	TIME	Get,Set	-//-	T#45s	Earliest time on expiry of which a new connection can be established.
tAdsTimeout	TIME	Get,Set	-//-	DEFAULT_ADS_TIMEOUT	Maximum timeout time for ADS services that may not be exceeded.
tSocketTimeout	TIME	Get,Set	-//-	T#5s	Maximum timeout time during the execution of the confirmed services that may not be exceeded.
bThrottleMode	BOOL	Get,Set	-//-	TRUE	Enables/disables the receive data throttle polling mode.
throttleTimes	T_ThrottleTimes [▶ 530]	Get,Set	-//-	Param_Sockets.cDefaultReceiverThrottleTimes	Configuration of the cycle times for the receive data throttle polling mode.
eTraceSocket	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
ipTLS	I_SocketTlsSettingsClass [▶ 394]	Get,Set	-//-	0	Configuration parameters of the TLS (Transport Layer Security) protocol layer. Value null = not configured.
bEvaLinkStatusErr	BOOL	Get,Set	-//-	FALSE	Configures the evaluation mode for the link status error of the network adapter. If "FALSE" then disabled, short connection interruptions are bridged. If "TRUE" is enabled, the connection is terminated when a connection error is detected.
tTpktTimeout	TIME	Get,Set	Inherited from FB_TpktClientSettingsClass [▶ 229]	T#10s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceTpkt	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.

Name	Type	Access	Definition location	Initial value	Description
sCalling_T_Selector	STRING	Get,Set	Inherited from FB_Rfc1006ClientSettingsClass [▶ 221]	'00 00'	RFC 1006 parameter: Calling transport address selector (source).
sCalled_T_Selector	STRING	Get,Set	-//-	'00 01'	RFC 1006 parameter: Called transport address selector (destination).
eTpduSize	E_Rfc1006TpduSize [▶ 524]	Get,Set	-//-	E_Rfc1006TpduSize.Size_1024	Maximum TPDU byte length.
tRfc1006Timeout	TIME	Get,Set	-//-	T#15s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceRfc1006	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
sCalling_S_Selector	STRING	Get,Set	Inherited from FB_UloSiClientSettingsClass [▶ 208]	'00 01‘	Calling-S-selector.
sCalled_S_Selector	STRING	Get,Set	-//-	'00 01‘	Called-S-selector.
nSessionRequirement	WORD	Get,Set	-//-	2	Session requirements. With the setting 16#FFFF, the value is optional and is not transmitted.
sCalling_P_Selector	STRING	Get,Set	-//-	'00 00 00 01‘	Calling-P-selector.
sCalled_P_Selector	STRING	Get,Set	-//-	'00 00 00 01‘	Called-P-selector.
nPresentRequirements	BYTE	Get,Set	-//-	0	Presentation requirements. With the setting: 0, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
nPresentProtocolVersion	BYTE	Get,Set	-//-	16#FF	Presentation protocol version. With the setting 1, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
nProtocol_Version	BYTE	Get,Set	-//-	16#FF	Association Control Service protocol version. With the setting 1, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
sContext_Name	STRING	Get,Set	-//-	'1.0.9506.2.3'	Application context name (object identifier, dotted presentation), MMS.
sCalled_AP_Title	STRING	Get,Set	-//-	'1.1.1.999.1'	Called-AP-title. With the empty string, the value is optional and is not transmitted.
nCalled_AE_Qualifier	UDINT	Get,Set	-//-	12	Called-AE-qualifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalled_AP_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Called-AP-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.

Name	Type	Access	Definition location	Initial value	Description
nCalled_AE_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Called-AE-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
sCalling_AP_Title	STRING	Get,Set	-//-	'1.1.1.999'	Calling-AP-title. With the empty string, the value is optional and is not transmitted.
nCalling_AE_Qualifier	UDINT	Get,Set	-//-	12	Calling-AE-qualifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalling_AP_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Calling-AP-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalling_AE_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Calling-AE-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nRequirements	BYTE	Get,Set	-//-	0	Association Control Service requirements. The value: 1 enables the authentication function and the value: 0 disables it.
sMechanism_Name	STRING	Get,Set	-//-	'2.2.3.1'	Authentication mechanism name. If there is no empty string and the authentication function is enabled, the value is transmitted.
sAuthent_Value	STRING	Get,Set	-//-	‘‘	Authentication value (password). If there is no empty string and the authentication function is enabled, the value is transmitted.
sResponding_Value	STRING	Get,Set	-//-	‘‘	Responding authentication value (password). If no empty string, the authentication function is enabled and authentication mode: Mutual then the value is transmitted.
eAcse_Authent	E_AcseAuthentication [▶ 474]	Get,Set	-//-	E_AcseAuthentication.Unilateral	Authentication mode.
tUlosiTimeout	TIME	Get,Set	-//-	T#20s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceUlosi	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
nLocalDetail	T_MmsInteger32 [▶ 557]	Get	I_MmsClientSettingsClass [▶ 378]	65000	MMS Initiate-RequestPDU-Parameter: localDetailCalling.
		Set	Local		
nMaxServOutCalling	T_MmsInteger16 [▶ 557]	Get	I_MmsClientSettingsClass [▶ 378]	10	MMS Initiate-RequestPDU parameter: proposedMaxServOutstandingCalling
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
nMaxServOutCalled	T_MmsInteger16 [▶ 557]	Get	I_MmsClientSettingsClass [▶ 378]	10	MMS Initiate-RequestPDU parameter: proposedMaxServOutstandingCalled.
		Set	Local		
nNestingLevel	T_MmsInteger8 [▶ 557]	Get	I_MmsClientSettingsClass [▶ 378]	5	MMS Initiate-RequestPDU parameter: proposedDataStructureNestingLevel.
		Set	Local		
nMmsVersionNumber	T_MmsInteger16 [▶ 557]	Get	I_MmsClientSettingsClass [▶ 378]	16#0001	MMS initRequestDetail parameter: proposedVersionNumber.
		Set	Local		
ipParameters	I_MmsParameterSupportOptions [▶ 380]	Get	I_MmsClientSettingsClass [▶ 378]	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: proposedParameterCBB.
		Set	Local		
ipServices	I_MmsServiceSupportOptions [▶ 380]	Get	I_MmsClientSettingsClass [▶ 378]	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: servicesSupportedCalling.
		Set	Local		
ipExtendedServices	I_MmsAdditionalSupportOptions [▶ 381]	Get	I_MmsClientSettingsClass [▶ 378]	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: additionalSupportedCalling.
		Set	Local		
ipExtendedParameters	I_MmsAdditionalCBBOptions [▶ 381]	Get	I_MmsClientSettingsClass [▶ 378]	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: additionalCbbSupportedCalling.
		Set	Local		
sProvilegeClassIdentity	T_MmsVisibleString [▶ 558]	Get	I_MmsClientSettingsClass [▶ 378]	‘‘	MMS-initRequestDetail parameter: privilegeClassIdentityCalled.
		Set	Local		
tMmsTimeout	TIME	Get	I_MmsClientSettingsClass [▶ 378]	T#25s	Maximum time that may not be exceeded during the execution of the confirmed services.
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
eTraceMms	E_TraceLevel [▶ 526]	Get	I_MmsClientSettingsClass [▶ 378]	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
		Set	Local		

6.3.2.5.1 FB_MmsServiceSupportOptions

The function block FB_MmsServiceSupportOptions is used to configure the supported MMS protocol services.

Inheritance hierarchy

FB_MmsServiceSupportoptions

```
FUNCTION_BLOCK FB_MmsServiceSupportOptions IMPLEMENTS I_MmsServiceSupportOptions
```

Interfaces

Type	Description
I_MmsServiceSupportOptions [▶ 380]	Interface for the configuration of the supported MMS protocol services.

Properties

Name	Type	Access	Initial value	Description
bCancel	BOOL	Get,Set	FALSE	
bConclude	BOOL	Get,Set	FALSE	
bDefineNamedType	BOOL	Get,Set	FALSE	
bDefineNamedVariable	BOOL	Get,Set	FALSE	
bDefineNamedVariableList	BOOL	Get,Set	FALSE	
bDefineScatteredAccess	BOOL	Get,Set	FALSE	
bDeleteNamedType	BOOL	Get,Set	FALSE	
bDeleteNamedVariableList	BOOL	Get,Set	FALSE	
bDeleteVariableAccess	BOOL	Get,Set	FALSE	
bGetDomainAttributes	BOOL	Get,Set	FALSE	
bGetNamedTypeAttributes	BOOL	Get,Set	FALSE	
bGetNamedVariableListAttributes	BOOL	Get,Set	FALSE	
bGetNameList	BOOL	Get,Set	FALSE	
bGetScatteredAccessAttributes	BOOL	Get,Set	FALSE	
bGetVariableAccessAttributes	BOOL	Get,Set	FALSE	
bIdentify	BOOL	Get,Set	FALSE	
bInformationReport	BOOL	Get,Set	FALSE	
bRead	BOOL	Get,Set	FALSE	
bRename	BOOL	Get,Set	FALSE	
bStatus	BOOL	Get,Set	FALSE	
bWrite	BOOL	Get,Set	FALSE	

6.3.2.5.2 FB_MmsParameterSupportOptions

The function block FB_MmsParameterSupportOptions is used to configure the supported MMS protocol parameters.

Inheritance hierarchy

FB_MmsParameterSupportOptions

```
FUNCTION_BLOCK FB_MmsParameterSupportOptions IMPLEMENTS I_MmsParameterSupportOptions
```

Interfaces

Type	Description
I_MmsParameterSupportOptions [▶ 380]	Interface for the configuration of the supported MMS protocol parameters.

Properties

Name	Type	Access	Initial value	Description
bCspi	BOOL	Get,Set	FALSE	
bCsr	BOOL	Get,Set	FALSE	
bStr1	BOOL	Get,Set	FALSE	Array support
bStr2	BOOL	Get,Set	FALSE	Structure support
bVadr	BOOL	Get,Set	FALSE	
bValt	BOOL	Get,Set	FALSE	Support for index and index range values (alternate access)
bVlis	BOOL	Get,Set	FALSE	Support for namedVariableList
bVnam	BOOL	Get,Set	FALSE	Support for namedVariables

6.3.2.5.3 FB_MmsAdditionalSupportOptions

The function block FB_MmsAdditionalSupportOptions is used to configure the additionally supported MMS protocol services.

Inheritance hierarchy

FB_MmsAdditionalSupportOptions

```
FUNCTION_BLOCK FB_MmsAdditionalSupportOptions IMPLEMENTS I_MmsAdditionalSupportOptions
```

Interfaces

Type	Description
I_MmsAdditionalSupportOptions [▶ 381]	Interface for the configuration of the additionally supported MMS protocol services.

Properties

Name	Type	Access	Initial value	Description
bVmdReset	BOOL	Get,Set	FALSE	
bVmdStop	BOOL	Get,Set	FALSE	
bSelect	BOOL	Get, Set	FALSE	

6.3.2.5.4 FB_MmsAdditionalCBBOptions

The function block FB_MmsAdditionalCBBOptions is used to configure the additionally supported MMS CBB Protocol parameters.

Inheritance hierarchy

FB_MmsAdditionalCBBOptions

```
FUNCTION_BLOCK FB_MmsAdditionalCBBOptions IMPLEMENTS I_MmsAdditionalCBBOptions
```

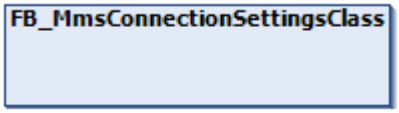
Interfaces

Type	Description
I_MmsAdditionalCBBOptions [▶ 381]	Interface for the configuration of the additionally supported MMS-CBB protocol parameters.

Properties

Name	Type	Access	Initial value	Description
bDei	BOOL	Get,Set	FALSE	
bDes	BOOL	Get,Set	FALSE	
bRed	BOOL	Get, Set	FALSE	

6.3.2.6 FB_MmsConnectionSettingsClass



The function block FB_MmsConnectionSettingsClass is used to configure the MMS communication layer. The configuration settings only apply to a selected server session function block instance.

Namespace: Tc3_Mms [▶ 102]
Library: Tc3_Mms (Tc3_Mms.compiled-library)

Inheritance hierarchy

FB_SocketConnectionSettingsClass [▶ 239] -> FB_TpktConnectionSettingsClass [▶ 232] -
> FB_Rfc1006ConnectionSettingsClass [▶ 224] -> FB_UlosiConnectionSettingsClass [▶ 213] -
> FB_MmsConnectionSettingsClass

```
FUNCTION_BLOCK FB_MmsConnectionSettingsClass EXTENDS FB_UlosiConnectionSettingsClass IMPLEMENTS I_Mm
sConnectionSettingsClass
```

Interfaces

Type	Description
I_MmsConnectionSettingsClass [▶ 382]	Interface for configuring the MMS communication layer.

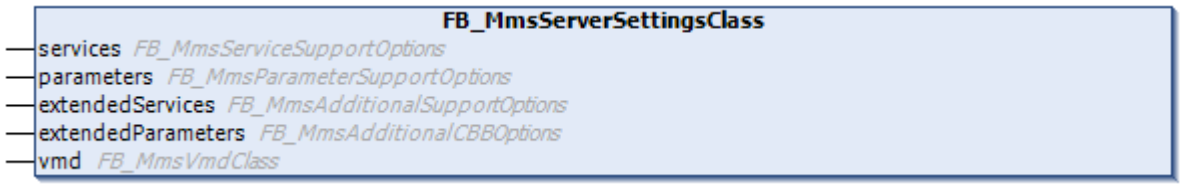
Methods

Name	Description
Defaults [▶ 382]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Definition location	Initial value	Description
bEnable	BOOL	Get,Set	Inherited from FB_SocketConnectionSettingsClass [▶ 239]	FALSE	Enables/disables the establishment of the incoming connection in a server session function block instance.
bTrace	BOOL	Get,Set	-//-	TRUE	Enables/disables the logging of error messages and log messages in a server session function block instance.
eAccept	E_SocketAcceptMode	Get,Set	-//-	E_SocketAcceptMode.eACCEPT_ALL	Configures the acceptance mode for incoming connections. The incoming connections can be accepted or rejected by a session function block instance depending on the network address and/or port number of the client. By default, connections to all clients are accepted.
nAcceptPort	UDINT	Get,Set	-//-	102	Configures the TCP/IP port number of the remote communication partner (clients) to be accepted.
sAcceptHost	T_IPv4Addr [▶ 529]	Get,Set	-//-	'127.0.0.1'	Configures the IPv4 network address of the remote communication partner (clients) to be accepted.
sAuthent_Value	STRING	Get,Set	Inherited from FB_UlosiConnectionSettingsClass [▶ 213]	''	Connection-specific authentication password. Remote communication partner (client) connected to the selected server session function block instance can successfully authenticate with the server using this password.

6.3.2.7 FB_MmsServerSettingsClass



The function block FB_MmsServerSettingsClass is used for the configuration of the MMS communication layer.

Namespace: Tc3_Mms [▶ 102]
Library: Tc3_Mms (Tc3_Mms.compiled-library)

Inheritance hierarchy

FB_SocketServerSettingsClass [▶ 240] -> FB_TpktServerSettingsClass [▶ 233] -> FB_Rfc1006ServerSettingsClass [▶ 225] -> FB_UlosiServerSettingsClass [▶ 215] -> FB_MmsServerSettingsClass

```
FUNCTION_BLOCK FB_MmsServerSettingsClass EXTENDS FB_UlosiServerSettingsClass IMPLEMENTS I_MmsServerSettingsClass
```

Interfaces

Type	Description
I_MmsServerSettingsClass [▶ 382]	Interface for configuring the MMS communication layer.

Methods

Name	Description
Defaults [▶ 383]	Resets the configuration settings to the default values.

Inputs

Name	Type	Initial value	Description
services	FB_MmsServiceSupportOptions [▶ 198]	.nUsedBits := cMmsServicesSupportOptionsBitSize (* default: 85 *), .bStatus := TRUE, .bGetNameList := TRUE, .bIdentify := TRUE, .bRead := TRUE, .bWrite := TRUE, .bGetVariableAccessAttributes := TRUE, .bDefineNamedVariableList := TRUE, .bGetNamedVariableListAttributes := TRUE, .bDeleteNamedVariableList := TRUE, .bGetDomainAttributes := TRUE, .bInformationReport := TRUE, .bConclude := TRUE, .bCancel := TRUE	Object for the configuration of the supported MMS protocol services.
parameters	FB_MmsParameterSupportOptions [▶ 199]	.nUsedBits := .cMmsParameterSupportOptionsBitSize (* default := 11 *), .bStr1 := TRUE, .bStr2 := TRUE, .bVnam := TRUE, .bValt := TRUE, .bVlis := TRUE	Object for the configuration of the supported MMS protocol parameters.
extendedServices	FB_MmsAdditionalSupportOptions [▶ 199]	.nUsedBits := cMmsAdditionalSupportOptionsBitSize (* default := 0, not used *)	Object for the configuration of the additionally supported MMS protocol services.
extendedParameters	FB_MmsAdditionalCBBOptions [▶ 200]	.nUsedBits := cMmsAdditionalCbbOptionsBitSize (* default := 0, not used *)	Object for the configuration of the additionally supported MMS-CBB protocol parameters.

Properties

Name	Type	Access	Definition location	Initial value	Description
bAutoCleanup	BOOL	Get,Set	Inherited from FB_SocketServerSettingsClass [▶ 240]	TRUE	Enables/disables the automatic clean-up/ closure of the unused sockets (e.g. after program download or PLC reset).
bEnable	BOOL	Get,Set	-//-	FALSE	Enables/disables establishing the server connections. "TRUE" enables the Listen command (listening and incoming connections are accepted) and "FALSE" disables it.
sSrvNetID	T_AmsNetID [▶ 529]	Get,Set	-//-	''	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
nLocalPort	UDINT	Get,Set	-//-	102	TCP/IP port number of the server.
sLocalHost	T_Ipv4Addr [▶ 529]	Get,Set	-//-	'127.0.0.1'	IPv4 network address of the server.
tAccept	TIME	Get,Set	-//-	T#1s	Retry cycle time for the internal TCP/IP socket Accept commands.
tListen	TIME	Get,Set	-//-	T#1s	Retry cycle time for the internal TCP/IP socket Listen command.
tAdsTimeout	TIME	Get,Set	-//-	DEFAULT_ADS_TIMEOUT	Maximum timeout time for ADS services that may not be exceeded.
tSocketTimeout	TIME	Get,Set	-//-	T#5s	Maximum timeout time during the execution of the confirmed services that may not be exceeded.
bThrottleMode	BOOL	Get,Set	-//-	TRUE	Enables/disables the receive data throttle polling mode.
throttleTimes	T_ThrottleTimes [▶ 530]	Get,Set	-//-	Param_Sockets.cDefaultReceiverThrottleTimes	Configuration of the cycle times for the receive data throttle polling mode.
eTraceSocket	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
ipTLS	I_SocketTlsSettingsClass [▶ 394]	Get,Set	-//-	0	Configuration parameters of the TLS (Transport Layer Security) protocol layer. Value null = not configured.
bEvaLinkStatusErr	BOOL	Get,Set	-//-	FALSE	Configures the evaluation mode for the link status error of the network adapter. If "FALSE" then disabled, short connection interruptions are bridged. If "TRUE" is enabled, the connection is terminated when a connection error is detected.
tTpktTimeout	TIME	Get,Set	Inherited from FB_TpktServerSettingsClass [▶ 233]	T#10s	Maximum time that may not be exceeded during the execution of the confirmed services.

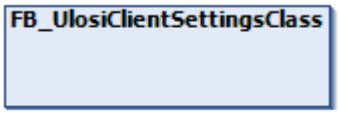
Name	Type	Access	Definition location	Initial value	Description
eTraceTpkt	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
bEvaLinkStatusErr	BOOL				
sCalling_T_Selector	STRING	Get,Set	Inherited from FB_Rfc1006ServerSettingsClass [▶ 225]	'00 00'	RFC 1006 parameter: Calling transport address selector (source).
sCalled_T_Selector	STRING	Get,Set	-//-	'00 01'	RFC 1006 parameter: Called transport address selector (destination).
eTpduSize	E_Rfc1006TpduSize [▶ 524]	Get,Set	-//-	E_Rfc1006TpduSize.Size_1024	Maximum TPDU byte length.
tRfc1006Timeout	TIME	Get,Set	-//-	T#15s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceRfc1006	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
sCalling_S_Selector	STRING	Get,Set	Inherited from FB_UlosiServerSettingsClass [▶ 215]	'00 01'	Calling-S-selector.
sCalled_S_Selector	STRING	Get,Set	-//-	'00 01'	Called-S-selector.
nSessionRequirement	WORD	Get,Set	-//-	2	Session requirements. With the setting 16#FFFF, the value is optional and is not transmitted.
sCalling_P_Selector	STRING	Get,Set	-//-	'00 00 00 01'	Calling-P-selector.
sCalled_P_Selector	STRING	Get,Set	-//-	'00 00 00 01'	Called-P-selector.
nPresentRequirements	BYTE	Get,Set	-//-	0	Presentation requirements. With the setting: 0, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
nPresentProtocolVersion	BYTE	Get,Set	-//-	16#FF	Presentation protocol version. With the setting 1, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
nProtocol_Version	BYTE	Get,Set	-//-	16#FF	Association Control Service protocol version. With the setting 1, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
sContext_Name	STRING	Get,Set	-//-	'1.0.9506.2.3'	Application context name (object identifier, dotted presentation), MMS.
sCalled_AP_Title	STRING	Get,Set	-//-	'1.1.1.999.1'	Called-AP-title. With the empty string, the value is optional and is not transmitted.

Name	Type	Access	Definition location	Initial value	Description
nCalled_AE_Qualifier	UDINT	Get,Set	-//-	12	Called-AE-qualifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalled_AP_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Called-AP-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalled_AE_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Called-AE-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
sCalling_AP_Title	STRING	Get,Set	-//-	'1.1.1.999'	Calling-AP-title. With the empty string, the value is optional and is not transmitted.
nCalling_AE_Qualifier	UDINT	Get,Set	-//-	12	Calling-AE-qualifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalling_AP_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Calling-AP-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nCalling_AE_InvID	UDINT	Get,Set	-//-	16#FFFFFFFF	Calling-AE-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
nRequirements	BYTE	Get,Set	-//-	0	Association Control Service requirements. The value: 1 enables the authentication function and the value: 0 disables it.
sMechanism_Name	STRING	Get,Set	-//-	'2.2.3.1'	Authentication mechanism name. If there is no empty string and the authentication function is enabled, the value is transmitted.
sAuthent_Value	STRING	Get,Set	-//-	‘‘	Authentication value (password). If there is no empty string and the authentication function is enabled, the value is transmitted.
sResponding_Value	STRING	Get,Set	-//-	‘‘	Responding authentication value (password). If no empty string, the authentication function is enabled and authentication mode: Mutual then the value is transmitted.
eAcse_Authent	E_AcseAuthentication [▶ 474]	Get,Set	-//-	E_AcseAuthentication.Unilateral	Authentication mode.
tUlosiTimeout	TIME	Get,Set	-//-	T#20s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceUlosi	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
nLocalDetail	T_MmsInteger32 [▶ 557]	Get	I_MmsServerSettingsClass [▶ 382]	65000	MMS Initiate-RequestPDU-Parameter: localDetailCalling.
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
nMaxServOutCalling	T_MmsInteger16 [▶ 557]	Get	I_MmsServerSettings Class [▶ 382]	10	MMS Initiate-RequestPDU parameter: proposedMaxServOutstandingCalling
		Set	Local		
nMaxServOutCalled	T_MmsInteger16 [▶ 557]	Get	I_MmsServerSettings Class [▶ 382]	10	MMS Initiate-RequestPDU parameter: proposedMaxServOutstandingCalled.
		Set	Local		
nNestingLevel	T_MmsInteger8 [▶ 557]	Get	I_MmsServerSettings Class [▶ 382]	5	MMS Initiate-RequestPDU parameter: proposedDataStructureNestingLevel.
		Set	Local		
nMmsVersionNumber	T_MmsInteger16 [▶ 557]	Get	I_MmsServerSettings Class [▶ 382]	16#0001	MMS initRequestDetail parameter: proposedVersionNumber.
		Set	Local		
ipParameters	I_MmsParameterSupportOptions [▶ 380]	Get	I_MmsServerSettings Class [▶ 382]	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: proposedParameterCBB.
		Set	Local		
ipServices	I_MmsServiceSupportOptions [▶ 380]	Get	I_MmsServerSettings Class [▶ 382]	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: servicesSupportedCalling.
		Set	Local		
ipExtendedServices	I_MmsAdditionalSupportOptions [▶ 381]	Get	I_MmsServerSettings Class [▶ 382]	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: additionalSupportedCalling.
		Set	Local		
ipExtendedParameters	I_MmsAdditionalCBBOptions [▶ 381]	Get	I_MmsServerSettings Class [▶ 382]	-	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: additionalCbbSupportedCalling.
		Set	Local		
sProvilegeClassIdentity	T_MmsVisibleString [▶ 558]	Get	I_MmsServerSettings Class [▶ 382]	‘‘	MMS-initRequestDetail parameter: privilegeClassIdentityCalled.
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
tMmsTimeout	TIME	Get	I_MmsServerSettingsClass [▶ 382]	T#25s	Maximum time that may not be exceeded during the execution of the confirmed services.
		Set	Local		
eTraceMms	E_TraceLevel [▶ 526]	Get	I_MmsServerSettingsClass [▶ 382]	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
		Set	Local		

6.3.2.8 FB_UlosiClientSettingsClass



The function block FB_UlosiClientSettingsClass is used to configure the ULOSI (Upper Layer OSI) communication layer.

Namespace: Tc3_Ulosi [▶ 102]
Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Inheritance hierarchy

FB_SocketClientSettingsClass [▶ 236] -> FB_TpktClientSettingsClass [▶ 229] -> FB_Rfc1006ClientSettingsClass [▶ 221] -> FB_UlosiClientSettingsClass

```
FUNCTION_BLOCK FB_UlosiClientSettingsClass EXTENDS FB_Rfc1006ClientSettingsClass IMPLEMENTS I_UlosiClientSettingsClass
```

Interfaces

Type	Description
I_UlosiClientSettingsClass [▶ 384]	Interface for configuring the ULOSI communication layer.

Methods

Name	Description
Defaults [▶ 385]	Resets the configuration settings to the default values.

Properties

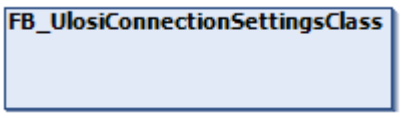
Name	Type	Access	Definition location	Initial value	Description
bAutoCleanup	BOOL	Get,Set	Inherited from FB_SocketClientSettingsClass [▶ 236]	TRUE	Enables/disables the automatic clean-up/ closure of the unused sockets (e.g. after program download or PLC reset).
sSrvNetID	T_AmsNetID [▶ 529]	Get,Set	-//-	‘‘	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
nRemotePort	UDINT	Get,Set	-//-	102	TCP/IP port number of the server.
sRemoteHost	T_Ipv4Addr [▶ 529]	Get,Set	-//-	‘127.0.0.1‘	IPv4 network address of the server.
tConnect	TIME	Get,Set	-//-	T#45s	Earliest time on expiry of which a new connection can be established.
tAdsTimeout	TIME	Get,Set	-//-	DEFAULT_ADS_TIMEOUT	Maximum timeout time for ADS services that may not be exceeded.
tSocketTimeout	TIME	Get,Set	-//-	T#5s	Maximum timeout time during the execution of the confirmed services that may not be exceeded.
bThrottleMode	BOOL	Get,Set	-//-	TRUE	Enables/disables the receive data throttle polling mode.
throttleTimes	T_ThrottleTimes [▶ 530]	Get,Set	-//-	Param_Sockets.cDefaultReceiverThrottleTimes	Configuration of the cycle times for the receive data throttle polling mode.
eTraceSocket	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
ipTLS	I_SocketTlsSettingsClass	Get,Set	-//-	0	Configuration parameters of the TLS (Transport Layer Security) protocol layer. Value null = not configured.
bEvaLinkStatusErr	BOOL	Get,Set	-//-	FALSE	Configures the evaluation mode for the link status error of the network adapter. If "FALSE" then disabled, short connection interruptions are bridged. If "TRUE" is enabled, the connection is terminated when a connection error is detected.
tTpktTimeout	TIME	Get,Set	Inherited from FB_TpktClientSettingsClass [▶ 229]	T#10s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceTpkt	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
sCalling_T_Selector	STRING	Get,Set	Inherited from FB_Rfc1006ClientSettingsClass [▶ 221]	’00 00’	RFC 1006 parameter: Calling transport address selector (source).

Name	Type	Access	Definition location	Initial value	Description
sCalled_T_Selector	STRING	Get,Set	-//-	'00 01'	RFC 1006 parameter: Called transport address selector (destination).
eTpduSize	E_Rfc1006TpduSize [▶ 524]	Get,Set	-//-	E_Rfc1006TpduSize.Size_1024	Maximum TPDU byte length.
tRfc1006Timeout	TIME	Get,Set	-//-	T#15s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceRfc1006	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
sCalling_S_Selector	STRING	Get	I_UlosiClientSettingsClass [▶ 384]	'00 01'	Calling-S-selector.
		Set	Local		
sCalled_S_Selector	STRING	Get	I_UlosiClientSettingsClass [▶ 384]	'00 01'	Called-S-selector.
		Set	Local		
nSessionRequirement	WORD	Get	I_UlosiClientSettingsClass [▶ 384]	2	Session requirements. With the setting 16#FFFF, the value is optional and is not transmitted.
		Set	Local		
sCalling_P_Selector	STRING	Get	I_UlosiClientSettingsClass [▶ 384]	'00 00 00 01'	Calling-P-selector.
		Set	Local		
sCalled_P_Selector	STRING	Get	I_UlosiClientSettingsClass [▶ 384]	'00 00 00 01'	Called-P-selector.
		Set	Local		
nPresentRequirements	BYTE	Get	I_UlosiClientSettingsClass [▶ 384]	0	Presentation requirements. With the setting: 0, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
		Set	Local		
nPresentProtocolVersion	BYTE	Get	I_UlosiClientSettingsClass [▶ 384]	16#FF	Presentation protocol version. With the setting 1, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
nProtocol_Version	BYTE	Get	I_UlosiClientSettings Class [▶ 384]	16#FF	Association Control Service protocol version. With the setting 1, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
		Set	Local		
sContext_Name	STRING	Get	I_UlosiClientSettings Class [▶ 384]	'1.0.9506.2.3'	Application context name (object identifier, dotted presentation), MMS.
		Set	Local		
sCalled_AP_Title	STRING	Get	I_UlosiClientSettings Class [▶ 384]	'1.1.1.999.1'	Called-AP-title. With the empty string, the value is optional and is not transmitted.
		Set	Local		
nCalled_AE_Qualifier	UDINT	Get	I_UlosiClientSettings Class [▶ 384]	12	Called-AE-qualifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
		Set	Local		
nCalled_AP_InvID	UDINT	Get	I_UlosiClientSettings Class [▶ 384]	16#FFFFFFFF	Called-AP-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
		Set	Local		
nCalled_AE_InvID	UDINT	Get	I_UlosiClientSettings Class [▶ 384]	16#FFFFFFFF	Called-AE-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
		Set	Local		
sCalling_AP_Title	STRING	Get	I_UlosiClientSettings Class [▶ 384]	'1.1.1.999'	Calling-AP-title. With the empty string, the value is optional and is not transmitted.
		Set	Local		
nCalling_AE_Qualifier	UDINT	Get	I_UlosiClientSettings Class [▶ 384]	12	Calling-AE-qualifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
		Set	Local		
nCalling_AP_InvID	UDINT	Get	I_UlosiClientSettings Class [▶ 384]	16#FFFFFFFF	Calling-AP-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
nCalling_AE_InvID	UDINT	Get	I_UlosiClientSettingsClass [▶ 384]	16#FFFFFFFF	Calling-AE-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
		Set	Local		
nRequirements	BYTE	Get	I_UlosiClientSettingsClass [▶ 384]	0	Association Control Service requirements. The value: 1 enables the authentication function and the value: 0 disables it.
		Set	Local		
sMechanism_Name	STRING	Get	I_UlosiClientSettingsClass [▶ 384]	'2.2.3.1'	Authentication mechanism name. If there is no empty string and the authentication function is enabled, the value is transmitted.
		Set	Local		
sAuthent_Value	STRING	Get	I_UlosiClientSettingsClass [▶ 384]	‘‘	Authentication value (password). If there is no empty string and the authentication function is enabled, the value is transmitted.
		Set	Local		
sResponding_Value	STRING	Get	I_UlosiClientSettingsClass [▶ 384]	‘‘	Responding authentication value (password). If no empty string, the authentication function is enabled and authentication mode: Mutual then the value is transmitted.
		Set	Local		
eAcse_Authent	E_AcseAuthentication [▶ 474]	Get	I_UlosiClientSettingsClass [▶ 384]	E_AcseAuthentication.Unilateral	Authentication mode.
		Set	Local		
tUlosiTimeout	TIME	Get	I_UlosiClientSettingsClass [▶ 384]	T#20s	Maximum time that may not be exceeded during the execution of the confirmed services.
		Set	Local		
eTraceUlosi	E_TraceLevel [▶ 526]	Get	I_UlosiClientSettingsClass [▶ 384]	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
		Set	Local		

6.3.2.9 FB_UlosiConnectionSettingsClass



The function block FB_UlosiConnectionSettingsClass is used to configure the ULOSI communication layer (Upper Layer OSI). The configuration settings only apply to a selected server session function block instance.

Namespace: Tc3_Ulosi [▶ 102]
Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Inheritance hierarchy

FB_SocketConnectionSettingsClass [▶ 239] -> FB_TpktConnectionSettingsClass [▶ 232] -
> FB_Rfc1006ConnectionSettingsClass [▶ 224] -> FB_UlosiConnectionSettingsClass

```
FUNCTION_BLOCK FB_UlosiConnectionSettingsClass EXTENDS FB_Rfc1006ConnectionSettingsClass IMPLEMENTS
I_UlosiConnectionSettingsClass
```

Interfaces

Type	Description
I_UlosiConnectionSettingsClass [▶ 386]	Interface for configuring the ULOSI communication layer.

Methods

Name	Description
Defaults [▶ 386]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Definition location	Initial value	Description
bEnable	BOOL	Get,Set	Inherited from FB_SocketConnectionSettingsClass [▶ 239]	FALSE	Enables/disables the establishment of the incoming connection in a server session function block instance.
bTrace	BOOL	Get,Set	-//-	TRUE	Enables/disables the logging of error messages and log messages in a server session function block instance.
eAccept	E_SocketAcceptMode	Get,Set	-//-	E_SocketAcceptMode.eACCEPT_ALL	Configures the acceptance mode for incoming connections. The incoming connections can be accepted or rejected by a session function block instance depending on the network address and/or port number of the client. By default, connections to all clients are accepted.
nAcceptPort	UDINT	Get,Set	-//-	102	Configures the TCP/IP port number of the remote communication partner (clients) to be accepted.
sAcceptHost	T_IPv4Addr [▶ 529]	Get,Set	-//-	'127.0.0.1'	Configures the IPv4 network address of the remote communication partner (clients) to be accepted.
sAuthent_Value	STRING	Get	I_UlosiConnectionSettingsClass [▶ 386]	''	Connection-specific authentication password. Remote communication partner (client) connected to the selected server session function block instance can successfully authenticate with the server using this password.
		Set	Local		

6.3.2.10 FB_UlosiServerSettingsClass



The function block FB_UlosiServerSettingsClass is used to configure the ULOSI (Upper Layer OSI) communication layer.

Namespace: Tc3_Ulosi [▶ 102]
Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Inheritance hierarchy

FB_SocketServerSettingsClass [▶ 240] -> FB_TpktServerSettingsClass [▶ 233] -> FB_Rfc1006ServerSettingsClass [▶ 225] -> FB_UlosiServerSettingsClass

```
FUNCTION_BLOCK FB_UlosiServerSettingsClass EXTENDS FB_Rfc1006ServerSettingsClass IMPLEMENTS I_UlosiS
erverSettingsClass
```

Interfaces

Type	Description
I_UlosiServerSettingsClass [▶ 386]	Interface for configuring the ULOSI communication layer.

Methods

Name	Description
Defaults [▶ 387]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Definition location	Initial value	Description
bAutoCleanup	BOOL	Get,Set	Inherited from FB_SocketServerSettingsClass [▶ 240]	TRUE	Enables/disables the automatic clean-up/ closure of the unused sockets (e.g. after program download or PLC reset).
bEnable	BOOL	Get,Set	-//-	FALSE	Enables/disables establishing the server connections. "TRUE" enables the Listen command (listening and incoming connections are accepted) and "FALSE" disables it.
sSrvNetID	T_AmsNetID [▶ 529]	Get,Set	-//-	''	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
nLocalPort	UDINT	Get,Set	-//-	102	TCP/IP port number of the server.
sLocalHost	T_Ipv4Addr [▶ 529]	Get,Set	-//-	'127.0.0.1'	IPv4 network address of the server.
tAccept	TIME	Get,Set	-//-	T#1s	Retry cycle time for the internal TCP/IP socket Accept commands.
tListen	TIME	Get,Set	-//-	T#1s	Retry cycle time for the internal TCP/IP socket Listen command.
tAdsTimeout	TIME	Get,Set	-//-	DEFAULT_ADS_TIMEOUT	Maximum timeout time for ADS services that may not be exceeded.
tSocketTimeout	TIME	Get,Set	-//-	T#5s	Maximum timeout time during the execution of the confirmed services that may not be exceeded.
bThrottleMode	BOOL	Get,Set	-//-	TRUE	Enables/disables the receive data throttle polling mode.
throttleTimes	T_ThrottleTimes [▶ 530]	Get,Set	-//-	Param_Sockets.cDefaultReceiverThrottleTimes	Configuration of the cycle times for the receive data throttle polling mode.
eTraceSocket	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
ipTLS	I_SocketTlsSettingsClass [▶ 394]	Get,Set	-//-	0	Configuration parameters of the TLS (Transport Layer Security) protocol layer. Value null = not configured.
bEvaLinkStatusErr	BOOL	Get,Set	-//-	FALSE	Configures the evaluation mode for the link status error of the network adapter. If "FALSE" then disabled, short connection interruptions are bridged. If "TRUE" is enabled, the connection is terminated when a connection error is detected.
tTpktTimeout	TIME	Get,Set	Inherited from FB_TpktServerSettingsClass [▶ 233]	T#10s	Maximum time that may not be exceeded during the execution of the confirmed services.

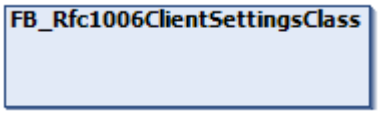
Name	Type	Access	Definition location	Initial value	Description
eTraceTpkt	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
sCalling_T_Selector	STRING	Get,Set	Inherited from FB_Rfc1006ServerSettingsClass [▶ 225]	'00 00'	RFC 1006 parameter: Calling transport address selector (source).
sCalled_T_Selector	STRING	Get,Set	-//-	'00 01'	RFC 1006 parameter: Called transport address selector (destination).
eTpduSize	E_Rfc1006TpduSize [▶ 524]	Get,Set	-//-	E_Rfc1006TpduSize.Size_1024	Maximum TPDU byte length.
tRfc1006Timeout	TIME	Get,Set	-//-	T#15s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceRfc1006	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
sCalling_S_Selector	STRING	Get	I_UlosiServerSettingsClass [▶ 386]	'00 01'	Calling-S-selector.
		Set	Local		
sCalled_S_Selector	STRING	Get	I_UlosiServerSettingsClass [▶ 386]	'00 01'	Called-S-selector.
		Set	Local		
nSessionRequirement	WORD	Get	I_UlosiServerSettingsClass [▶ 386]	2	Session requirements. With the setting 16#FFFF, the value is optional and is not transmitted.
		Set	Local		
sCalling_P_Selector	STRING	Get	I_UlosiServerSettingsClass [▶ 386]	'00 00 00 01'	Calling-P-selector.
		Set	Local		
sCalled_P_Selector	STRING	Get	I_UlosiServerSettingsClass [▶ 386]	'00 00 00 01'	Called-P-selector.
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
nPresentRequirements	BYTE	Get	I_UlosiServerSettings Class [▶ 386]	0	Presentation requirements. With the setting: 0, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
		Set	Local		
nPresentProtocolVersion	BYTE	Get	I_UlosiServerSettings Class [▶ 386]	16#FF	Presentation protocol version. With the setting 1, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
		Set	Local		
nProtocol_Version	BYTE	Get	I_UlosiServerSettings Class [▶ 386]	16#FF	Association Control Service protocol version. With the setting 1, the value is transmitted and with the setting 16#FF, the value is optional and is not transmitted.
		Set	Local		
sContext_Name	STRING	Get	I_UlosiServerSettings Class [▶ 386]	'1.0.9506.2.3'	Application context name (object identifier, dotted presentation), MMS.
		Set	Local		
sCalled_AP_Title	STRING	Get	I_UlosiServerSettings Class [▶ 386]	'1.1.1.999.1'	Called-AP-title. With the empty string, the value is optional and is not transmitted.
		Set	Local		
nCalled_AE_Qualifier	UDINT	Get	I_UlosiServerSettings Class [▶ 386]	12	Called-AE-qualifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
		Set	Local		
nCalled_AP_InvID	UDINT	Get	I_UlosiServerSettings Class [▶ 386]	16#FFFFFFFF	Called-AP-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
		Set	Local		
nCalled_AE_InvID	UDINT	Get	I_UlosiServerSettings Class [▶ 386]	16#FFFFFFFF	Called-AE-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
		Set	Local		
sCalling_AP_Title	STRING	Get	I_UlosiServerSettings Class [▶ 386]	'1.1.1.999'	Calling-AP-title. With the empty string, the value is optional and is not transmitted.
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
nCalling_AE_Qualifier	UDINT	Get	I_UlosiServerSettings Class [▶ 386]	12	Calling-AE-qualifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
		Set	Local		
nCalling_AP_InvID	UDINT	Get	I_UlosiServerSettings Class [▶ 386]	16#FFFFFFFF	Calling-AP-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
		Set	Local		
nCalling_AE_InvID	UDINT	Get	I_UlosiServerSettings Class [▶ 386]	16#FFFFFFFF	Calling-AE-invocation-identifier. With the setting 16#FFFFFFFFF, the value is optional and is not transmitted.
		Set	Local		
nRequirements	BYTE	Get	I_UlosiServerSettings Class [▶ 386]	0	Association Control Service requirements. The value: 1 enables the authentication function and the value: 0 disables it.
		Set	Local		
sMechanism_Name	STRING	Get	I_UlosiServerSettings Class [▶ 386]	'2.2.3.1'	Authentication mechanism name. If there is no empty string and the authentication function is enabled, the value is transmitted.
		Set	Local		
sAuthent_Value	STRING	Get	I_UlosiServerSettings Class [▶ 386]	‘‘	Authentication value (password). If there is no empty string and the authentication function is enabled, the value is transmitted.
		Set	Local		
sResponding_Value	STRING	Get	I_UlosiServerSettings Class [▶ 386]	‘‘	Responding authentication value (password). If no empty string, the authentication function is enabled and authentication mode: Mutual then the value is transmitted.
		Set	Local		
eAcse_Authent	E_AcseAuthentication [▶ 474]	Get	I_UlosiServerSettings Class [▶ 386]	E_AcseAuthentication.Unilateral	Authentication mode.
		Set	Local		
tUlosiTimeout	TIME	Get	I_UlosiServerSettings Class [▶ 386]	T#20s	Maximum time that may not be exceeded during the execution of the confirmed services.
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
eTraceUlosi	E_TraceLevel [▶ 526]	Get	I_UlosiServerSettingsClass [▶ 386]	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
		Set	Local		

6.3.2.11 FB_Rfc1006ClientSettingsClass



The function block FB_Rfc1006ClientSettingsClass is used to configure the RFC1006 communication layer.

Namespace: Tc3_Rfc1006 [▶ 102]
Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Inheritance hierarchy

FB_SocketClientSettingsClass [▶ 236] -> FB_TpktClientSettingsClass [▶ 229] -
> FB_Rfc1006ClientSettingsClass

```
FUNCTION_BLOCK FB_Rfc1006ClientSettingsClass EXTENDS FB_TpktClientSettingsClass IMPLEMENTS I_Rfc1006
ClientSettingsClass
```

Interfaces

Type	Description
I_Rfc1006ClientSettingsClass [▶ 388]	Interface for configuring the RFC1006 communication layer

Methods

Name	Description
Defaults [▶ 388]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Definition location	Initial value	Description
bAutoCleanup	BOOL	Get,Set	Inherited from FB_SocketClientSettingsClass [▶ 236]	TRUE	Enables/disables the automatic clean-up/ closure of the unused sockets (e.g. after program download or PLC reset).
sSrvNetID	T_AmsNetID [▶ 529]	Get,Set	-//-	‘‘	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
nRemotePort	UDINT	Get,Set	-//-	102	TCP/IP port number of the server.
sRemoteHost	T_Ipv4Addr [▶ 529]	Get,Set	-//-	‘127.0.0.1‘	IPv4 network address of the server.
tConnect	TIME	Get,Set	-//-	T#45s	Earliest time on expiry of which a new connection can be established.
tAdsTimeout	TIME	Get,Set	-//-	DEFAULT_ADS_TIMEOUT	Maximum timeout time for ADS services that may not be exceeded.
tSocketTimeout	TIME	Get,Set	-//-	T#5s	Maximum timeout time during the execution of the confirmed services that may not be exceeded.
bThrottleMode	BOOL	Get,Set	-//-	TRUE	Enables/disables the receive data throttle polling mode.
throttleTimes	T_ThrottleTimes [▶ 530]	Get,Set	-//-	Param_Sockets.cDefaultReceiverThrottleTimes	Configuration of the cycle times for the receive data throttle polling mode.
eTraceSocket	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
ipTLS	I_SocketTlsSettingsClass [▶ 394]	Get,Set	-//-	0	Configuration parameters of the TLS (Transport Layer Security) protocol layer. Value null = not configured.
bEvaLinkStatusErr	BOOL	Get,Set	-//-	FALSE	Configures the evaluation mode for the link status error of the network adapter. If "FALSE" then disabled, short connection interruptions are bridged. If "TRUE" is enabled, the connection is terminated when a connection error is detected.
tTpktTimeout	TIME	Get,Set	Inherited from FB_TpktClientSettingsClass [▶ 229]	T#10s	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceTpkt	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.

Name	Type	Access	Definition location	Initial value	Description
sCalling_T_Selector	STRING	Get	I_Rfc1006ClientSettingsClass [▶ 388]	'00 00'	RFC 1006 parameter: Calling transport address selector (source).
		Set	Local		
sCalled_T_Selector	STRING	Get	I_Rfc1006ClientSettingsClass [▶ 388]	'00 01'	RFC 1006 parameter: Called transport address selector (destination).
		Set	Local		
eTpduSize	E_Rfc1006TpduSize [▶ 524]	Get	I_Rfc1006ClientSettingsClass [▶ 388]	E_Rfc1006TpduSize.Size_1024	Maximum TPDU byte length.
		Set	Local		
tRfc1006Timeout	TIME	Get	I_Rfc1006ClientSettingsClass [▶ 388]	T#15s	Maximum time that may not be exceeded during the execution of the confirmed services.
		Set	Local		
eTraceRfc1006	E_TraceLevel [▶ 526]	Get	I_Rfc1006ClientSettingsClass [▶ 388]	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
		Set	Local		

6.3.2.12 FB_Rfc1006ConnectionSettingsClass



The function block FB_Rfc1006ConnectionSettingsClass is used to configure the RFC1006 communication layer. The configuration settings only apply to a selected server session function block instance.

Namespace: Tc3_Rfc1006 [▶ 102]
Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Inheritance hierarchy

FB_SocketConnectionSettingsClass [▶ 239] -> FB_TpktConnectionSettingsClass [▶ 232] -
> FB_Rfc1006ConnectionSettingsClass

```
FUNCTION_BLOCK FB_Rfc1006ConnectionSettingsClass EXTENDS FB_TpktConnectionSettingsClass IMPLEMENTS I
_Rfc1006ConnectionSettingsClass
```

Interfaces

Type	Description
I_Rfc1006ConnectionSettingsClass [▶ 388]	Interface for configuring the RFC1006 communication layer.

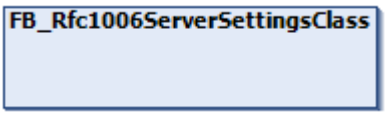
Methods

Name	Description
Defaults [▶ 389]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Definition location	Initial value	Description
bEnable	BOOL	Get,Set	Inherited from FB_SocketConnectionSettingsClass [▶ 239]	FALSE	Enables/disables the establishment of the incoming connection in a server session function block instance.
bTrace	BOOL	Get,Set	-//-	TRUE	Enables/disables the logging of error messages and log messages in a server session function block instance.
eAccept	E_SocketAcceptMode	Get,Set	-//-	E_SocketAcceptMode.eACCEPT_ALL	Configures the acceptance mode for incoming connections. The incoming connections can be accepted or rejected by a session function block instance depending on the network address and/or port number of the client. By default, connections to all clients are accepted.
nAcceptPort	UDINT	Get,Set	-//-	102	Configures the TCP/IP port number of the remote communication partner (clients) to be accepted.
sAcceptHost	T_IPv4Addr [▶ 529]	Get,Set	-//-	'127.0.0.1'	Configures the IPv4 network address of the remote communication partner (clients) to be accepted.

6.3.2.13 FB_Rfc1006ServerSettingsClass



The function block FB_Rfc1006ServerSettingsClass is used to configure the RFC1006 communication layer.

Namespace: Tc3_Rfc1006 [▶ 102]
Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Inheritance hierarchy

FB_SocketServerSettingsClass [▶ 240] -> FB_TpktServerSettingsClass [▶ 233] -> FB_Rfc1006ServerSettingsClass

```
FUNCTION_BLOCK FB_Rfc1006ServerSettingsClass EXTENDS FB_TpktServerSettingsClass IMPLEMENTS I_Rfc1006ServerSettingsClass
```

Interfaces

Type	Description
I_Rfc1006ServerSettingsClass [▶ 389]	Interface for configuring the RFC1006 communication layer

Methods

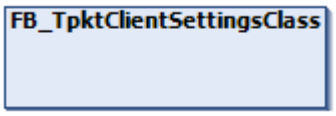
Name	Description
Defaults [▶ 389]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Definition location	Initial value	Description
bAutoCleanup	BOOL	Get,Set	Inherited from FB_SocketServerSettingsClass [▶ 240]	TRUE	Enables/disables the automatic clean-up/ closure of the unused sockets (e.g. after program download or PLC reset).
bEnable	BOOL	Get,Set	-//-	FALSE	Enables/disables establishing the server connections. "TRUE" enables the Listen command (listening and incoming connections are accepted) and "FALSE" disables it.
sSrvNetID	T_AmsNetID [▶ 529]	Get,Set	-//-	‘‘	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
nLocalPort	UDINT	Get,Set	-//-	102	TCP/IP port number of the server.
sLocalHost	T_Ipv4Addr [▶ 529]	Get,Set	-//-	‘127.0.0.1‘	IPv4 network address of the server.
tAccept	TIME	Get,Set	-//-	T#1s	Retry cycle time for the internal TCP/IP socket Accept commands.
tListen	TIME	Get,Set	-//-	T#1s	Retry cycle time for the internal TCP/IP socket Listen command.
tAdsTimeout	TIME	Get,Set	-//-	DEFAULT_ADS_TIMEOUT	Maximum timeout time for ADS services that may not be exceeded.
tSocketTimeout	TIME	Get,Set	-//-	T#5s	Maximum timeout time during the execution of the confirmed services that may not be exceeded.
bThrottleMode	BOOL	Get,Set	-//-	TRUE	Enables/disables the receive data throttle polling mode.
throttleTimes	T_ThrottleTimes [▶ 530]	Get,Set	-//-	Param_Sockets.cDefaultReceiverThrottleTimes	Configuration of the cycle times for the receive data throttle polling mode.
eTraceSocket	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
ipTLS	I_SocketTlsSettingsClass [▶ 394]	Get,Set	-//-	0	Configuration parameters of the TLS (Transport Layer Security) protocol layer. Value null = not configured.
bEvaLinkStatusErr	BOOL	Get,Set	-//-	FALSE	Configures the evaluation mode for the link status error of the network adapter. If "FALSE" then disabled, short connection interruptions are bridged. If "TRUE" is enabled, the connection is terminated when a connection error is detected.
tTpktTimeout	TIME	Get,Set	Inherited from FB_TpktServerSettingsClass [▶ 233]	T#10s	Maximum time that may not be exceeded during the execution of the confirmed services.

Name	Type	Access	Definition location	Initial value	Description
eTraceTpkt	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
sCalling_T_Selector	STRING	Get	I_Rfc1006ServerSettingsClass [▶ 389]	'00 00'	RFC 1006 parameter: Calling transport address selector (source).
		Set	Local		
sCalled_T_Selector	STRING	Get	I_Rfc1006ServerSettingsClass [▶ 389]	'00 01'	RFC 1006 parameter: Called transport address selector (destination).
		Set	Local		
eTpduSize	E_Rfc1006TpduSize [▶ 524]	Get	I_Rfc1006ServerSettingsClass [▶ 389]	E_Rfc1006TpduSize.Size_1024	Maximum TPDU byte length.
		Set	Local		
tRfc1006Timeout	TIME	Get	I_Rfc1006ServerSettingsClass [▶ 389]	T#15s	Maximum time that may not be exceeded during the execution of the confirmed services.
		Set	Local		
eTraceRfc1006	E_TraceLevel [▶ 526]	Get	I_Rfc1006ServerSettingsClass [▶ 389]	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
		Set	Local		

6.3.2.14 FB_TpktClientSettingsClass



The function block FB_TpktClientSettingsClass is used to configure the TPKT communication layer.

Namespace: Tc3_Tpkt [▶ 102]
Library: Tc3_Tpkt (Tc3_Tpkt.compiled-library)

Inheritance hierarchy

FB_SocketClientSettingsClass [▶ 236] -> FB_TpktClientSettingsClass

```
FUNCTION_BLOCK FB_TpktClientSettingsClass EXTENDS FB_SocketClientSettingsClass IMPLEMENTS I_TpktClientSettingsClass
```

Interfaces

Type	Description
I_TpktClientSettingsClass [▶ 390]	Interface for configuring the TPKT communication layer.

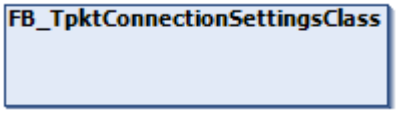
Methods

Name	Description
Defaults [▶ 390]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Definition location	Initial value	Description
bAutoCleanup	BOOL	Get,Set	Inherited from FB_SocketClientSettingsClass [▶ 236]	TRUE	Enables/disables the automatic clean-up/ closure of the unused sockets (e.g. after program download or PLC reset).
sSrvNetID	T_AmsNetID [▶ 529]	Get,Set	-//-	‘‘	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
nRemotePort	UDINT	Get,Set	-//-	102	TCP/IP port number of the server.
sRemoteHost	T_Ipv4Addr [▶ 529]	Get,Set	-//-	‘127.0.0.1‘	IPv4 network address of the server.
tConnect	TIME	Get,Set	-//-	T#45s	Earliest time on expiry of which a new connection can be established.
tAdsTimeout	TIME	Get,Set	-//-	DEFAULT_ADS_TIMEOUT	Maximum timeout time for ADS services that may not be exceeded.
tSocketTimeout	TIME	Get,Set	-//-	T#5s	Maximum timeout time during the execution of the confirmed services that may not be exceeded.
bThrottleMode	BOOL	Get,Set	-//-	TRUE	Enables/disables the receive data throttle polling mode.
throttleTimes	T_ThrottleTimes [▶ 530]	Get,Set	-//-	Param_Sockets.cDefaultReceiverThrottleTimes	Configuration of the cycle times for the receive data throttle polling mode.
eTraceSocket	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
ipTLS	I_SocketTlsSettingsClass [▶ 394]	Get,Set	-//-	0	Configuration parameters of the TLS (Transport Layer Security) protocol layer. Value null = not configured.
bEvaLinkStatusErr	BOOL	Get,Set	-//-	FALSE	Configures the evaluation mode for the link status error of the network adapter. If "FALSE" then disabled, short connection interruptions are bridged. If "TRUE" is enabled, the connection is terminated when a connection error is detected.
tTpktTimeout	TIME	Get	I_TpktClientSettingsClass [▶ 390]	T#10s	Maximum time that may not be exceeded during the execution of the confirmed services.
		Set	Local		
eTraceTpkt	E_TraceLevel [▶ 526]	Get	I_TpktClientSettingsClass [▶ 390]	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
		Set	Local		

6.3.2.15 FB_TpktConnectionSettingsClass



The function block FB_TpktConnectionSettingsClass is used to configure the TPKT communication layer. The configuration settings only apply to a selected server session function block instance.

Namespace: Tc3_Tpkt [▶ 102]
Library: Tc3_Tpkt (Tc3_Tpkt.compiled-library)

Inheritance hierarchy

FB_SocketConnectionSettingsClass [▶ 239] -> FB_TpktConnectionSettingsClass

```
FUNCTION_BLOCK FB_TpktConnectionSettingsClass EXTENDS FB_SocketConnectionSettingsClass IMPLEMENTS I_
TpktConnectionSettingsClass
```

Interfaces

Type	Description
I_TpktConnectionSettingsClass [▶ 390]	Interface for configuring the TPKT communication layer.

Methods

Name	Description
Defaults [▶ 391]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Definition location	Initial value	Description
bEnable	BOOL	Get,Set	Inherited from FB_SocketConnectionSettingsClass [▶ 239]	FALSE	Enables/disables the establishment of the incoming connection in a server session function block instance.
bTrace	BOOL	Get,Set	-//-	TRUE	Enables/disables the logging of error messages and log messages in a server session function block instance.
eAccept	E_SocketAcceptMode	Get,Set	-//-	E_SocketAcceptMode.eACCEPT_ALL	Configures the acceptance mode for incoming connections. The incoming connections can be accepted or rejected by a session function block instance depending on the network address and/or port number of the client. By default, connections to all clients are accepted.
nAcceptPort	UDINT	Get,Set	-//-	102	Configures the TCP/IP port number of the remote communication partner (clients) to be accepted.
sAcceptHost	T_IPv4Addr [▶ 529]	Get,Set	-//-	'127.0.0.1'	Configures the IPv4 network address of the remote communication partner (clients) to be accepted.

6.3.2.16 FB_TpktServerSettingsClass



The function block FB_TpktServerSettingsClass is used to configure the TPKT communication layer.

Namespace: Tc3_Tpkt [▶ 102]
Library: Tc3_Tpkt (Tc3_Tpkt.compiled-library)

Inheritance hierarchy

FB_SocketServerSettingsClass [▶ 240] -> FB_TpktServerSettingsClass

```
FUNCTION_BLOCK FB_TpktServerSettingsClass EXTENDS FB_SocketServerSettingsClass IMPLEMENTS I_TpktServ
erSettingsClass
```

Interfaces

Type	Description
I_TpktServerSettingsClass [▶ 391]	Interface for configuring the TPKT communication layer.

Methods

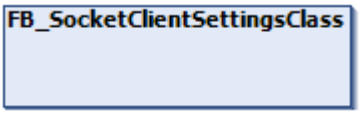
Name	Description
Defaults [▶ 391]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Definition location	Initial value	Description
bAutoCleanup	BOOL	Get,Set	Inherited from FB_SocketServerSettingsClass [▶ 240]	TRUE	Enables/disables the automatic clean-up/ closure of the unused sockets (e.g. after program download or PLC reset).
bEnable	BOOL	Get,Set	-//-	FALSE	Enables/disables establishing the server connections. "TRUE" enables the Listen command (listening and incoming connections are accepted) and "FALSE" disables it.
sSrvNetID	T_AmsNetID [▶ 529]	Get,Set	-//-	‘‘	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
nLocalPort	UDINT	Get,Set	-//-	102	TCP/IP port number of the server.
sLocalHost	T_Ipv4Addr [▶ 529]	Get,Set	-//-	‘127.0.0.1‘	IPv4 network address of the server.
tAccept	TIME	Get,Set	-//-	T#1s	Retry cycle time for the internal TCP/IP socket Accept commands.
tListen	TIME	Get,Set	-//-	T#1s	Retry cycle time for the internal TCP/IP socket Listen command.
tAdsTimeout	TIME	Get,Set	-//-	DEFAULT_ADS_TIMEOUT	Maximum timeout time for ADS services that may not be exceeded.
tSocketTimeout	TIME	Get,Set	-//-	T#5s	Maximum timeout time during the execution of the confirmed services that may not be exceeded.
bThrottleMode	BOOL	Get,Set	-//-	TRUE	Enables/disables the receive data throttle polling mode.
throttleTimes	T_ThrottleTimes [▶ 530]	Get,Set	-//-	Param_Sockets.cDefaultReceiverThrottleTimes	Configuration of the cycle times for the receive data throttle polling mode.
eTraceSocket	E_TraceLevel [▶ 526]	Get,Set	-//-	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
ipTLS	I_SocketTlsSettingsClass [▶ 394]	Get,Set	-//-	0	Configuration parameters of the TLS (Transport Layer Security) protocol layer. Value null = not configured.
bEvaLinkStatusErr	BOOL	Get,Set	-//-	FALSE	Configures the evaluation mode for the link status error of the network adapter. If "FALSE" then disabled, short connection interruptions are bridged. If "TRUE" is enabled, the connection is terminated when a connection error is detected.
tTpktTimeout	TIME	Get	I_TpktServerSettingsClass [▶ 391]	T#10s	Maximum time that may not be exceeded during the execution of the confirmed services.
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
eTraceTpkt	E_TraceLevel [▶ 526]	Get	I_TpktServerSettingsClass [▶ 391]	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
		Set	Local		

6.3.2.17 FB_SocketClientSettingsClass



The function block FB_SocketClientSettingsClass is used to configure the TCP/IP communication layer.

Namespace: Tc3_Sockets [▶ 102]
Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Inheritance hierarchy

FB_SocketClientSettingsClass

```
FUNCTION_BLOCK FB_SocketClientSettingsClass IMPLEMENTS I_SocketClientSettingsClass
```

Interfaces

Type	Description
I_SocketClientSettingsClass [▶ 391]	Interface for configuring the TCP/IP communication layer.

Methods

Name	Description
Defaults [▶ 392]	Resets the configuration settings to the default values.



Properties

Name	Type	Access	Definition location	Initial value	Description
bAutoCleanup	BOOL	Get	I_SocketClientSettings Class [▶ 391]	TRUE	Enables/disables the automatic clean-up/ closure of the unused sockets (e.g. after program download or PLC reset).
		Set	Local		
sSrvNetID	T_AmsNetID [▶ 529]	Get	I_SocketClientSettings Class [▶ 391]	''	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
		Set	Local		
nRemotePort	UDINT	Get	I_SocketClientSettings Class [▶ 391]	102	TCP/IP port number of the server.
		Set	Local		
sRemoteHost	T_Ipv4Addr [▶ 529]	Get	I_SocketClientSettings Class [▶ 391]	'127.0.0.1'	IPv4 network address of the server.
		Set	Local		
tConnect	TIME	Get	I_SocketClientSettings Class [▶ 391]	T#45s	Earliest time on expiry of which a new connection can be established.
		Set	Local		
tAdsTimeout	TIME	Get	I_SocketClientSettings Class [▶ 391]	DEFAULT_ADS_TIMEOUT	Maximum timeout time for ADS services that may not be exceeded.
		Set	Local		
tSocketTimeout	TIME	Get	I_SocketClientSettings Class [▶ 391]	T#5s	Maximum timeout time during the execution of the confirmed services that may not be exceeded.
		Set	Local		
bThrottleMode	BOOL	Get	I_SocketClientSettings Class [▶ 391]	TRUE	Enables/disables the receive data throttle polling mode.
		Set	Local		
throttleTimes	T_ThrottleTimes [▶ 530]	Get	I_SocketClientSettings Class [▶ 391]	Param_Sockets.cDefaultReceiverThrottleTimes	Configuration of the cycle times for the receive data throttle polling mode.
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
eTraceSocket	E_TraceLevel [▶ 526]	Get	I_SocketClientSettingsClass [▶ 391]	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
		Set	Local		
ipTLS	I_SocketTlsSettingsClass [▶ 394]	Get	I_SocketTlsSettingsClass [▶ 394]	0	Configuration parameters of the TLS (Transport Layer Security) protocol layer. Value null = not configured.
		Set	Local		
bEvaLinkStatusErr	BOOL	Get	I_SocketClientSettingsClass [▶ 391]	FALSE	Configures the evaluation mode for the link status error of the network adapter. If "FALSE" then disabled, short connection interruptions are bridged. If "TRUE" is enabled, the connection is terminated when a connection error is detected.
		Set	Local		

6.3.2.18 FB_SocketConnectionSettingsClass



The function block FB_SocketConnectionSettingsClass is used to configure the TCP/IP communication layer. The configuration settings only apply to a selected server session function block instance.

Namespace: Tc3_Sockets [▶ 102]
Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Inheritance hierarchy

FB_SocketConnectionSettingsClass

```
FUNCTION_BLOCK FB_SocketConnectionSettingsClass IMPLEMENTS I_SocketConnectionSettingsClass
```

Interfaces

Type	Description
I_SocketConnectionSettingsClass [▶ 392]	Interface for configuring the TCP/IP communication layer.

Methods

Name	Description
Defaults [▶ 393]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Definition location	Initial value	Description
bEnable	BOOL	Get	I_SocketConnectionSettingsClass [▶ 392]	FALSE	Enables/disables the establishment of the incoming connection in a server session function block instance.
		Set	Local		
bTrace	BOOL	Get	I_SocketConnectionSettingsClass [▶ 392]	TRUE	Enables/disables the logging of error messages and log messages in a server session function block instance.
		Set	Local		
eAccept	E_SocketAcceptMode	Get	I_SocketConnectionSettingsClass [▶ 392]	E_SocketAcceptMode.eACCEPT_ALL	Configures the acceptance mode for incoming connections. The incoming connections can be accepted or rejected by a session function block instance depending on the network address and/or port number of the client. By default, connections to all clients are accepted.
		Set	Local		
nAcceptPort	UDINT	Get	I_SocketConnectionSettingsClass [▶ 392]	102	Configures the TCP/IP port number of the remote communication partner (clients) to be accepted.
		Set	Local		
sAcceptHost	T_IPv4Addr [▶ 529]	Get	I_SocketConnectionSettingsClass [▶ 392]	'127.0.0.1'	Configures the IPv4 network address of the remote communication partner (clients) to be accepted.
		Set	Local		

6.3.2.19 FB_SocketServerSettingsClass



The function block FB_SocketServerSettingsClass is used to configure the TCP/IP communication layer.

Namespace: Tc3_Sockets [▶ 102]
Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Inheritance hierarchy

FB_SocketServerSettingsClass

```
FUNCTION_BLOCK FB_SocketServerSettingsClass IMPLEMENTS I_SocketServerSettingsClass
```

Interfaces

Type	Description
I_SocketServerSettingsClass [▶ 393]	Interface for configuring the TCP/IP communication layer.

Methods

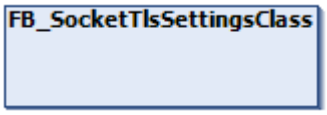
Name	Description
Defaults [▶ 394]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Definition location	Initial value	Description
bAutoCleanup	BOOL	Get	I_SocketServerSettingsClass [▶ 393]	TRUE	Enables/disables the automatic clean-up/ closure of the unused sockets (e.g. after program download or PLC reset).
		Set	Local		
bEnable	BOOL	Get	I_SocketServerSettingsClass [▶ 393]	FALSE	Enables/disables establishing the server connections. "TRUE" enables the Listen command (listening and incoming connections are accepted) and "FALSE" disables it.
		Set	Local		
sSrvNetID	T_AmsNetID [▶ 529]	Get	I_SocketServerSettingsClass [▶ 393]	''	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
		Set	Local		
nLocalPort	UDINT	Get	I_SocketServerSettingsClass [▶ 393]	102	TCP/IP port number of the server.
		Set	Local		
sLocalHost	T_Ipv4Addr [▶ 529]	Get	I_SocketServerSettingsClass [▶ 393]	'127.0.0.1'	IPv4 network address of the server.
		Set	Local		
tAccept	TIME	Get	I_SocketServerSettingsClass [▶ 393]	T#1s	Retry cycle time for the internal TCP/IP socket Accept commands.
		Set	Local		
tListen	TIME	Get	I_SocketServerSettingsClass [▶ 393]	T#1s	Retry cycle time for the internal TCP/IP socket Listen command.
		Set	Local		
tAdsTimeout	TIME	Get	I_SocketServerSettingsClass [▶ 393]	DEFAULT_ADS_TIMEOUT	Maximum timeout time for ADS services that may not be exceeded.
		Set	Local		
tSocketTimeout	TIME	Get	I_SocketServerSettingsClass [▶ 393]	T#5s	Maximum timeout time during the execution of the confirmed services that may not be exceeded.
		Set	Local		

Name	Type	Access	Definition location	Initial value	Description
bThrottleMode	BOOL	Get	I_SocketServerSettingsClass [▶ 393]	TRUE	Enables/disables the receive data throttle polling mode.
		Set	Local		
throttleTimes	T_ThrottleTimes [▶ 530]	Get	I_SocketServerSettingsClass [▶ 393]	Param_Sockets.cDefaultReceiverThrottleTimes	Configuration of the cycle times for the receive data throttle polling mode.
		Set	Local		
eTraceSocket	E_TraceLevel [▶ 526]	Get	I_SocketServerSettingsClass [▶ 393]	E_TraceLevel.None	Configures the priority level for the logging of the error messages and log messages.
		Set	Local		
ipTLS	I_SocketTlsSettingsClass [▶ 394]	Get	I_SocketTlsSettingsClass [▶ 394]	0	Configuration parameters of the TLS (Transport Layer Security) protocol layer. Value null = not configured.
		Set	Local		
bEvaLinkStatusErr	BOOL	Get	I_SocketServerSettingsClass [▶ 393]	FALSE	Configures the evaluation mode for the link status error of the network adapter. If "FALSE" then disabled, short connection interruptions are bridged. If "TRUE" is enabled, the connection is terminated when a connection error is detected.
		Set	Local		

6.3.2.20 FB_SocketTlsSettingsClass



The FB_SocketTlsSettingsClass function block is used to configure the TLS (Transport Layer Security) communication layer. TLS enables secure IEC 61850 client-server communication.

Namespace: Tc3_Sockets [▶ 102]
Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Inheritance hierarchy

FB_SocketTlsSettingsClass

```
FUNCTION_BLOCK FB_SocketTlsSettingsClass IMPLEMENTS I_SocketTlsSettingsClass
```

Interfaces

Type	Description
I_SocketTlsSettingsClass [▶ 394]	TLS configuration interface

Methods

Name	Description
AddCa [▶ 395]	Saves the file path of the CA certificate in the TLS configuration settings
AddCert [▶ 396]	Saves the file path of the client or server certificate in the TLS configuration settings
AddCrl [▶ 396]	Saves the file path of the Certificate Revocation List (CRL) in the TLS configuration settings
AddPsk [▶ 396]	Saves the PSK (pre-shared key) and PSK identity string in the TLS configuration settings
Reset [▶ 397]	Resets all TLS configuration settings
SetConnectFlags [▶ 397]	Saves additional, optional TLS configuration settings of the client
SetListenFlags [▶ 397]	Saves additional, optional TLS configuration settings of the server

Properties

Name	Type	Access	Definition location	Initial value	Description
nSecurePort	UDINT	Get	I_SocketTlsSettingsClass [▶ 394]	3782	Secure ISO TP0 (IANA) port number for TLS communication
		Set	Local		
connectFlags	ST_TlsConnectFlags	Get	I_SocketTlsSettingsClass [▶ 394]	DEFAULT_TLSCONNECTFLAGS	Additional TLS configuration settings of the client
listenFlags	ST_TlsListenFlags	Get	I_SocketTlsSettingsClass [▶ 394]	DEFAULT_TLSLISTENFLAGS	Additional TLS configuration settings of the server
pskKey	PVOID	Get	I_SocketTlsSettingsClass [▶ 394]	0	Points to the first byte of the PSK key
pskKeyLen	UDINT(0..TCPADS_TLS_MAX_PSK_KEY_SIZE)	Get	I_SocketTlsSettingsClass [▶ 394]	0	Byte length of the PSK key
sCaPath	STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE)	Get	I_SocketTlsSettingsClass [▶ 394]	''	File path of the CA certificate
sCertPath	STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE)	Get	I_SocketTlsSettingsClass [▶ 394]	''	File path of the client or server certificate
sCrlPath	STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE)	Get	I_SocketTlsSettingsClass [▶ 394]	''	File path of the certificate revocation list (CRL)
sIdentity	STRING(TCPADS_TLS_PSK_IDENTITY_SIZE)	Get	I_SocketTlsSettingsClass [▶ 394]	''	Identity string of the PSK key
sKeyPath	STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE)	Get	I_SocketTlsSettingsClass [▶ 394]	''	File path of the private key
sKeyPwd	STRING(TCPADS_TLS_KEY_PASSWORD_SIZE)	Get	I_SocketTlsSettingsClass [▶ 394]	''	Password string of the certificate

6.3.3 Communication

6.3.3.1 FB_iec61850ClientClass



The function block FB_iec61850CommonClass implements interfaces for communication and data exchange between the TwinCAT PLC client application and the IEC 61850 server.

Namespace: Tc3_iec61850 [▶ 102]
Library: Tc3_iec61850 (Tc3_iec61850.compiled-library)

Inheritance hierarchy

FB_iec61850ClientClass

```
FUNCTION_BLOCK FB_iec61850ClientClass IMPLEMENTS
    I_ScsmClientClass,
    I_ScsmSystemClockEventSink,
    I_ScsmAbortIndEventSink,
    I_ScsmAssociateCnfEventSink,
    I_ScsmReleaseIndEventSink,
    I_ScsmReleaseCnfEventSink,
    I_ScsmGetDataValuesCnfEventSink,
    I_ScsmSetDataValuesCnfEventSink,
    I_ScsmControlCnfEventSink,
    I_ScsmGetServerDirectoryCnfEventSink,
    I_ScsmGetLogicalDeviceDirectoryCnfEventSink,
    I_ScsmGetLogicalNodeDirectoryCnfEventSink,
    I_ScsmCreateDataSetCnfEventSink,
    I_ScsmDeleteDataSetCnfEventSink,
    I_ScsmGetDataSetDirectoryCnfEventSink,
    I_ScsmReportIndEventSink,
    I_ScsmLastApplErrorIndEventSink,
    I_ScsmCommandTerminationIndEventSink,
    I_ScsmIdentifyCnfEventSink,
    I_ScsmStatusCnfEventSink
VAR_INPUT
    settings : FB_ScsmClientSettingsClass;
END_VAR
```

Interfaces

Type	Description
I_ScsmClientClass [▶ 409]	Interfaces for establishment of communication and data exchange.
I_ScsmAbortIndEventSink [▶ 438]	Interface of the Abort service event handling routine.
I_ScsmAssociateCnfEventSink [▶ 439]	Interface of the Associate service event handling routine.
I_ScsmReleaseCnfEventSink [▶ 450]	Interface of the Release service event handling routine.
I_ScsmReleaseIndEventSink [▶ 451]	Interface of the Release service event handling routine.
I_ScsmGetDataValuesCnfEventSink [▶ 446]	Interface of the GetDataValues service event handling routine.
I_ScsmSetDataValuesCnfEventSink [▶ 452]	Interface of the SetDataValues event handling routine.
I_ScsmControlCnfEventSink [▶ 443]	Interface of the control service event handling routine for switch control (Operate, Cancel, Select, SelectWithValue).
I_ScsmGetServerDirectoryCnfEventSink [▶ 448]	Interface of the GetServerDirectory service event handling routine.
I_ScsmGetLogicalDeviceDirectoryCnfEventSink [▶ 447]	Interface of the GetLogicalDeviceDirectory service event handling routine.
I_ScsmGetLogicalNodeDirectoryCnfEventSink [▶ 447]	Interface of the GetLogicalNodeDirectory service event handling routine.
I_ScsmCreateDataSetCnfEventSink [▶ 444]	Interface of the CreateDataSet service event handling routine.
I_ScsmDeleteDataSetCnfEventSink [▶ 445]	Interface of the DeleteDataSet service event handling routine.
I_ScsmGetDataSetDirectoryCnfEventSink [▶ 445]	Interface of the GetDataSetDirectory service event handling routine.
I_ScsmSystemClockEventSink [▶ 454]	Interface of the SystemClock service event handling routine.
I_ScsmReportIndEventSink [▶ 451]	Interface of the Report service event handling routine.
I_ScsmLastApplErrorIndEventSink [▶ 449]	Interface of the LastAppError service event handling routine.
I_ScsmCommandTerminationIndEventSink [▶ 443]	Interface of the command termination event handling routine interface for switch control.
I_ScsmIdentifyCnfEventSink [▶ 449]	Interface of the MMS-Identify service event handling routine.
I_ScsmStatusCnfEventSink [▶ 453]	Interface of the MMS-Status service event handling routine.

Inputs

Name	Type	Description
settings	FB_ScsmClientSettingsClass [▶ 176]	Protocol settings for the communication layer.

Methods

Name	Description
Execute	This method triggers the execution of the communication layer.
AssociateReq [▶ 413]	Enables the service: Associate.
ReleaseReq [▶ 428]	Enables the service: Release.
ReleaseRsp [▶ 428]	Responds to the service: Release.
AbortReq [▶ 413]	Enables the service: Abort.
ControlCancelReq [▶ 414]	Enables the service: Cancel.
ControlOperateReq [▶ 415]	Enables the service: Operate.
ControlSelectReq [▶ 416]	Enables the service: Select.
ControlSelectWithValueReq [▶ 417]	Enables the service: SelectWithValue.
CreateDataSetReq [▶ 418]	Enables the service: CreateDataSet.
DeleteDataSetReq [▶ 419]	Enables the service: DeleteDataSet.
GetAllDataValuesReq [▶ 419]	Enables the service: GetAllDataValues.
GetAllLogicalDeviceValuesReq [▶ 420]	Enables the service: GetAllLogicalDeviceValues (TwinCAT-specific).
GetAllServerValuesReq [▶ 420]	Enables the service: GetAllServerValues (TwinCAT-specific).
GetBrCBValuesReq [▶ 421]	Enables the service: GetBRCBValues.
GetDataSetDirectoryReq [▶ 422]	Enables the service: GetDataSetDirectory.
GetDataSetValuesReq [▶ 422]	Enables the service: GetDataSetValues.
GetDataValuesReq [▶ 423]	Enables the service: GetDataValues.
GetGoCBValuesReq [▶ 424]	Enables the service: GetGoCBValues.
GetGsCBValuesReq	Enables the service: GetGsCBValues.
GetLCBValuesReq	Enables the service: GetLCBValues.
GetLogicalDeviceDirectoryReq [▶ 424]	Enables the service: GetLogicalDeviceDirectory.
GetLogicalNodeDirectoryReq [▶ 425]	Enables the service: GetLogicalNodeDirectory.
GetMsvCBValuesReq [▶ 425]	Enables the service: GetMSVCBValues.
GetServerDirectoryReq [▶ 426]	Enables the service: GetServerDirectory.
GetSgCBValuesReq	Enables the service: GetSGCBValuesReq.
GetUrCBValuesReq [▶ 427]	Enables the service: GetURCBValues.
GetUsvCBValuesReq	Enables the service: GetUSVCBValues.
SetAllDataValuesReq [▶ 429]	Enables the service: SetAllDataValues (TwinCAT-specific).
SetBrCBValuesReq [▶ 429]	Enables the service: SetBRCBValues.
SetDataSetValuesReq [▶ 430]	Enables the service: SetDataSetValues (TwinCAT-specific).
SetDataValuesReq [▶ 431]	Enables the service: SetDataValues.
SetGoCBValuesReq [▶ 431]	Enables the service: SetGoCBValues.
SetGsCBValuesReq	Enables the service: SetGsCBValues.
SetLCBValuesReq	Enables the service: SetLCBValues.
SetMsvCBValuesReq [▶ 432]	Enables the service: SetMSVCBValues.
SetSgCBValuesReq	Enables the service: SetSGCBValues.
SetUrCBValuesReq [▶ 433]	Enables the service: SetURCBValues.

Name	Description
SetUsvCBValuesReq	Enables the service: SetUSVCBValues.
IdentifyReq [▶ 427]	Enables the service: MMS-Identify.
StatusReq [▶ 433]	Enables the service: MMS-Status.

Methods of event handling (callback methods)

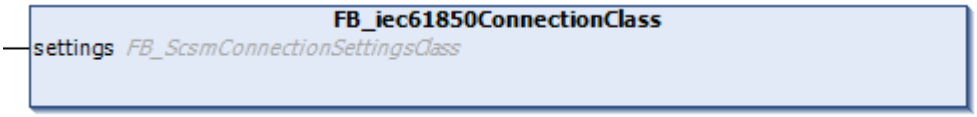
Name	Description
OnAssociateCnf [▶ 439]	User-defined associate service event handling routine.
OnReleaseCnf [▶ 450]	User-defined release service event handling routine.
OnReleaseInd [▶ 451]	User-defined release service event handling routine.
OnAbortInd [▶ 438]	User-defined abort service event handling routine.
OnControlCnf [▶ 444]	User-defined control service event handling routine for switch control (Operate, Cancel, Select, SelectWithValue).
OnCreateDataSetCnf [▶ 444]	User-defined CreateDataSet service event handling routine.
OnDeleteDataSetCnf [▶ 445]	User-defined DeleteDataSet service event handling routine.
OnGetDataSetDirectoryCnf [▶ 446]	User-defined GetDataSetDirectory service event handling routine.
OnGetDataValuesCnf [▶ 446]	User-defined GetDataValues service event handling routine.
OnGetLogicalDeviceDirectoryCnf [▶ 447]	User-defined GetLogicalDeviceDirectory service event handling routine.
OnGetLogicalNodeDirectoryCnf [▶ 448]	User-defined GetLogicalNodeDirectory service event handling routine.
OnGetServerDirectoryCnf [▶ 448]	User-defined GetServerDirectory service event handling routine.
OnLastApplErrorInd [▶ 450]	User-defined LastApplError service event handling routine.
OnBufferedReportInd [▶ 451]	User-defined buffered report service event handling routine.
OnUnbufferedReportInd [▶ 452]	User-defined unbuffered report service event handling routine.
OnSetDataValuesCnf [▶ 453]	User-defined SetDataValues service event handling routine.
OnGetSystemTime [▶ 454]	User-defined system clock service event handling routine.
OnCommandTerminationInd [▶ 443]	User-defined command termination event handling routine for switch control.
OnIdentifyCnf [▶ 449]	User-defined MMS-identify service event handling routine.
OnStatusCnf [▶ 453]	User-defined MMS-Status service event handling routine.

Properties

Name	Type	Access	Description
ipIED [▶ 428]	I_AcsiCommonIntelligentElectronicDeviceClass [▶ 368]	Get,Set	Interface pointer of an object with the implementation of the top-level of the IEC 61850 data model.
eLinkStatus	E_ScsmLinkStatus [▶ 525]	Get	Client-Link-Status. Link status of the network adapter (client sockets).
eState	E_AsyncEnvironmentState [▶ 519]	Get	Environmental state of the communication connection.
ipSettings	I_ScsmClientSettingsClass [▶ 376]	Get	Protocol settings for the communication layer.
ipAssociateCnf	I_ScsmAssociateCnfEventSink [▶ 439]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnAssociateCnf [▶ 439].
ipReleaseCnf	I_ScsmReleaseCnfEventSink [▶ 450]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnReleaseCnf [▶ 450].
ipReleaseInd	I_ScsmReleaseIndEventSink [▶ 451]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnReleaseInd [▶ 451].
ipAbortInd	I_ScsmAbortIndEventSink [▶ 438]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnAbortInd [▶ 438].
ipSystemClock	I_ScsmSystemClockEventSink [▶ 454]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetSystemTime [▶ 454].
ipControlCnf	I_ScsmControlCnfEventSink [▶ 443]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnControlCnf [▶ 444].
ipCreateDataSetCnf	I_ScsmCreateDataSetCnfEventSink [▶ 444]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnCreateDataSetCnf [▶ 444].
ipDeleteDataSetCnf	I_ScsmDeleteDataSetCnfEventSink [▶ 445]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnDeleteDataSetCnf [▶ 445].
ipGetDataSetDirectoryCnf	I_ScsmGetDataSetDirectoryEventSink [▶ 445]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetDataSetDirectoryCnf [▶ 446].
ipGetDataValuesCnf	I_ScsmGetDataValuesCnfEventSink [▶ 446]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetDataValuesCnf [▶ 446].
ipGetLogicalDeviceDirectoryCnf	I_ScsmGetLogicalDeviceDirectoryCnfEventSink [▶ 447]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetLogicalDeviceDirectoryCnf [▶ 447].
ipGetLogicalNodeDirectoryCnf	I_ScsmGetLogicalNodeDirectoryCnfEventSink [▶ 447]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetLogicalNodeDirectoryCnf [▶ 448].
ipGetServerDirectoryCnf	I_ScsmGetServerDirectoryCnfEventSink [▶ 448]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetServerDirectoryCnf [▶ 448].
ipIdentifyCnf	I_ScsmIdentifyCnfEventSink [▶ 449]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnIdentifyCnf [▶ 449].

Name	Type	Access	Description
ipLastApplErrorInd	I_ScsmLastApplErrorIndEventSink [▶ 449]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnLastApplErrorInd [▶ 450].
ipReportInd	I_ScsmReportIndEventSink [▶ 451]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnBufferedReportInd [▶ 451] and OnUnbufferedReportInd [▶ 452].
ipSetDataValuesCnf	I_ScsmSetDataValuesCnfEventSink [▶ 452]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnSetDataValuesCnf [▶ 453].
ipStatusCnf	I_ScsmStatusCnfEventSink [▶ 453]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnStatusCnf [▶ 453].
ipCommandTerminationInd	I_ScsmCommandTerminationIndEventSink [▶ 443]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnCommandTerminationInd [▶ 443].
ipAssociation	I_ScsmAssociationClass [▶ 440]	Get	Interface pointer of an object that identifies the client connection.
sDesc	STRING	Get,Set	The description of the client connection.

6.3.3.2 FB_iec61850ConnectionClass



The function block FB_iec61850ConnectionClass implements interfaces for establishing communication and exchanging data between the TwinCAT PLC server application and the IEC 61850 client. A function block instance implements a server session (a server connection with a remote client).

Namespace: Tc3_iec61850 [▶ 102]
Library: Tc3_iec61850 (Tc3_iec61850.compiled-library)

Inheritance hierarchy

FB_iec61850ConnectionClass

```
FUNCTION_BLOCK FB_iec61850ConnectionClass IMPLEMENTS
    I_ScsmConnectionClass,
    I_ScsmAbortIndEventSink,
    I_ScsmAssociateIndEventSink,
    I_ScsmReleaseCnfEventSink,
    I_ScsmReleaseIndEventSink
VAR_INPUT
    settings : FB_ScsmConnectionSettingsClass;
END_VAR
```

Interfaces

Type	Description
I_ScsmConnectionClass [▶ 434]	Interfaces for establishment of communication and data exchange.
I_ScsmAbortIndEventSink [▶ 438]	Interface of the Abort service event handling routine.
I_ScsmAssociateIndEventSink [▶ 439]	Interface of the Associate service event handling routine.
I_ScsmReleaseCnfEventSink [▶ 450]	Interface of the Release service event handling routine.
I_ScsmReleaseIndEventSink [▶ 451]	Interface of the Release service event handling routine.

Inputs

Name	Type	Description
settings	FB_ScsmConnectionSettingsClass [▶ 183]	Protocol settings for the communication layer.

Methods

Name	Description
Execute	This method triggers the execution of the communication layer.
AssociateRsp [▶ 436]	Responds to the service: Associate.
ReleaseReq [▶ 436]	Enables the service: Release.
ReleaseRsp [▶ 436]	Responds to the service: Release.
AbortReq [▶ 435]	Enables the service: Abort.

Methods of event handling (callback methods)

Name	Description
OnAssociateInd [▶ 439]	User-defined Associate service event handling routine.
OnReleaseCnf [▶ 450]	User-defined release service event handling routine.
OnReleaseInd [▶ 451]	User-defined release service event handling routine.
OnAbortInd [▶ 438]	User-defined abort service event handling routine.

Properties

Name	Type	Access	Description
eLinkStatus	E_ScsmLinkStatus [▶ 525]	Get	Server Link status. Link status of the network adapter (accepted sockets).
eState	E_AsyncEnvironmentState [▶ 519]	Get	Environmental state of the communication connection.
ipSettings	I_ScsmConnectionSettingsClass [▶ 377]	Get	Protocol settings for the communication layer.
ipServer	I_ScsmServerClass [▶ 437]	Get,Set	Interface pointer of an object that implements the server (container).
ipAssociateInd	I_ScsmAssociateIndEventSink [▶ 439]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnAssociateInd [▶ 439].
ipReleaseCnf	I_ScsmReleaseCnfEventSink [▶ 450]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnReleaseCnf [▶ 450].
ipReleaseInd	I_ScsmReleaseIndEventSink [▶ 451]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnReleaseInd [▶ 451].
ipAbortInd	I_ScsmAbortIndEventSink [▶ 438]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnAbortInd [▶ 438].
ipAssociation	I_ScsmAssociationClass [▶ 440]	Get	Interface pointer of an object that identifies the server connection (session).
sDesc	STRING	Get,Set	The description of the server connection (session).

6.3.3.3 FB_iec61850ServerClass



The function block FB_iec61850ServerClass implements an interface for establishment of communication and data exchange between the TwinCAT Server application and the IEC 61850 Client.

Namespace: Tc3_iec61850 [▶ 102]
Library: Tc3_iec61850 (Tc3_iec61850.compiled-library)

Inheritance hierarchy

FB_iec61850ServerClass

```
FUNCTION_BLOCK FB_iec61850ServerClass IMPLEMENTS I_ScsmServerClass
VAR_INPUT
    settings : FB_ScsmServerSettingsClass;
END_VAR
```

Interfaces

Type	Description
I_ScsmServerClass [▶ 437]	Interfaces for establishment of communication and data exchange.

Inputs

Name	Type	Description
settings	FB_ScsmServerSettingsClass [▶ 184]	Protocol settings for the communication layer.

Methods

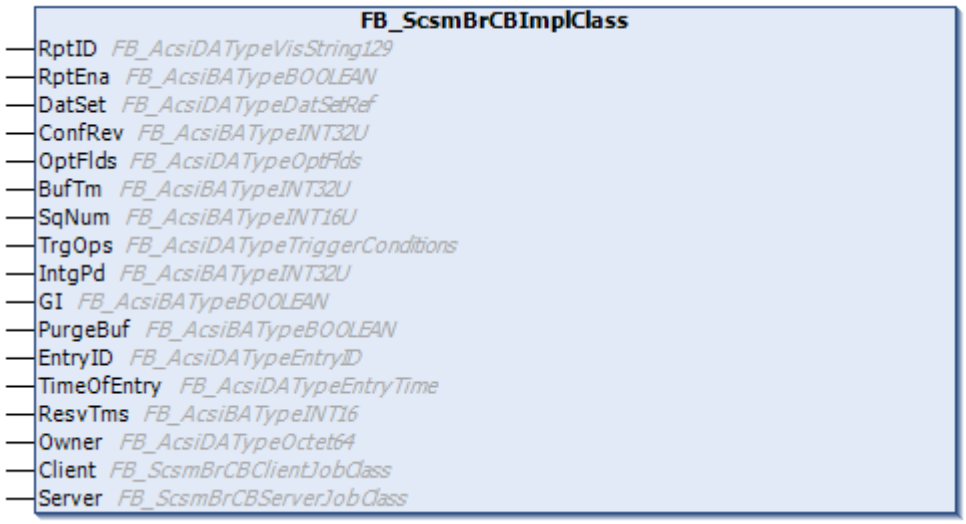
Name	Description
AddConnection [▶ 437]	Adds a connection instance to the Server.
Execute	This method triggers the execution of the communication layer as well as the connection instances.
RemoveConnection [▶ 438]	Removes a connection instance from the server.

Properties

Name	Type	Access	Description
eLinkStatus	E_ScsmLinkStatus [▶ 525]	Get	Server link status of the network adapter (listener sockets).
ipConnections	I_ScsmConnectionClass [▶ 434]	Get	Interface pointer from an object with the implementation of the first connection instance.
ipIED	I_AcsiCommonIntelligentElectronicDeviceClass [▶ 368]	Get, Set	Interface pointer of an object with the implementation of the top-level of the IEC 61850 data model.
ipSettings	I_ScsmServerSettingsClass [▶ 378]	Get	Protocol settings for the communication layer.
ipSystemClock	I_ScsmSystemClockEventSink [▶ 454]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetSystemTime [▶ 454].
nItems	UDINT	Get	The number of connection instances.
sDesc	STRING	Get, Set	The description of the server (container).

6.3.3.4 Control block access

6.3.3.4.1 FB_ScsmBrCBImplClass



The function block FB_ScsmBrCBImplClass contains the client and server implementation of the properties and functions of a BRCB control block. For each configured GOOSE control block an instance of FB_GseBrCBImplClass is created by the TwinCAT Telecontrol Configurator.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonBufferedReportControlBlockClass [▶ 152] -
> FB_AcsiBufferedReportControlBlock [▶ 170] -> FB_ScsmBrCBImplClass

```
FUNCTION_BLOCK FB_ScsmBrCBImplClass EXTENDS FB_AcsiBufferedReportControlBlock IMPLEMENTS I_ScsmBrCBI
mplClass
VAR_INPUT
```

```
    Client : FB_ScsmBrCBClientJobClass;
    Server : FB_ScsmBrCBServerJobClass;
END_VAR
```

Interfaces

Type	Description
I_ScsmBrCBImplClass [▶ 454]	Interface of the BRCB control block implementation.

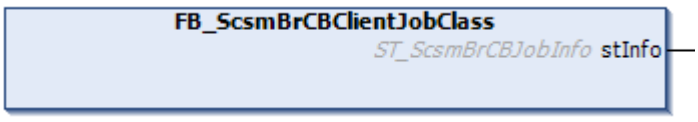
Inputs

Name	Type	Definition location	Description
Client	FB_ScsmBrCBClientJobClass [▶ 259]	Local	Client implementation of the BRCB control block.
Server	FB_ScsmBrCBServerJobClass [▶ 260]	Local	Server implementation of the BRCB control block.
RptID	FB_AcsiDATypeVisString129 [▶ 151]	Inherited from FB_AcsiBufferedReportControlBlock [▶ 170]	Data attribute, Ed1=>RptID:VisStr[65]!
RptEna	FB_AcsiBATypeBOOLEAN [▶ 103]	-//-	Data attribute
DatSet	FB_AcsiDATypeDatSetRef [▶ 117]	-//-	Data attribute
ConfRev	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
OptFlds	FB_AcsiDATypeOptFlds [▶ 128]	-//-	Data attribute
BufTm	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
SqNum	FB_AcsiBATypeINT16U [▶ 105]	-//-	Data attribute
TrgOps	FB_AcsiDATypeTriggerConditions [▶ 147]	-//-	Data attribute
IntgPd	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
GI	FB_AcsiBATypeBOOLEAN [▶ 103]	-//-	Data attribute
PurgeBuf	FB_AcsiBATypeBOOLEAN [▶ 103]	-//-	Data attribute
EntryID	FB_AcsiDATypeEntryID [▶ 119]	-//-	Data attribute
TimeOfEntry	FB_AcsiDATypeEntryTime [▶ 119]	-//-	Data attribute
ResvTms	FB_AcsiBATypeINT16 [▶ 105]	-//-	Data attribute, Ed2.
Owner	FB_AcsiDATypeOctet64 [▶ 127]	-//-	Data attribute, Ed2.

Properties

Name	Type	Access	Definition location	Description
bGI	BOOL	Get, Set	I_ScsmBrCBImplClass [▶ 454]	Attribute value: Enabling the general interrogation command.
bPurgeBuf	BOOL	Get, Set	-//-	Attribute value: Enabling the purge command.
bRptEna	BOOL	Get, Set	-//-	Attribute value: Enabling/disabling the RCB control block. Further configuration needed.
cOptFlds	ST_AcsiOptionalFields [▶ 536]	Get, Set	-//-	Attribute value: Configuration of the optional fields that are transferred with a report.
cTrgOps	ST_AcsiTriggerConditions [▶ 544]	Get, Set	-//-	Attribute value: Configuration of the trigger options for sending a report.
iResvTms	INT	Get, Set	-//-	Attribute value: Configuration of the time for reserving the RCB for a specific client.
nBufTm	DWORD	Get, Set	-//-	Attribute value: Configuration of the max. report buffer time in milliseconds.
nConfRev	DWORD	Get, Set	-//-	Attribute value: Revision number of the RCB control block configuration.
nEntryID	LWORD	Get, Set	-//-	Attribute value: ID of the report entry as 64-bit numerical value.
nIntgPd	DWORD	Get, Set	-//-	Attribute value: Configuration of the max. time between integrity reports in milliseconds.
nSqNum	WORD	Get, Set	-//-	Attribute value: Report sequence number.
oEntryID	T_OCTET8 [▶ 558]	Get, Set	-//-	Attribute value: ID of the report entry as octet string.
oOwner	T_OCTET64 [▶ 559]	Get, Set	-//-	Attribute value: Owner of the RCB. IP address of the client which has reserved the RCB for itself (offline/online reservation).
sDatSet	T_AcsiObjectReference [▶ 555]	Get, Set	-//-	Attribute value: Name of the linked data set.
sRptID	T_AcsiVisString129 [▶ 556]	Get, Set	-//-	Attribute value: Report identification string.
tTimeOfEntry	T_BinaryTime [▶ 556]	Get, Set	-//-	Attribute value: Report timestamp.
ipDatSet	I_AcsiCommonDataSetClass [▶ 366]	Get	-//-	Interface pointer of the linked data set.
ipClient	I_ScsmBrCBClientJobClass [▶ 455]	Get	-//-	Interface pointer of the client implementation of the BRCB control block.
ipServer	I_ScsmBrCBServerJobClass [▶ 457]	Get	-//-	Interface pointer of the server implementation of the BRCB control block.

6.3.3.4.2 FB_ScsmBrCBClientJobClass



Client implementation of the BRCB control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

FB_AsyncServiceResultClass [▶ 299] -> FB_ScsmBrCBClientJobClass

```
FUNCTION_BLOCK FB_ScsmBrCBClientJobClass EXTENDS FB_AsyncServiceResultClass IMPLEMENTS I_ScsmBrCBCli
entJobClass
VAR_OUTPUT
    stInfo : ST_ScsmBrCBJobInfo;
END_VAR
```

Interfaces

Type	Description
I_ScsmBrCBClientJobClass [▶ 455]	Client interface of the BRCB control block implementation.

Outputs

Name	Type	Description
stInfo	ST_ScsmBrCBJobInfo [▶ 550]	Information received in the last report message.

Methods

Name	Description
EnableReq [▶ 456]	Enables the control block.
DisableReq [▶ 456]	Disables the control block.
GIreq [▶ 456]	Enables the general interrogation command.
PurgeBufReq [▶ 457]	Enables the Purge command.
ResyncReq [▶ 457]	Enables the Resync command.

Properties

Name	Type	Access	Description
cOptFlds	ST_AcsiOptionalFields [▶ 536]	Set	Optional fields that are transferred in a report.
cTrgOps	ST_AcsiTriggerConditions [▶ 544]	Set	Trigger options for sending a report.
iResvTms	INT(-1..3600)	Set	Configures the time for reserving the RCB for a specific client.
nBufTm	DWORD	Set	Max. report buffer time in milliseconds.
nIntgPd	DWORD	Set	Max. time between integrity reports in milliseconds.
sRptID	T_AcsiVisString129 [▶ 556]	Set	Report identification string.

6.3.3.4.3 FB_ScsmBrCBServerJobClass



Server implementation of the BRCB control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

FB_AsyncServiceResultClass [▶ 299] -> FB_ScsmBrCBServerJobClass

```
FUNCTION_BLOCK FB_ScsmBrCBServerJobClass EXTENDS FB_AsyncServiceResultClass IMPLEMENTS I_ScsmBrCBSer
verJobClass
VAR_OUTPUT
    stInfo : ST_ScsmBrCBJobInfo;
END_VAR
```

Interfaces

Type	Description
I_ScsmBrCBServerJobClass [▶ 457]	Server interface of the BRCB control block implementation.

Outputs

Name	Type	Description
stInfo	ST_ScsmBrCBJobInfo [▶ 550]	Information sent in the last report message.

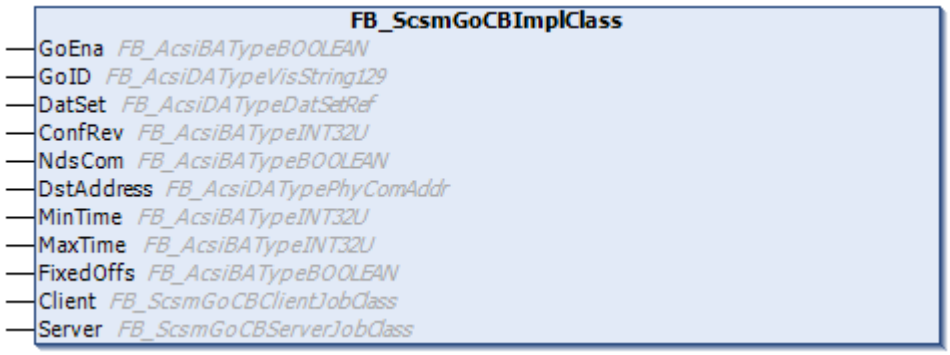
Methods

Name	Description
Execute [▶ 457]	Runs the RCB state machine of the server.

Properties

Name	Type	Access	Description
bOverwriteSameCycleChanges	BOOL	Set	If TRUE, multiple value changes within a PLC cycle and a data object are treated as single value changes (only one report is generated). If FALSE, each value change generates its own report.
bReplaceBufferedMX	BOOL	Set	Configures whether data of the functional group "MX" should be handled separately, like data of the functional group "ST".
nMaxBufferEntries	UDINT	Set	Configures the maximum number of buffered report entries.
sResvClient	STRING	Set	String with the IP address of the client preconfigured in SCL (offline reservation). Example: '192.168.10.140'. If set, then in the server access to the RCB instance is restricted to the client with this IP address ("ClientLN" entries in the SCL). Default: Empty string (online reservation). With the empty string, each client can dynamically reserve access to the RCB for itself.

6.3.3.4.4 FB_ScsmGoCBImplClass



The function block FB_ScsmGoCBImplClass contains the client and server implementation of the properties and functions of a GOOSE control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonGooseControlBlockClass [▶ 155] -
> FB_AcsiGooseControlBlock [▶ 171] -> FB_ScsmGoCBImplClass

```
FUNCTION_BLOCK FB_ScsmGoCBImplClass EXTENDS FB_AcsiGooseControlBlock IMPLEMENTS I_ScsmGoCBImplClass
VAR_INPUT
    Client : FB_ScsmGoCBClientJobClass;
    Server : FB_ScsmGoCBServerJobClass;
END_VAR
```

Interfaces

Type	Description
I_ScsmGoCBImplClass [▶ 458]	Interface of the GOOSE control block implementation.

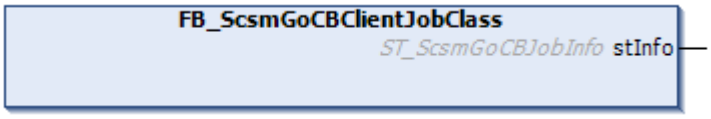
Inputs

Name	Type	Definition location	Description
Client	FB_ScsmGoCBClientJobClass [▶ 264]	Local	Client implementation of the GOOSE control block.
Server	FB_ScsmGoCBServerJobClass [▶ 265]	Local	Server implementation of the GOOSE control block.
GoEna	FB_AcsiBATypeBOOLEAN [▶ 103]	Inherited from FB_AcsiGooseControlBlock [▶ 171]	Data attribute
GoID	FB_AcsiDATypeVisString129 [▶ 151]	-//-	Data attribute
DatSet	FB_AcsiDATypeDatSetRef [▶ 117]	-//-	Data attribute
ConfRev	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
NdsCom	FB_AcsiBATypeBOOLEAN [▶ 103]	-//-	Data attribute
DstAddress	FB_AcsiDATypePhyComAddr [▶ 132]	-//-	Data attribute
MinTime	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
MaxTime	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
FixedOffs	FB_AcsiBATypeBOOLEAN [▶ 103]	-//-	Data attribute

Properties

Name	Type	Access	Definition location	Description
bFixedOffs	BOOL	Get	I_ScsmGoCBImplClass [▶ 458]	Attribute value: Enables/disables coding with fixed lengths.
bGoEna	BOOL	Get, Set	-//-	Attribute value: Enables/disables the GOOSE control block.
bNdsCom	BOOL	Get	-//-	Attribute value: Indicates whether the GOOSE control block requires further configuration.
ePRIORITY	E_AcsiVLanPriority [▶ 517]	Get	-//-	Priority level in the VLAN network.
ipDatSet	I_AcsiCommonDataSetClass [▶ 366]	Get	-//-	Interface pointer to the linked data set.
macAddr	ETHERNET_ADDRESS [▶ 528]	Get	-//-	Target MAC address.
nAPPID	WORD	Get	-//-	Application Identifier.
nConfRev	UDINT	Get	-//-	Revision number of the configuration of the GOOSE control block.
nMaxTime	DWORD	Get	-//-	Maximum time between repeated transmissions of a GOOSE message in milliseconds.
nMinTime	DWORD	Get	-//-	Time between direct sending after data change and first repetition in milliseconds.
nVID	WORD(0..4095)	Get	-//-	ID of the VLAN network.
oAddr	T_OCTET6 [▶ 558]	Get	-//-	Target MAC address.
sAddr	STRING(17)	Get	-//-	Target MAC address.
sDatSet	T_AcsiObjectReference [▶ 555]	Get, Set	-//-	Name of the linked data set.
sGoID	T_AcsiVisString129 [▶ 556]	Get, Set	-//-	The ID of the GOOSE messages.
ipClient	I_ScsmGoCBClientJobClass [▶ 458]	Get	-//-	Interface pointer of the client implementation of the GOCB control block.
ipServer	I_ScsmGoCBServerJobClass [▶ 459]	Get	-//-	Interface pointer of the server implementation of the GOCB control block.

6.3.3.4.5 FB_ScsmGoCBClientJobClass



Client implementation of the GOCB control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

FB_AsyncServiceResultClass [▶ 299] -> FB_ScsmGoCBClientJobClass

```
FUNCTION_BLOCK FB_ScsmGoCBClientJobClass EXTENDS FB_AsyncServiceResultClass IMPLEMENTS I_ScsmGoCBCli
entJobClass
VAR_OUTPUT
    stInfo : ST_ScsmGoCBJobInfo;
END_VAR
```

Interfaces

Type	Description
I_ScsmGoCBClientJobClass [▶ 458]	Client interface of the GOCB control block implementation.

Outputs

Name	Type	Description
stInfo	ST_ScsmGoCBJobInfo [▶ 552]	

Methods

Name	Description
EnableReq [▶ 459]	Enables the control block.
DisableReq [▶ 459]	Disables the control block.

Properties

Name	Type	Access	Description
sGoID	T_AcsiVisString129 [▶ 556]	Set	GOOSE message identification string.

6.3.3.4.6 FB_ScsmGoCBServerJobClass



Server implementation of the GOCB control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

FB_AsyncServiceResultClass [▶ 299] -> FB_ScsmGoCBServerJobClass

```
FUNCTION_BLOCK FB_ScsmGoCBServerJobClass EXTENDS FB_AsyncServiceResultClass IMPLEMENTS I_ScsmGoCBSer
verJobClass
```

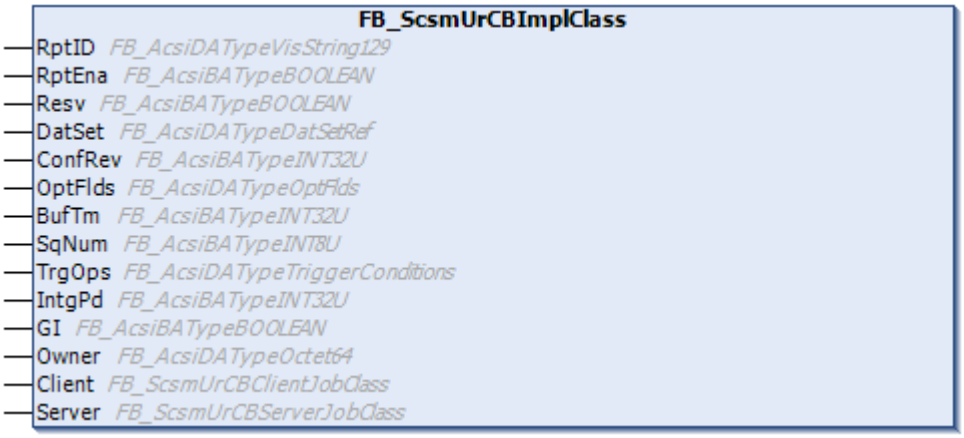
Interfaces

Type	Description
I_ScsmGoCBServerJobClass [▶ 459]	Server interface of the GOCB control block implementation.

Methods

Name	Description
Execute [▶ 459]	Runs the GOCB state machine of the server.

6.3.3.4.7 FB_ScsmUrCBImplClass



The function block FB_ScsmUrCBImplClass contains the client and server implementation of the properties and functions of a URCB control block. For each configured GOOSE control block an instance of FB_ScsmUrCBImplClass is created by the TwinCAT Telecontrol Configurator.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonUnbufferedReportControlBlockClass [▶ 160] -
> FB_AcsiUnbufferedReportControlBlock [▶ 172] -> FB_ScsmUrCBImplClass

```
FUNCTION_BLOCK FB_ScsmUrCBImplClass EXTENDS FB_AcsiUnbufferedReportControlBlock IMPLEMENTS I_ScsmUrC
BImplClass
VAR_INPUT
    Client : FB_ScsmUrCBClientJobClass;
    Server : FB_ScsmUrCBServerJobClass;
END_VAR
```

Interfaces

Type	Description
I_ScsmUrCBImplClass [▶ 460]	Interface of the URCB control block implementation.

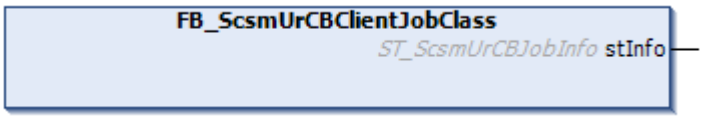
Inputs

Name	Type	Definition location	Description
Client	FB_ScsmUrCBClientJobClass [▶ 268]	Local	Client implementation of the URCB control block.
Server	FB_ScsmUrCBServerJobClass [▶ 269]	Local	Server implementation of the URCB control block.
RptID	FB_AcsiDATypeVisString129 [▶ 151]	Inherited from FB_AcsiUnbufferedReportControlBlock [▶ 172]	Data attribute, Ed1=>RptID:VisStr[65]!
RptEna	FB_AcsiBATypeBOOLEAN [▶ 103]	-//-	Data attribute
Resv	FB_AcsiBATypeBOOLEAN [▶ 103]	-//-	Data attribute
DatSet	FB_AcsiDATypeDatSetRef [▶ 117]	-//-	Data attribute
ConfRev	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
OptFlds	FB_AcsiDATypeOptFlds [▶ 128]	-//-	Data attribute
BufTm	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
SqNum	FB_AcsiBATypeINT8U [▶ 105]	-//-	Data attribute
TrgOps	FB_AcsiDATypeTriggerConditions [▶ 147]	-//-	Data attribute
IntgPd	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
GI	FB_AcsiBATypeBOOLEAN [▶ 103]	-//-	Data attribute
Owner	FB_AcsiDATypeOctet64 [▶ 127]	-//-	Data attribute, Ed2.

Properties

Name	Type	Access	Definition location	Description
bGI	BOOL	Get, Set	I_ScsmUrCBImplClass [▶ 460]	Attribute value: Enabling the general interrogation command.
bResv	BOOL	Get, Set	-//-	Attribute value: Reservation of the RCB.
bRptEna	BOOL	Get, Set	-//-	Attribute value: Enabling/disabling the RCB control block. Further configuration needed.
cOptFlds	ST_AcsiOptionalFields [▶ 536]	Get, Set	-//-	Attribute value: Configuration of the optional fields that are transmitted with a report.
cTrgOps	ST_AcsiTriggerConditions [▶ 544]	Get, Set	-//-	Attribute value: Configuration of the trigger options for sending a report.
nBufTm	DWORD	Get, Set	-//-	Attribute value: Configuration of the max. report buffer time in milliseconds.
nConfRev	DWORD	Get, Set	-//-	Attribute value: Revision number of the RCB control block configuration.
nIntgPd	DWORD	Get, Set	-//-	Attribute value: Configuration of the max. time between integrity reports in milliseconds.
nSqNum	BYTE	Get, Set	-//-	Attribute value: Report sequence number.
oOwner	T_OCTET64 [▶ 559]	Get, Set	-//-	Attribute value: Owner of the RCB. IP address of the client which has reserved the RCB for itself (offline/online reservation).
sDatSet	T_AcsiObjectReference [▶ 555]	Get, Set	-//-	Attribute value: Name of the linked data set.
sRptID	T_AcsiVisString129 [▶ 556]	Get, Set	-//-	Attribute value: Report identification string.
ipDatSet	I_AcsiCommonDataSetClass [▶ 366]	Get	-//-	Interface pointer of the linked data set.
ipClient	I_ScsmUrCBClientJobClass [▶ 460]	Get	-//-	Interface pointer of the client implementation of the URCB control block.
ipServer	I_ScsmUrCBServerJobClass [▶ 461]	Get	-//-	Interface pointer of the server implementation of the URCB control block.

6.3.3.4.8 FB_ScsmUrCBClientJobClass



Client implementation of the URCB control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

FB_AsyncServiceResultClass [▶ 299] -> FB_ScsmUrCBClientJobClass

```
FUNCTION_BLOCK FB_ScsmUrCBClientJobClass EXTENDS FB_AsyncServiceResultClass IMPLEMENTS I_ScsmUrCBCli
entJobClass
VAR_OUTPUT
    stInfo : ST_ScsmUrCBJobInfo;
END_VAR
```

Interfaces

Type	Description
I_ScsmUrCBClientJobClass [▶ 460]	Client interface of the URCB control block implementation.

Outputs

Name	Type	Description
stInfo	ST_ScsmUrCBJobInfo [▶ 554]	Information received in the last report message.

Methods

Name	Description
EnableReq [▶ 461]	Enables the control block.
DisableReq [▶ 461]	Disables the control block.
GIreq [▶ 461]	Enables the general interrogation command.

Properties

Name	Type	Access	Description
cOptFlds	ST_AcsiOptionalFields [▶ 536]	Set	Optional fields that are transferred in a report.
cTrgOps	ST_AcsiTriggerConditions [▶ 544]	Set	Trigger options for sending a report.
nBufTm	DWORD	Set	Max. report buffer time in milliseconds.
nIntgPd	DWORD	Set	Max. time between integrity reports in milliseconds.
sRptID	T_AcsiVisString129 [▶ 556]	Set	Report identification string.

6.3.3.4.9 FB_ScsmUrCBServerJobClass



Server implementation of the URCB control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

FB_AsyncServiceResultClass [▶ 299] -> FB_ScsmUrCBServerJobClass

```
FUNCTION_BLOCK FB_ScsmUrCBServerJobClass EXTENDS FB_AsyncServiceResultClass IMPLEMENTS I_ScsmUrCBSer
verJobClass
VAR_OUTPUT
    stInfo : ST_ScsmUrCBJobInfo;
END_VAR
```

Interfaces

Type	Description
I_ScsmUrCBServerJobClass [▶ 461]	Server interface of the URCB control block implementation.

Outputs

Name	Type	Description
stInfo	ST_ScsmUrCBJobInfo [▶ 554]	Information sent in the last report message.

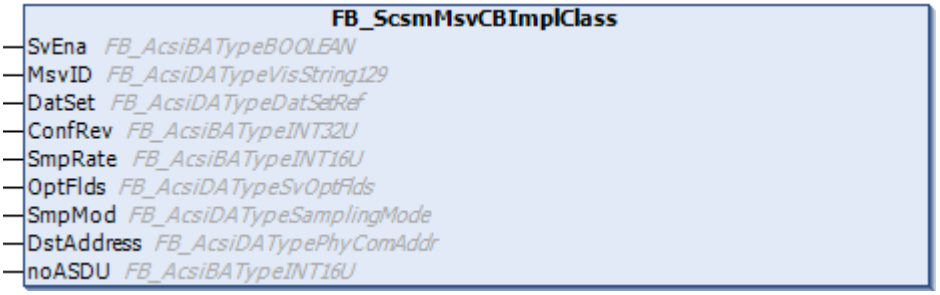
Methods

Name	Description
Execute [▶ 462]	Runs the RCB state machine of the server.

Properties

Name	Type	Access	Description
bOverwriteSameCycleChanges	BOOL	Set	If TRUE, multiple value changes within a PLC cycle and a data object are treated as single value changes (only one report is generated). If FALSE, each value change generates its own report.
bReplaceBufferedMX	BOOL	Set	Configures whether data of the functional group "MX" should be handled separately, like data of the functional group "ST".
sResvClient	STRING	Set	String with the IP address of the client preconfigured in SCL (offline reservation). Example: '192.168.10.140'. If set, then in the server access to the RCB instance is restricted to the client with this IP address ("ClientLN" entries in the SCL). Default: Empty string (online reservation). With the empty string, each client can dynamically reserve access to the RCB for itself.

6.3.3.4.10 FB_ScsmMsvCBImplClass



The function block FB_ScsmMsvCBImplClass contains the client and server implementations of the properties and functions of an MSV control block (MSV = Multicast Sampled Value).

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonMulticastSampledValueControlBlockClass [▶ 159] -> FB_AcsiMulticastSampledValueControlBlock [▶ 171] -> FB_ScsmMsvCBImplClass

```
FUNCTION_BLOCK FB_ScsmMsvCBImplClass EXTENDS FB_AcsiMulticastSampledValueControlBlock IMPLEMENTS I_S
csmMsvCBImplClass
VAR_INPUT
END_VAR
```

Interfaces

Type	Description
I_ScsmMsvCBImplClass [▶ 462]	Interface of the MSV control block implementation.

Inputs

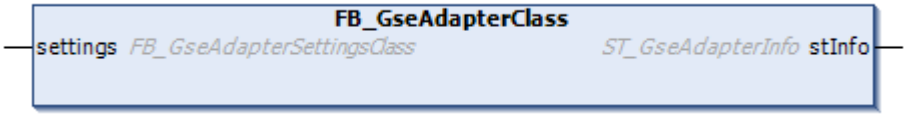
Name	Type	Definition location	Description
SvEna	FB_AcsiBATypeBOOLEAN [▶ 103]	Inherited from FB_AcsiMulticastSampledValueControlBlock [▶ 171]	Data attribute
MsvID	FB_AcsiDATypeVisString129 [▶ 151]	-//-	Data attribute
DatSet	FB_AcsiDATypeDatSetRef [▶ 117]	-//-	Data attribute
ConfRev	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
SmpRate	FB_AcsiBATypeINT16U [▶ 105]	-//-	Data attribute
OptFlds	FB_AcsiDATypeSvOptFlds [▶ 143]	-//-	Data attribute
SmpMod	FB_AcsiDATypeSamplingMode [▶ 139]	-//-	Data attribute
DstAddress	FB_AcsiDATypePhyComAddr [▶ 132]	-//-	Data attribute
NoAsdu	FB_AcsiBATypeINT16U [▶ 105]	-//-	Data attribute

Properties

Name	Type	Access	Definition location	Description
bSvEna	BOOL	Get, Set	I_ScsmMsvCBImplClass [▶ 462]	Attribute value: Enables/disables the MSV control block.
sMsvID	T_AcsiVisString129 [▶ 556]	Get, Set	-//-	Attribute value: The ID of the SV messages.
sDatSet	T_AcsiObjectReference [▶ 555]	Get, Set	-//-	Attribute value: Name of the linked data set.
nConfRev	UDINT	Get	-//-	Attribute value: Revision number of the MSV control block configuration.
nSmpRate	WORD	Get, Set	-//-	Attribute value: Sampling rate.
cOptFlds	ST_AcsiSvOptionalFields [▶ 543]	Get, Set	-//-	Attribute value: Configuration of the optional fields that are transmitted with an SV message.
eSmpMod	E_AcsiSamplingMode [▶ 503]	Get, Set	-//-	Attribute value: Sampling mode.
ePRIORITY	E_AcsiVLanPriority [▶ 517]	Get	-//-	DstAddress attribute value: Priority level in the VLAN network.
nVID	WORD(0..4095)	Get	-//-	DstAddress attribute value: ID of the VLAN network.
nAPPID	WORD	Get	-//-	DstAddress attribute value: Application Identifier.
macAddr	ETHERNET_ADDRESS [▶ 528]	Get	-//-	DstAddress attribute value: Destination MAC address.
oAddr	T_OCTET6 [▶ 558]	Get	-//-	DstAddress attribute value: Destination MAC address.
sAddr	STRING(17)	Get	-//-	DstAddress attribute value: Destination MAC address.
nNoASDU	WORD	Get, Set	-//-	Attribute value: Number of ASDUs in an SV message.
ipDatSet	I_AcsiCommonDataSetClass [▶ 366]	Get	-//-	Interface pointer to the linked data set.
ipClient	I_ScsmMsvCBClientJobClass [▶ 462]	Get	-//-	Interface pointer of the client implementation of the MSV control block.
ipServer	I_ScsmMsvCBServerJobClass [▶ 462]	Get	-//-	Interface pointer of the server implementation of the MSV control block.

6.3.3.5 GOOSE

6.3.3.5.1 FB_GseAdapterClass



The function block FB_GseAdapterClass implements interfaces for exchanging the high-priority GSE data (Global System Event) over a selected network interface. These data include GOOSE, GSSE or SV (Sampled Values) messages.
To be able to execute the communication, each instance of FB_GseAdapterClass must be connected to a corresponding network adapter. This is explained in RT Ethernet adapter configuration [▶ 35].

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

FB_ServiceErrorClass [▶ 300] -> FB_GseAdapterClass

```
FUNCTION_BLOCK FB_GseAdapterClass EXTENDS FB_ServiceErrorClass IMPLEMENTS I_GseAdapterClass, I_GseAd
apterEventSink
VAR_INPUT
    settings : FB_GseAdapterSettingsClass;
END_VAR
VAR_OUTPUT
    stInfo   : ST_GseAdapterInfo;
END_VAR
```

Interfaces

Type	Description
I_GseAdapterClass [▶ 397]	Interface for data exchange.
I_GseAdapterEventSink	

Inputs

Name	Type	Description
settings	FB_GseAdapterSettingsClass [▶ 173]	Adapter configuration settings.

Methods

Name	Description
StartPublisher [▶ 274]	Starts the publishing process on all GOCBs.
StopPublisher [▶ 274]	Stops the publishing process on all GOCBs.
Subscribe [▶ 274]	Starts the subscriber process on all GOCBs.
Unsubscribe [▶ 275]	Stops the subscriber process for all GOCBs.

Properties

Name	Type	Access	Description
ipGroup	I_AcsiCommonIEDGroupClass [▶ 368]	Set	Interface pointer from an IED group.
ipIED	I_AcsiCommonIntelligentElectronicDeviceClass [▶ 368]	Set	Interface pointer of an object with the implementation of the top-level of the IEC 61850 data model.
ipLinkStatus	I_GseLinkStatusEventSink [▶ 401]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnLinkStatusChange.
ipSettings	I_GseAdapterSettingsClass [▶ 375]	Get	Protocol settings of the GSE adapter. Returns an interface pointer to "settings".
ipSystemClock	I_GseSystemClockEventSink [▶ 402]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetSystemTime.

Outputs

Name	Type	Description
stInfo	ST_GseAdapterInfo [▶ 545]	Statistical information on GSE communication (e.g. on the number of telegrams received or sent).

6.3.3.5.1.1 StartPublisher

Starts the publishing process on all GOCBs.

```
METHOD FINAL StartPublisher : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.3.3.5.1.2 StopPublisher

Stops the publishing process on all GOCBs.

```
METHOD FINAL StopPublisher : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.3.3.5.1.3 Subscribe

Starts the subscriber process on all GOCBs.

```
METHOD FINAL Subscribe : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.3.3.5.1.4 Unsubscribe

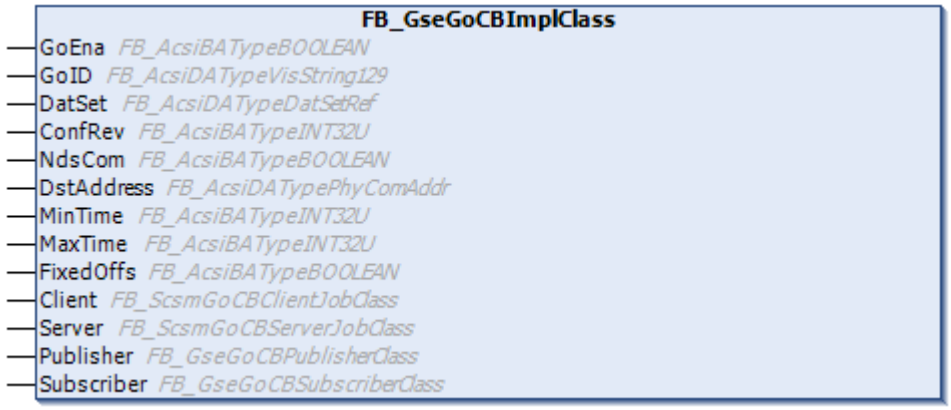
Stops the subscriber process for all GOCBs.

```
METHOD FINAL Unsubscribe : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.3.3.5.2 FB_GseGoCBImplClass



The function block FB_GseGoCBImplClass contains the implementation of the properties of a GOOSE control block as well as the functions for Publisher and Subscriber. For each configured GOOSE control block an instance of FB_GseGoCBImplClass is created by the TwinCAT Telecontrol Configurator. For more information on code generation for GOOSE, see Automatic code generation [▶ 80].

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonGooseControlBlockClass [▶ 155] -
> FB_AcsiGooseControlBlock [▶ 171] -> FB_ScsmGoCBImplClass [▶ 262] -> FB_GseGoCBImplClass

```
FUNCTION_BLOCK FB_GseGoCBImplClass EXTENDS FB_ScsmGoCBImplClass IMPLEMENTS I_GseGoCBImplClass
VAR_INPUT
    Publisher  : FB_GseGoCBPublisherClass;
    Subscriber : FB_GseGoCBSubscriberClass;
END_VAR
```

Interfaces

Type	Description
I_GseGoCBImplClass [▶ 398]	Interface of the GOOSE control block implementation.

Inputs

Name	Type	Definition location	Description
Publisher	FB_GseGoCBPublisherClass [▶ 276]	Local	Publisher implementation of the GOOSE control block.
Subscriber	FB_GseGoCBSubscriberClass [▶ 277]	Local	Subscriber implementation of the GOOSE control block.
Client	FB_ScsmGoCBClientJobClass [▶ 264]	Inherited from FB_ScsmGoCBImplClass [▶ 262]	Client implementation of the GOOSE control block.
Server	FB_ScsmGoCBServerJobClass [▶ 265]	-//-	Server implementation of the GOOSE control block.
GoEna	FB_AcsiBATypeBOOLEAN [▶ 103]	Inherited from FB_AcsiGooseControlBlock [▶ 171]	Data attribute
GoID	FB_AcsiDATypeVisString129 [▶ 151]	-//-	Data attribute
DatSet	FB_AcsiDATypeDatSetRef [▶ 117]	-//-	Data attribute
ConfRev	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
NdsCom	FB_AcsiBATypeBOOLEAN [▶ 103]	-//-	Data attribute
DstAddress	FB_AcsiDATypePhyComAddr [▶ 132]	-//-	Data attribute
MinTime	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
MaxTime	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
FixedOffs	FB_AcsiBATypeBOOLEAN [▶ 103]	-//-	Data attribute

Example

For more information on how to use the GOOSE control block, see: Client - GOOSE Subscriber (without Client-Server communication) [▶ 693] and Server - GOOSE Publisher (without Client-Server communication) [▶ 704].

6.3.3.5.3 FB_GseGoCBPublisherClass



Publisher implementation of the GOOSE control block.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

FB_ServiceErrorClass [▶ 300] -> FB_GseGoCBPublisherClass

```
FUNCTION_BLOCK FB_GseGoCBPublisherClass EXTENDS FB_ServiceErrorClass IMPLEMENTS I_GseGoCBPublisherCl
ass
VAR_OUTPUT
    stTx : ST_GseGoCBPublisherTx;
END_VAR
```

Interfaces

Type	Description
I_GseGoCBPublisherClass [▶ 398]	Interface of the GOOSE control block implementation.

Outputs

Name	Type	Description
stTx	ST_GseGoCBPublisherTx [▶ 546]	Information sent in the last GOOSE message.

Methods

Name	Description
Execute [▶ 399]	Executes the Publisher state machine.
Start [▶ 399]	Starts sending GOOSE messages.
Stop [▶ 399]	Stops sending GOOSE messages.
Update [▶ 399]	Triggers an immediate sending of a GOOSE message.

Properties

Name	Type	Access	Description
bTest	BOOL	Set	Simulation/test flag of GOOSE messages. At value TRUE the simulation and test flag is set in the GOOSE message.
eStrategy	E_GseRetransmissionStrategy [▶ 522]	Set	Configures the algorithm for the repeated GOOSE messages.
nMultiplier	BYTE(1..16#FF)	Set	Configures the multiplier for the time interval of repeated GOOSE messages. Default value: Param_Acsi.cDefault_GoCB_Multiplier (2).
ipAdapter	I_GseAdapterClass [▶ 397]	Set	Interface of the GOOSE network adapter.

6.3.3.5.4 FB_GseGoCBSubscriberClass



Subscriber implementation of the GOOSE control block.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

FB_ServiceErrorClass [▶ 300] -> FB_GseGoCBSubscriberClass

```
FUNCTION_BLOCK FB_GseGoCBSubscriberClass EXTENDS FB_ServiceErrorClass IMPLEMENTS I_GseGoCBSubscriber
Class
VAR_OUTPUT
    stRx : ST_GseGoCBSubscriberRx;
END_VAR
```

Interfaces

Type	Description
I_GseGoCBSubscriberClass [▶ 400]	Interface of the GOOSE control block implementation.

Outputs

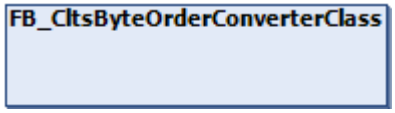
Name	Type	Description
stRx	ST_GseGoCBSubscriberRx [▶ 547]	Information received in the last GOOSE message.

Methods

Name	Description
Execute [▶ 400]	Executes the Subscriber state machine.
Enable [▶ 400]	Starts receiving GOOSE messages.
Disable [▶ 401]	Stops receiving GOOSE messages.

6.3.3.6 Sampled Values

6.3.3.6.1 FB_CltsByteOrderConverterClass



The natural byte order of the data on a TwinCAT system is referred to as the host byte order. The byte order of the data transmitted over the network is referred to as the network byte order. The network byte order of IEC 61850 data largely corresponds to the big-endian format, while the TwinCAT system uses the byte order of the little-endian format. This function block implements methods that allow the byte order of the IEC 61850 data to be changed (little-endian <-> big-endian).

The HostToNetXxx methods convert (encode) the data values to be sent from host byte order to network byte order. After encoding, the data values are stored in the network transmission buffer of the sender (e.g., SV-Publisher). The transmitter sends the contents of the network transmission buffer.

The receiver (e.g., SV subscriber) receives the data values into the network receive buffer. The NetToHostXxx methods convert (decode) the data values in the network receive buffer from network byte order to host byte order.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION_BLOCK FB_CltsByteOrderConverterClass
```

Methods

Name	Description
HostToNetBoolean	Encodes values of type BOOL (IEC 61850 type BOOLEAN) into the network transmission buffer (8 bits as a byte; all bits set to 0 => FALSE, otherwise TRUE).
HostToNetEntryID	Encodes values of type T_OCTET8 [▶ 558] (IEC 61850 EntryID type) into the network transmission buffer (byte array with 8 elements).
HostToNetEntryTime	Encodes values of type T_BinaryTime [▶ 556] (IEC 61850 type EntryTime) into the network transmission buffer (48-bit timestamp as a byte array with 6 elements).
HostToNetFloat32 [▶ 281]	Encodes REAL-type values (IEC 61850 type FLOAT32) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNetFloat64	Encodes values of type LREAL (IEC 61850 type FLOAT64) into the network transmission buffer (64 bits as a byte array with 8 elements).
HostToNetInt16	Encodes values of type INT (IEC 61850 type INT16) into the network transmission buffer (16 bits as a byte array with 2 elements).
HostToNetInt16U	Encodes values of type UINT (IEC 61850 type INT16U) into the network transmission buffer (16 bits as a byte array with 2 elements).
HostToNetInt24	Encodes values of type T_INT24 [▶ 557] (IEC 61850 type INT24) into the network transmission buffer (24 bits as a byte array with 3 elements).
HostToNetInt24U	Encodes values of type T_UINT24 [▶ 559] (IEC 61850 type INT24U) into the network transmission buffer (24 bits as a byte array with 3 elements).
HostToNetInt32 [▶ 281]	Encodes values of type DINT (IEC 61850 type INT32) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNetInt32U	Encodes values of type UDINT (IEC 61850 type INT32U) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNetInt64	Encodes values of type LINT (IEC 61850 type INT64) into the network transmission buffer (64 bits as a byte array with 8 elements).
HostToNetInt64U	Encodes values of type ULINT (IEC 61850 type INT64U) into the network transmission buffer (64 bits as a byte array with 8 elements).
HostToNetInt8	Encodes values of type SINT (IEC 61850 type INT8) into the network transmission buffer (8 bits as a byte).
HostToNetInt8U	Encodes values of type USINT (IEC 61850 type INT8U) into the network transmission buffer (8 bits as a byte).
HostToNetTimestamp	Encodes values of type T_UtcTime [▶ 560] (IEC 61850 timestamp type) into the network transmission buffer (64-bit timestamp as an 8-element byte array).
NetToHostBoolean	Decodes BOOL values (IEC 61850 type BOOLEAN) from the network receive buffer (8 bits as a byte; all bits set to 0 => FALSE, otherwise TRUE).
NetToHostEntryID	Decodes values of type T_OCTET8 [▶ 558] (IEC 61850 EntryID type) from the network receive buffer (byte array with 8 elements).
NetToHostEntryTime	Decodes values of type T_BinaryTime [▶ 556] (IEC 61850 type EntryTime) from the network receive buffer (48-bit timestamp as a byte array with 6 elements).
NetToHostFloat32 [▶ 281]	Decodes REAL-type values (IEC 61850 type FLOAT32) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostFloat64	Decodes values of type LREAL (IEC 61850 type FLOAT64) from the network receive buffer (64 bits as an 8-element byte array).
NetToHostInt16	Decodes INT-type values (IEC 61850 type INT16) from the network receive buffer (16 bits as a byte array with 2 elements).
NetToHostInt16U	Decodes values of type UINT (IEC 61850 type INT16U) from the network receive buffer (16 bits as a byte array with 2 elements).
NetToHostInt24	Decodes values of type T_INT24 [▶ 557] (IEC 61850 type INT24) from the network receive buffer (24 bits as a byte array with 3 elements).
NetToHostInt24U	Decodes values of type T_UINT24 [▶ 559] (IEC 61850 type INT24U) from the network receive buffer (24 bits as a byte array with 3 elements).
NetToHostInt32 [▶ 281]	Decodes values of type DINT (IEC 61850 type INT32) from the network receive buffer (32 bits as a byte array with 4 elements).

Name	Description
NetToHostInt32U	Decodes values of type UDINT (IEC 61850 type INT32U) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostInt64	Decodes values of type LINT (IEC 61850 type INT64) from the network receive buffer (64 bits as an 8-element byte array).
NetToHostInt64U	Decodes ULINT-type values (IEC 61850 type INT64U) from the network receive buffer (64 bits as an 8-element byte array).
NetToHostInt8	Decodes SINT-type values (IEC 61850 type INT8) from the network receive buffer (8 bits as a byte).
NetToHostInt8U	Decodes values of type USINT (IEC 61850 type INT8U) from the network receive buffer (8 bits as a byte).
NetToHostTimestamp	Decodes values of type T_UtcTime [▶ 560] (IEC 61850 timestamp type) from the network receive buffer (64-bit timestamp as an 8-element byte array).

Example:

See here: SV Publisher/Subscriber according to UCA 61850-9-2 Light Edition [▶ 760].

6.3.3.6.1.1 HostToNetFloat32

Encodes REAL-type values into the T_OCTET4-type [▶ 558] network transmission buffer.

```
METHOD FINAL HostToNetFloat32
VAR_IN_OUT
    in  : REAL;
    out : T_OCTET4;
END_VAR
```

6.3.3.6.1.2 HostToNetInt32

Encodes values of type DINT into the network transmission buffer of type T_OCTET4 [▶ 558].

```
METHOD FINAL HostToNetInt32
VAR_IN_OUT
    in  : DINT;
    out : T_OCTET4;
END_VAR
```

6.3.3.6.1.3 NetToHostFloat32

Decodes REAL-type values from the network receive buffer of type T_OCTET4 [▶ 558].

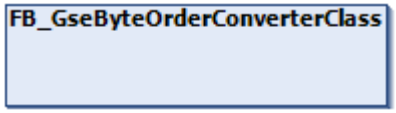
```
METHOD FINAL NetToHostFloat32
VAR_IN_OUT
    in  : T_OCTET4;
    out : REAL;
END_VAR
```

6.3.3.6.1.4 NetToHostInt32

Decodes values of type DINT from the network receive buffer of type T_OCTET4 [▶ 558].

```
METHOD FINAL NetToHostInt32
VAR_IN_OUT
    in  : T_OCTET4;
    out : DINT;
END_VAR
```

6.3.3.6.2 FB_GseByteOrderConverterClass



The natural byte order of the data on a TwinCAT system is referred to as the host byte order. The byte order of the data transmitted over the network is referred to as the network byte order. The network byte order of IEC 61850 data largely corresponds to the big-endian format, while the TwinCAT system uses the byte order of the little-endian format. This function block implements methods that allow the byte order of the IEC 61850 data to be changed (little-endian <-> big-endian).

The HostToNetXxx methods convert (encode) the data values to be sent from host byte order to network byte order. After encoding, the data values are stored in the network transmission buffer of the sender (e.g., SV-Publisher). The transmitter sends the contents of the network transmission buffer.

The receiver (e.g., SV subscriber) receives the data values into the network receive buffer. The NetToHostXxx methods convert (decode) the data values in the network receive buffer from network byte order to host byte order.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

FB_CltsByteOrderConverterClass [▶ 278] -> FB_GseByteOrderConverterClass

```
FUNCTION_BLOCK FB_GseByteOrderConverterClass EXTENDS FB_CltsByteOrderConverterClass
```

Methods

Name	Definition location	Description
HostToNet Boolean	Inherited from FB_CltsByteOrderConverterClass [▶ 278]	Encodes values of type BOOL (IEC 61850 type BOOLEAN) into the network transmission buffer (8 bits as a byte; all bits set to 0 => FALSE, otherwise TRUE).
HostToNet Float32 [▶ 281]	-//-	Encodes REAL-type values (IEC 61850 type FLOAT32) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNet Int32 [▶ 281]	-//-	Encodes values of type DINT (IEC 61850 type INT32) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNet Xxx	-//-	Additional methods for encoding IEC 61850 base-type values in the network transmission buffers.
HostToNet Check	Local	Encodes values of type E_AcsiCheck [▶ 479] (IEC 61850 type CheckConditions) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNet Currency	-//-	Encodes values of type T_AcsiCurrency [▶ 555] (IEC 61850 type Currency, null-terminated ISO 4217 currency code as text) into the network transmission buffer (byte array with 4 elements).
HostToNet Dbpos	-//-	Encodes values of type E_AcsiDbpos [▶ 484] (IEC 61850 type Dbpos) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNet Enum	-//-	Encodes values of type SINT (IEC 61850 enumerated type) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNet Enum16	-//-	Encodes values of type INT (IEC 61850 enumerated type) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNet Enum32	-//-	Encodes values of type DINT (IEC 61850 enumerated type) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNet LogOptFlds	-//-	Encodes values of type ST_AcsiLogOptionalFields [▶ 536] (IEC 61850 type LCBLogEntryOptions) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNet ObjName	-//-	Encodes values of type T_AcsiObjectName [▶ 555] (IEC 61850 ObjectName type) into the network transmission buffer (null-terminated ASCII text as a byte array with 20 elements).
HostToNet ObjRef	-//-	Encodes values of type T_AcsiObjectReference [▶ 555] (IEC 61850 type ObjectReference) into the network transmission buffer (null-terminated ASCII text as a byte array with 20 elements).
HostToNet Octet64	-//-	Encodes values of type T_OCTET64 [▶ 559] (IEC 61850 type Octet64) into the network transmission buffer (null-terminated ASCII text as a byte array with 20 elements).
HostToNet OctetString	-//-	Encodes values of type ARRAY[*] OF BYTE (IEC 61850 type Octet[n]) into the network transmission buffer (null-terminated ASCII text as a byte array with 20 elements).
HostToNet OptFlds	-//-	Encodes values of type ST_AcsiOptionalFields [▶ 536] (IEC 61850 type RCBReportOptions) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNet Quality [▶ 286]	-//-	Encodes values of type ST_AcsiQuality [▶ 541] (IEC 61850 type Quality) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNet ReasonCode	-//-	Encodes values of type ST_AcsiReasonCode [▶ 542] (IEC 61850 ReasonCode type) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNet SvOptFlds	-//-	Encodes values of type ST_AcsiSvOptionalFields [▶ 543] (IEC 61850 type SVMessageOptions) into the network transmission buffer (32 bits as a byte array with 4 elements).

Name	Definition location	Description
HostToNetTcmd	-//-	Encodes values of type E_AcsiTcmd [▶ 513] (IEC 61850 type Tcmd) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNetTrgOps	-//-	Encodes values of type ST_AcsiTriggerConditions [▶ 544] (IEC 61850 type TriggerConditions) into the network transmission buffer (32 bits as a byte array with 4 elements).
HostToNetUnicode255	-//-	Encodes values of type T_AcsiUnicode255 [▶ 556] (IEC 61850 type Unicode255) into the network transmission buffer (null-terminated UTF-8-encoded text as a byte array with 20 elements).
HostToNetVisString129	-//-	Encodes values of type T_AcsiVisString129 [▶ 556] (IEC 61850 type VisString129) into the network transmission buffer (null-terminated ASCII text as a byte array with 35 elements).
HostToNetVisString255	-//-	Encodes values of type T_AcsiVisString255 [▶ 556] (IEC 61850 type VisString255) into the network transmission buffer (null-terminated ASCII text as a byte array with 35 elements).
HostToNetVisString64	-//-	Encodes values of type T_AcsiVisString64 [▶ 556] (IEC 61850 type VisString64) into the network transmission buffer (null-terminated ASCII text as a byte array with 35 elements).
NetToHostBoolean	Inherited from FB_CltsByteOrderConverterClass [▶ 278]	Decodes BOOL values (IEC 61850 type BOOLEAN) from the network receive buffer (8 bits as a byte; all bits set to 0 => FALSE, otherwise TRUE).
NetToHostFloat32 [▶ 281]	-//-	Decodes REAL-type values (IEC 61850 type FLOAT32) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostInt32 [▶ 281]	-//-	Decodes values of type DINT (IEC 61850 type INT32) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostXxx	-//-	Additional methods for decoding IEC 61850 base-type values from the network receive buffer.
NetToHostCheck	Local	Decodes values of type E_AcsiCheck [▶ 479] (IEC 61850 type CheckConditions) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostCurrency	-//-	Decodes values of type T_AcsiCurrency [▶ 555] (IEC 61850 type Currency, null-terminated ISO 4217 currency code as text) from the network receive buffer (byte array with 4 elements).
NetToHostDbpos	-//-	Decodes values of type E_AcsiDbpos [▶ 484] (IEC 61850 type Dbpos) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostEnum	-//-	Decodes SINT-type values (IEC 61850 enumerated type) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostEnum16	-//-	Decodes INT-type values (IEC 61850 enumerated type) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostEnum32	-//-	Decodes values of type DINT (IEC 61850 enumerated type) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostLogOptFlds	-//-	Decodes values of type ST_AcsiLogOptionalFields [▶ 536] (IEC 61850 type LCBLogEntryOptions) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostObjName	-//-	Decodes values of type T_AcsiObjectName [▶ 555] (IEC 61850 ObjectName type) from the network receive buffer (null-terminated ASCII text as a byte array with 20 elements).
NetToHostObjRef	-//-	Decodes values of type T_AcsiObjectReference [▶ 555] (IEC 61850 type ObjectReference) from the network receive buffer (null-terminated ASCII text as a byte array with 20 elements).

Name	Definition location	Description
NetToHostOctet64	-//-	Decodes values of type T_OCTET64 [▶ 559] (IEC 61850 Octet64 type) from the network receive buffer (null-terminated ASCII text as a byte array with 20 elements).
NetToHostOctetString	-//-	Decodes values of type ARRAY[*] OF BYTE (IEC 61850 type Octet[n]) from the network receive buffer (null-terminated ASCII text as a byte array with 20 elements).
NetToHostOptFlds	-//-	Decodes values of type ST_AcsiOptionalFields [▶ 536] (IEC 61850 type RCBReportOptions) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostQuality [▶ 286]	-//-	Decodes values of type ST_AcsiQuality [▶ 541] (IEC 61850 type Quality) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostReasonCode	-//-	Decodes values of type ST_AcsiReasonCode [▶ 542] (IEC 61850 ReasonCode type) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostSvOptFlds	-//-	Decodes values of type ST_AcsiSvOptionalFields [▶ 543] (IEC 61850 type SVMessageOptions) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostTcmd	-//-	Decodes values of type E_AcsiTcmd [▶ 513] (IEC 61850 type Tcmd) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostTrgOps	-//-	Decodes values of type ST_AcsiTriggerConditions [▶ 544] (IEC 61850 type TriggerConditions) from the network receive buffer (32 bits as a byte array with 4 elements).
NetToHostUnicode255	-//-	Decodes values of type T_AcsiUnicode255 [▶ 556] (IEC 61850 type Unicode255) from the network receive buffer (null-terminated UTF-8-encoded text as a byte array with 20 elements).
NetToHostVisString129	-//-	Decodes values of type T_AcsiVisString129 [▶ 556] (IEC 61850 type VisString129) from the network receive buffer (null-terminated ASCII text as a byte array with 35 elements).
NetToHostVisString255	-//-	Decodes values of type T_AcsiVisString255 [▶ 556] (IEC 61850 type VisString255) from the network receive buffer (null-terminated ASCII text as a byte array with 35 elements).
NetToHostVisString64	-//-	Decodes values of type T_AcsiVisString64 [▶ 556] (IEC 61850 type VisString64) from the network receive buffer (null-terminated ASCII text as a byte array with 35 elements).

Example:

See here: SV Publisher/Subscriber according to UCA 61850-9-2 Light Edition [▶ 760].

6.3.3.6.2.1 HostToNetQuality

Encodes values of type ST_AcsiQuality [▶ 541] into the network transmission buffer of type T_OCTET4 [▶ 558].

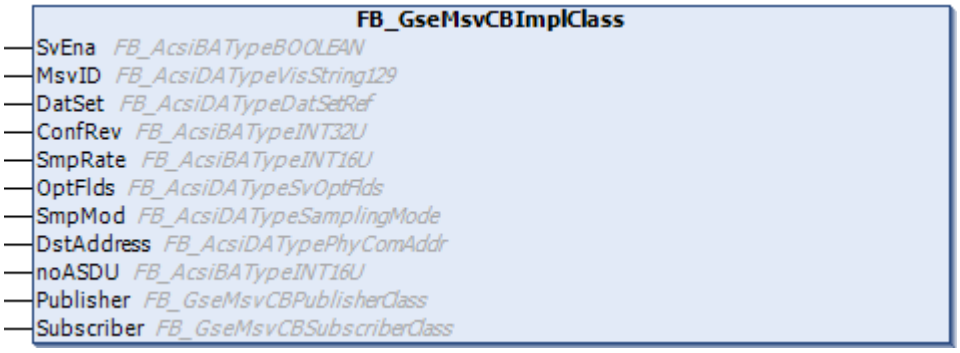
```
METHOD FINAL HostToNetQuality
VAR_IN_OUT
    in  : ST_AcsiQuality;
    out : T_OCTET4;
END_VAR
```

6.3.3.6.2.2 NetToHostQuality

Decodes values of type ST_AcsiQuality [▶ 541] from the network receive buffer of type T_OCTET4 [▶ 558].

```
METHOD FINAL NetToHostQuality
VAR_IN_OUT
    in  : T_OCTET4;
    out : ST_AcsiQuality;
END_VAR
```

6.3.3.6.3 FB_GseMsvCBImplClass



The FB_GseMsvCBImplClass function block contains the implementation of the properties of an SV control block (Sampled Values Control Block) as well as the functions for Publisher and Subscriber. For each configured SV control block an instance of FB_GseMsvCBImplClass is created by the TwinCAT Telecontrol Configurator. For more information on code generation for SV, see Automatic code generation [▶ 80].

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

FB_AcsiCommonNodeClass [▶ 160] -> FB_AcsiCommonDataClass [▶ 154] -
> FB_AcsiCommonControlBlockClass [▶ 153] -> FB_AcsiCommonMulticastSampledValueControlBlockClass [▶ 159] -> FB_AcsiMulticastSampledValueControlBlock [▶ 171] -> FB_ScsmMsvCBImplClass [▶ 270] -
> FB_GseMsvCBImplClass

```
FUNCTION_BLOCK FB_GseMsvCBImplClass EXTENDS FB_ScsmMsvCBImplClass IMPLEMENTS I_GseMsvCBImplClass
VAR_INPUT
    Publisher  : FB_GseMsvCBPublisherClass;
    Subscriber : FB_GseMsvCBSubscriberClass;
END_VAR
```

Interfaces

Type	Description
I_GseMsvCBImplClass [▶ 402]	Interface of the SV control block implementation.

Inputs

Name	Type	Definition location	Description
Publisher	FB_GseMsvCBPublisherClass [▶ 288]	Local	Publisher implementation of the SV control block.
Subscriber	FB_GseMsvCBSubscriberClass [▶ 289]	Local	Subscriber implementation of the SV control block.
Client	FB_ScsmMsvCBClientJobClass	Inherited from FB_ScsmMsvCBImplClass [▶ 270]	Client implementation of the SV control block.
Server	FB_ScsmMsvCBServerJobClass	-//-	Server implementation of the SV control block.
SvEna	FB_AcsiBATypeBOOLEAN [▶ 103]	Inherited from FB_AcsiMulticastSampledValueControlBlock [▶ 171]	Data attribute
MsvID	FB_AcsiDATypeVisString129 [▶ 151]	-//-	Data attribute
DatSet	FB_AcsiDATypeDatSetRef [▶ 117]	-//-	Data attribute
ConfRev	FB_AcsiBATypeINT32U [▶ 106]	-//-	Data attribute
SmpRate	FB_AcsiBATypeINT16U [▶ 105]	-//-	Data attribute
OptFlds	FB_AcsiDATypeSvOptFlds [▶ 143]	-//-	Data attribute
SmpMod	FB_AcsiDATypeSamplingMode [▶ 139]	-//-	Data attribute
DstAddress	FB_AcsiDATypePhyComAddr [▶ 132]	-//-	Data attribute
noASDU	FB_AcsiBATypeINT16U [▶ 105]	-//-	Data attribute

Sample

For more information on using the SV control block, see: SV-Publisher/Subscriber according to UCA 61850-9-2 Light Edition - Transmission of sampled measured values [▶ 760].

6.3.3.6.4 FB_GseMsvCBPublisherClass



Publisher implementation of the SV control block.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

FB_ServiceErrorClass [▶ 300] -> FB_GseMsvCBPublisherClass

```
FUNCTION_BLOCK FB_GseMsvCBPublisherClass EXTENDS FB_ServiceErrorClass IMPLEMENTS I_GseMsvCBPublisher
Class
VAR_OUTPUT
    stTx : ST_GseMsvCBPublisherTx;
END_VAR
```

Interfaces

Type	Description
I_GseMsvCBPublisherClass [▶ 403]	Interface of the SV control block implementation.

Outputs

Name	Type	Description
stTx	ST_GseMsvCBPublisherTx [▶ 548]	Information sent in the last SV message.

Methods

Name	Description
Execute [▶ 403]	Executes the Publisher state machine.
Start [▶ 403]	Starts sending SV messages.
Stop [▶ 404]	Stops sending SV messages.
Update [▶ 404]	Triggers immediate transmission of an SV message.

Properties

Name	Type	Access	Description
ipAdapter	I_GseAdapterClass [▶ 397]	Set	Interface of the SV network adapter.

6.3.3.6.5 FB_GseMsvCBSubscriberClass



Subscriber implementation of the SV control block.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

FB_ServiceErrorClass [▶ 300] -> FB_GseMsvCBSubscriberClass

```
FUNCTION_BLOCK FB_GseMsvCBSubscriberClass EXTENDS FB_ServiceErrorClass IMPLEMENTS I_GseMsvCBSubscrib
erClass
VAR_OUTPUT
    stRx : ST_GseMsvCBSubscriberRx;
END_VAR
```

Interfaces

Type	Description
I_GseMsvCBSubscriberClass [▶ 404]	Interface of the SV control block implementation.

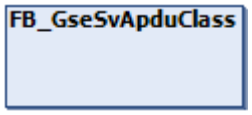
Outputs

Name	Type	Description
stRx	ST_GseMsvCBSubscriberRx [▶ 549]	Information received in the last SV message.

Methods

Name	Description
Execute [▶ 404]	Executes the Subscriber state machine.
Enable [▶ 405]	Starts receiving SV messages.
Disable [▶ 405]	Stops receiving SV messages.

6.3.3.6.6 FB_GseSvApduClass



Object instances of FB_GseSvApduClass implement functionalities of an SV-APDU.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

FB_CltsByteOrderConverterClass [▶ 278] -> FB_GseByteOrderConverterClass [▶ 281] -> FB_GseSvApduClass

```
FUNCTION_BLOCK FB_GseSvApduClass EXTENDS FB_GseByteOrderConverterClass IMPLEMENTS I_GseSvApduClass,
I_GseSvPublisherEventSink, I_GseSvSubscriberEventSink
```

Interfaces

Type	Description
I_GseSvApduClass [▶ 405]	Interface of the SV-APDU implementation.
I_GseSvPublisherEventSink [▶ 407]	Interface of the SV publisher event routine.
I_GseSvSubscriberEventSink [▶ 408]	Interface of the SV subscriber event routine.

Methods

Name	Description
AddAsduToContainer [▶ 291]	Adds an ASDU object to the container of ASDU objects.
Equals [▶ 291]	Compares itself and another object for equality.

Methods of event handling (callback methods)

Name	Description
OnSvPublisherHostToNet [▶ 291]	A custom event handling routine that converts SV Publisher data from host-byte order to network-byte order for transmission.
OnSvSubscriberNetToHost [▶ 291]	A user-defined event handling routine that converts the received SV subscriber data from network byte order to host byte order.

Properties

Name	Type	Access	Definition location	Initial value	Description
APPID	WORD	Get	Local	0	APPID in the GSE header of the SV message.
bSimulation	BOOL	Get, Set	-//-	FALSE	Simulation bit in the Ethernet frame header of the SV message. If the “bSimulation” adapter setting is set to FALSE, the simulation bit can be set individually for each APDU that is sent.
srcAddr	ETHERNET_ADDRESS [▶ 528]	Get	-//-	[0,0,0,0,0,0]	Source address in the Ethernet frame header of the SV message.
ipUpdate	I_GseSvUpdateEventSink [▶ 408]	Set	-//-	0	Used only by SV subscribers. Interface pointer of an object with a user-defined implementation of the event handling routine: OnSvUpdate [▶ 408].

6.3.3.6.6.1 OnSvPublisherHostToNet

In this method, the SV publisher data is converted from the host byte order (little-endian) to the network byte order (big-endian) for transmission.

```
METHOD OnSvPublisherHostToNet
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    ipApdu    : I_GseSvApduClass;
END_VAR
```

6.3.3.6.6.2 OnSvSubscriberNetToHost

In this method, the received SV subscriber data is converted from network byte order (big-endian) to host byte order (little-endian).

```
METHOD OnSvSubscriberNetToHost
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    ipApdu    : I_GseSvApduClass;
END_VAR
```

6.3.3.6.6.3 AddAsduToContainer

Adds an ASDU object to the container of ASDU objects.

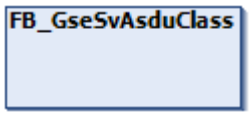
```
METHOD FINAL AddAsduToContainer : I_GseSvApduClass
VAR_INPUT
    ipAsdu   : I_GseSvAsduClass;
END_VAR
VAR_OUTPUT
    bMatched : BOOL;
END_VAR
```

6.3.3.6.6.4 Equals

Compares itself and another object for equality. If the values are equal, the return value is TRUE.

```
METHOD FINAL Equals : BOOL
VAR_INPUT
    ipOther : I_GseSvApduClass;
END_VAR
```

6.3.3.6.7 FB_GseSvAsduClass



Object instances of FB_GseSvAsduClass implement functionalities of an SV-ASDU.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

FB_CltsByteOrderConverterClass [▶ 278] -> FB_GseByteOrderConverterClass [▶ 281] -> FB_GseSvAsduClass

```
FUNCTION_BLOCK FB_GseSvAsduClass EXTENDS FB_GseByteOrderConverterClass IMPLEMENTS I_GseSvAsduClass
```

Interfaces

Type	Description
I_GseSvAsduClass [▶ 406]	Interface of the SV-ASDU implementation.

Methods

Name	Description
AddValuesToContainer [▶ 293]	Adds an SV data object to the container of SV data objects.

Properties

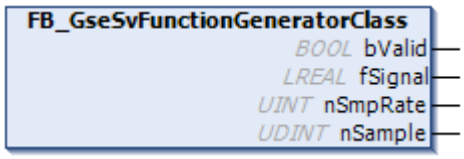
Name	Type	Access	Definition location	Description
eSmpMod	E_AcsiSamplingMode [▶ 503]	Get	Local	Sampling mode.
eSmpSynch	E_GseSvSmpSynch [▶ 523]	Get, Set	-//-	Synchronization mode of the sampled values.
hUser	PVOID	Get, Set	-//-	User-defined handle (also pointer).
index	UDINT	Get, Set	-//-	Index of the ASDU object (for several ASDUs in one SV message).
ipDataSet	I_AcsiCommonDataSetClass [▶ 366]	Get	-//-	Interface to the SV data set.
nConfRev	UDINT	Get	-//-	Version info of the SV configuration (always 1 for SV).
nSmpCnt	UINT	Get, Set	-//-	Sampling value counter.
nSmpRate	UINT	Get	-//-	Sampling rate.
nSmpSynch	USINT	Get, Set	-//-	Synchronization mode of the sampled values.
oSynchSrcID	T_OCTET8 [▶ 558]	Get, Set	-//-	Optional parameter value: “gmIdentity”.
sDatSet	T_AcsiObjectReference [▶ 555]	Get	-//-	SV data set reference string.
sSvID	T_AcsiVisString129 [▶ 556]	Get	-//-	SV identification string.
tRefrTm	T_UtcTime [▶ 560]	Get, Set	-//-	SV refresh time.

6.3.3.6.7.1 AddValuesToContainer

Adds an SV data object to the container of SV data objects.

```
METHOD FINAL AddValuesToContainer : BOOL
VAR_INPUT
    pValues  : PVOID;
    cbValues : UDINT;
END_VAR
```

6.3.3.6.8 FB_GseSvFunctionGeneratorClass



This function block implements a simple sine function test signal generator. The generated sample values can be used to simulate SV communication.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
FUNCTION_BLOCK FB_GseSvFunctionGeneratorClass
VAR_OUTPUT
    bValid   : BOOL;
    fSignal  : LREAL;
    nSmpRate : UINT;
    nSample  : UDINT;
END_VAR
```

Outputs

Name	Type	Description
bValid	BOOL	This output is TRUE if the signal generator configuration is valid, and FALSE if one or more parameter values are invalid. Examples of invalid parameter values: nOversampleFactor = 0, fFrequency <= 0, nSpecifiedSmpRate = 0, or fCycleTimeCorrectionFactor <= 0.
fSignal	LREAL	Returns the most recent signal sample value of the sine function generated by the method call.
nSmpRate	UINT	Effective sampling rate. In sampling mode: 'AtSmpRateProPeriod' is the specified number of samples per period. In sampling mode: 'AtSmpRateProSecond' is the specified number of samples per second. In sampling mode: 'AtPlcTaskCycleTime', this parameter value is not used, and the effective sampling rate is calculated based on the PLC task cycle time, the frequency, and the configured oversampling factor.
nSample	UDINT	Sample value number. The value increments on each method call. The first sample has the number 0, and the last sample has the number nSmpRate – 1. When the nSmpRate call is made, the sample number overflows and returns to 0.

Methods

Name	Description
Call [▶ 295]	The method generates a new signal sample. In the signal sampling mode 'AtPlcTaskCycleTime', this method is normally called nOversampleFactor times in each PLC task cycle.
Config [▶ 295]	This method configures the function generator based on the selected signal sampling mode (eMode), the PLC cycle time, the PLC cycle time correction factor (fCycleTimeCorrectionFactor), the configured oversampling factor (nOversampleFactor), the configured sampling rate (nSpecifiedSmpRate), and the signal frequency (fFrequency). The Config method is automatically called internally whenever these configuration parameters are changed.
Reset [▶ 296]	The method resets the initial value fSignal to 0 and the sample number nSample to 0. After this method call, the next call to the method generates the first sample value with the number 0 again. If the configuration parameters are changed, the reset method is automatically called internally by the config method.

Properties

Name	Type	Access	Description
eMode	E_GseSvGeneratorMode [▶ 523]	Set	Signal sampling mode. Default value: 'AtPlcTaskCycleTime' => Sampling within the PLC task cycle.
fAmplitude	LREAL	Set	Sets the peak amplitude of the signal. For example, in V or A. Default value: 230 V * √2 => ~ 325 V.
fCycleTimeCorrectionFactor	LREAL	Set	Sets the correction factor for the PLC task cycle time. Default value: 1.0 (no correction). Values <= 0 are invalid.
fFrequency	LREAL	Set	Sets the signal frequency in Hz. Default value: 50.0 Hz. Values <= 0 are invalid.
fOffset	LREAL	Set	Specifies the signal offset. Default value: 0 (no offset).
fPhi	LREAL	Set	Sets the signal phase shift in degrees. Default value: 0°.
nOversampleFactor	UINT(1..128)	Set	Sets the oversampling factor of the signal generator. The value 0 is invalid. Default value: 1 (no oversampling). In the UCA 61850-9-2LE configuration, an SV message from the MSVCB01 contains only one signal sample (no oversampling), and an SV message from the MSVCB02 contains eight signal samples (oversampling factor: 8). To generate 8 samples per PLC cycle, configure the oversampling factor to 8 for the MSVCB08 signal generator, for example, and to 1 for the MSVCB01 signal generator. The call method of MSVCB02 is called eight times per PLC cycle, and the call method of MSVCB01 is called once per PLC cycle. The cycle time of the PLC task then determines the effective sampling rate.
nSpecifiedSmpRate	UINT	Set	The sampling rate specified for the sampling mode: 'AtSmpRatePerPeriod' or 'AtSmpRatePerSecond'. Default value: 80. A value of 0 is invalid. In sampling mode: 'AtPlcTaskCycleTime', this parameter value is ignored and not used.

Example:

See here: SV Publisher/Subscriber according to UCA 61850-9-2 Light Edition [▶ 760].

6.3.3.6.8.1 Call

The method generates a new signal sample. In signal sampling mode: 'SecondsPerSample', this method is called nOversampleFactor times in each PLC task cycle.

```
METHOD FINAL Call
```

6.3.3.6.8.2 Config

This method configures the function generator based on the selected signal sampling mode (eMode), the PLC cycle time, the PLC cycle time correction factor (fCycleTimeCorrectionFactor), the configured oversampling factor (nOversampleFactor), the configured sampling rate (nSpecifiedSmpRate), and the signal frequency (fFrequency). The Config method is automatically called internally whenever these configuration parameters are changed.

```
METHOD FINAL Config: BOOL
```

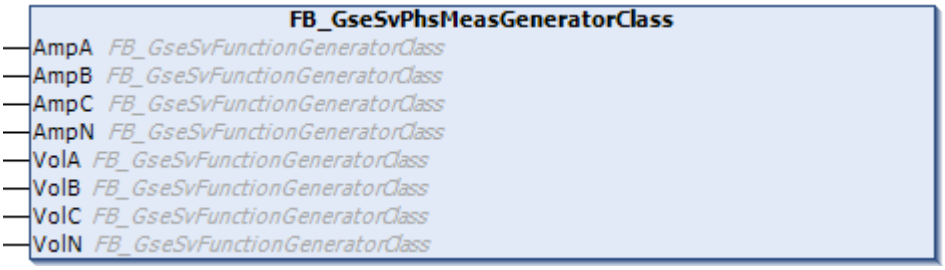
Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error. Invalid configuration parameter values, such as nOversamplingFactor = 0, fFrequency <= 0.0, nSpecifiedSmpRate = 0, or fCycleTimeCorrectionFactor <= 0.0, result in a negative acknowledgment.

6.3.3.6.8.3 Reset

The method resets the initial value, fSignal, to 0.0 and the sample number, nSample, to 0. After this method call, the next call to the method generates the first sample value with the number 0 again. If the configuration parameters are changed, the reset method is automatically called internally by the config method.

```
METHOD FINAL Reset
```

6.3.3.6.9 FB_GseSvPhsMeasGeneratorClass



This function block implements 8 test signal function generators according to the “PhsMeas1” data set of the UCA 61850-9-2LE (Light Edition). This function block can be used to generate simulation values for phases A, B, C, and N (up to 4 currents and 4 voltage values) for an SV communication test.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
FUNCTION_BLOCK FB_GseSvPhsMeasGeneratorClass
VAR_INPUT
    AmpA: FB_GseSvFunctionGeneratorClass; (* "InnATCTR1.AmpSv.instMag.i" signal function generator *
)
    AmpB: FB_GseSvFunctionGeneratorClass; (* "InnBTCTR2.AmpSv.instMag.i" signal function generator *
)
    AmpC: FB_GseSvFunctionGeneratorClass; (* "InnCTCTR3.AmpSv.instMag.i" signal function generator *
)
    AmpN: FB_GseSvFunctionGeneratorClass; (* "InnNTCTR4.AmpSv.instMag.i" signal function generator *
)
    VolA: FB_GseSvFunctionGeneratorClass; (* "UnnATVTR1.VolSv.instMag.i" signal function generator *
)
    VolB: FB_GseSvFunctionGeneratorClass; (* "UnnBTVTR2.VolSv.instMag.i" signal function generator *
)
    VolC: FB_GseSvFunctionGeneratorClass; (* "UnnCTVTR3.VolSv.instMag.i" signal function generator *
)
    VolN: FB_GseSvFunctionGeneratorClass; (* "UnnNTVTR4.VolSv.instMag.i" signal function generator *
)
END_VAR
```

Inputs

Name	Type	Description
AmpA	FB_GseSvFunctionGeneratorClass [▶ 293]	Test signal function generator for phase A current samples.
AmpB	FB_GseSvFunctionGeneratorClass [▶ 293]	Test signal function generator for phase B current samples.
AmpC	FB_GseSvFunctionGeneratorClass [▶ 293]	Test signal function generator for phase C current samples.
AmpN	FB_GseSvFunctionGeneratorClass [▶ 293]	Test signal function generator for current samples from phase N (neutral conductor).
VolA	FB_GseSvFunctionGeneratorClass [▶ 293]	Test signal function generator for the voltage samples of Phase A.
VolB	FB_GseSvFunctionGeneratorClass [▶ 293]	Test signal function generator for sampling the voltage of Phase B.
VolC	FB_GseSvFunctionGeneratorClass [▶ 293]	Test signal function generator for the voltage samples of Phase C.
VolN	FB_GseSvFunctionGeneratorClass [▶ 293]	Test signal function generator for sampling the voltage of phase N (neutral conductor).

Methods

Name	Description
Call [▶ 298]	The method generates a new sample value for 4 currents and 4 voltages of the phases: A, B, C, and N. In the signal sampling mode 'AtPlcTaskCycleTime' (standard), this method is called nOversampleFactor times in each PLC task cycle.
Config [▶ 298]	The method configures all 8 test signal function generators based on the selected signal sampling mode (eMode), the PLC cycle time, the PLC cycle time correction factor (fCycleTimeCorrectionFactor), the configured oversampling factor (nOversampleFactor), the configured sampling rate (nSpecifiedSmpRate), and the signal frequency (fFrequency). The Config method is automatically called internally whenever these configuration parameters are changed.
Reset [▶ 298]	For all 8 test signal function generators, this method resets the output values fSignal to 0 and the sample number nSample to 0. After the reset method is called, the next call method generates the first sample value with the number 0 again. If the configuration parameters are changed, the reset method is automatically called internally by the config method.

Properties

Name	Type	Access	Description
bValid	BOOL	Get	Provides a cumulative configuration status for all 8 test signal function generators. TRUE => Configuration is correct; FALSE => Configuration is incorrect. An incorrect configuration is caused by invalid parameter values: fCycleTimeCorrectionFactor <= 0, fFrequency <= 0, nOversampleFactor = 0, or nSpecifiedSmpRate = 0.
eMode	E_GseSvGeneratorMode [▶ 523]	Set	Signal sampling mode. Default value: 'AtPlcTaskCycleTime' => Sampling within the PLC task cycle. The sampling rate is determined by the cycle time of the PLC task and the frequency. In sampling mode: "AtSmpRatePerPeriod" or "AtSmpRatePerSecond," the parameter value "nSpecifiedSmpRate" determines the sampling rate.
fCycleTimeCorrectionFactor	LREAL	Set	Sets the correction factor for the PLC task cycle time. Default value: 1.0 (no correction). Values <= 0 are invalid.
fFrequency	LREAL	Set	Sets the signal frequency in Hz. Default value: 50 Hz. Values <= 0 are invalid.
nOversampleFactor	UINT(1..128)	Set	Sets the oversampling factor of the signal generator. The value 0 is invalid. Default value: 1 (no oversampling). In the IEC 61850-9-2LE configuration, for example, the oversampling factor is set to 1 for the MSVCB01 signal generator and to 8 for the MSVCB02 signal generator.
nSpecifiedSmpRate	UINT	Set	The sampling rate specified in sampling mode: 'AtSmpRatePerPeriod' or 'AtSmpRatePerSecond'. The value 0 is invalid. Default value: 80. In sampling mode: 'AtPlcTaskCycleTime', this parameter value is ignored and not used.

Example:

See here: SV Publisher/Subscriber according to UCA 61850-9-2 Light Edition [▶ 760].

6.3.3.6.9.1 Call

The method generates 4 new current samples (phases A, B, C, and N) and 4 new voltage samples (phases A, B, C, and N). In signal sampling mode: 'SecondsPerSample', this method is called nOversampleFactor times in each PLC task cycle.

```
METHOD FINAL Call
```

6.3.3.6.9.2 Config

The method configures all 8 test signal function generators based on the selected signal sampling mode (eMode), the PLC task cycle time, the PLC cycle time correction factor (fCycleTimeCorrectionFactor), the configured oversampling factor (nOversampleFactor), the configured sampling rate (nSpecifiedSmpRate), and the signal frequency (fFrequency).

```
METHOD FINAL Config: BOOL
```

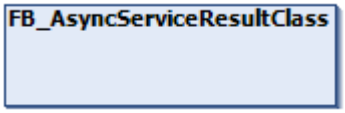
Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error. Invalid configuration parameter values, such as nOversamplingFactor = 0, fFrequency <= 0.0, nSpecifiedSmpRate = 0, or fCycleTimeCorrectionFactor <= 0.0, result in a negative acknowledgment.

6.3.3.6.9.3 Reset

For all test signal function generators, this method resets the output value fSignal to 0.0 and the sample number nSample to 0. After this method call, the next call to the method generates the first sample value with the number 0 again. If the configuration parameters are changed, the reset method is automatically called internally by the config method.

```
METHOD FINAL Reset
```

6.3.4 FB_AsyncServiceResultClass



The function block FB_AsyncServiceResultClass implements the I_AsyncServiceResultClass [▶ 463] interface for status query and tracking of asynchronous processes. The methods of the communication function block FB_iec61850CommonClass [▶ 247] for establishing/disconnecting the connection and activation of the IEC 61850 services return the interface pointer of type I_AsyncServiceResultClass [▶ 463] as parameter. The application can use this pointer to monitor and query the status/progress of service execution.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

Inheritance hierarchy

FB_AsyncServiceResultClass

```
FUNCTION_BLOCK FB_AsyncServiceResultClass IMPLEMENTS I_AsyncServiceResultClass
```

Interfaces

Type	Description
I_AsyncServiceResultClass [▶ 463]	Interface for status query and tracking of asynchronous processes.

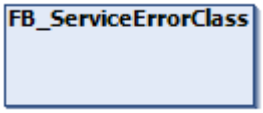
Methods

Name	Description
CopyFrom [▶ 464]	Copies object data from another object instance.
Dump [▶ 464]	Converts own object data into a formatted string.
Equal [▶ 464]	Value comparison of own object data with external data values.
EqualEx [▶ 465]	Value comparison of own object data with external data values.
EqualTo [▶ 465]	Value comparison of own object data with the data of another object instance.
IsAborted [▶ 465]	Checks whether the service was aborted.
IsBusy [▶ 465]	Checks whether the service is still running.
IsCompleted [▶ 466]	Checks whether the service execution was completed successfully.
IsFailed [▶ 466]	Checks whether the service execution was completed with an error.
SetAborted [▶ 466]	Sets the abort status.
SetAbortedEx [▶ 466]	Sets the abort status.
SetBusy [▶ 467]	Sets the status to busy.
SetCompleted [▶ 467]	Sets the status to completed (without error).
SetFailed [▶ 467]	Sets the error state.
SetFailedEx [▶ 467]	Sets the error state.

Properties

Name	Type	Access	Description
eState [▶ 468]	E_AsyncServiceState [▶ 519]	Get	Status information

6.3.5 FB_ServiceErrorClass



The function block FB_ServiceErrorClass implements the I_ServiceErrorClass [▶ 468] interface for error query and storage. The methods of the object instances of the TwinCAT IEC 61850 data model and some services for establishing/disconnecting the connection return the interface pointer of type I_ServiceErrorClass [▶ 468] in the event of an error. This pointer can be used to perform a more detailed error analysis.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

Inheritance hierarchy

FB_ServiceErrorClass

```
FUNCTION_BLOCK FB_ServiceErrorClass IMPLEMENTS I_ServiceErrorClass
```

Interfaces

Type	Description
I_ServiceErrorClass [▶ 468]	Interface for error query and storage.

Methods

Name	Description
CopyFrom [▶ 469]	Copies object data from another object instance.
Dump [▶ 469]	Converts own object data to a formatted string (e.g. for log outputs).
Equal [▶ 469]	Value comparison of own object data (error information) with external data values.
EqualEx [▶ 469]	Value comparison of own object data (error information) with external data values.
EqualTo [▶ 469]	Value comparison of own object data (error information) with the data of another object instance.
IsFailed [▶ 470]	Checks own object data (error information) for failure.
IsSucceeded [▶ 470]	Checks own object data (error information) for success.
SetFailedEx [▶ 470]	Stores error information in its own object instance.
SetSucceeded [▶ 470]	Stores success information in its own object instance.

Properties

Name	Type	Access	Description
stError [▶ 471]	ST_ServiceErrorClass [▶ 555]	Get	Object data as a structured type.

6.4 Functions

6.4.1 Accuracy_To_UtcTimeQualityAccuracy

Converts the value of the enumeration type E_UtcTimeAccuracy [▶ 527] (accuracy of the time information) into five Boolean values. These five output values correspond to the bit components: Accuracy0..Accuracy4 of the structured type T_UtcTimeQuality [▶ 560].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION Accuracy_To_UtcTimeQualityAccuracy
VAR_INPUT
    in         : E_UtcTimeAccuracy;
END_VAR
VAR_OUTPUT
    bAccuracy0 : BOOL;
    bAccuracy1 : BOOL;
    bAccuracy2 : BOOL;
    bAccuracy3 : BOOL;
    bAccuracy4 : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	E_UtcTimeAccuracy [▶ 527]	Accuracy of the seconds fraction. This value determines the max. number of significant bits of the T_UtcTime [▶ 560].fractionOfSecond component.

Outputs

Converted information as T_UtcTimeQuality [▶ 560].Accuracy0..4 bit components. See: Coding of the UTC-Time.Quality.Accuracy0..4 bits [▶ 774].

Name	Type	Description
bAccuracy0	BOOL	Accuracy MSB bit 0.
bAccuracy1	BOOL	Accuracy bit 1.
bAccuracy2	BOOL	Accuracy bit 2.
bAccuracy3	BOOL	Accuracy bit 3.
bAccuracy4	BOOL	Accuracy MSB bit 4.

Return value

None.

Sample

```
METHOD UtcTimeQualityAccuracy_ToFrom_Accuracy : BOOL
VAR_INPUT
    fbIED    : REFERENCE TO FB_IED;
END_VAR
VAR
    Quality  : T_UtcTimeQuality;
    sQuality : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

Quality.LeapSecondsKnown := FALSE;
Quality.ClockFailure := FALSE;
Quality.ClockNotSynchronized := FALSE;
Accuracy_To_UtcTimeQualityAccuracy( E_UtcTimeAccuracy.Null,(* Null bits accuracy *)
                                    bAccuracy0=>Quality.Accuracy0,
                                    bAccuracy1=>Quality.Accuracy1,
                                    bAccuracy2=>Quality.Accuracy2,
                                    bAccuracy3=>Quality.Accuracy3,
                                    bAccuracy4=>Quality.Accuracy4 );
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|0' *)
IF fbIED.Relay.LLN0.Beh.t.eAccuracy <> UtcTimeQualityAccuracy_To_Accuracy(Quality) THEN
    Quality := fbIED.Relay.LLN0.Beh.t.Quality;
END_IF
```

```
Quality.LeapSecondsKnown := FALSE;
Quality.ClockFailure := FALSE;
Quality.ClockNotSynchronized := FALSE;
Accuracy_To_UtcTimeQualityAccuracy( E_UtcTimeAccuracy._03,(* 3 bits accuracy *)
                                    bAccuracy0=>Quality.Accuracy0,
                                    bAccuracy1=>Quality.Accuracy1,
                                    bAccuracy2=>Quality.Accuracy2,
                                    bAccuracy3=>Quality.Accuracy3,
                                    bAccuracy4=>Quality.Accuracy4 );
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|3' *)
IF fbIED.Relay.LLN0.Health.t.eAccuracy <> UtcTimeQualityAccuracy_To_Accuracy(Quality) THEN
    Quality := fbIED.Relay.LLN0.Health.t.Quality;
END_IF

Quality.LeapSecondsKnown := FALSE;
Quality.ClockFailure := FALSE;
Quality.ClockNotSynchronized := FALSE;
Accuracy_To_UtcTimeQualityAccuracy( E_UtcTimeAccuracy._05,(* 5 bits accuracy *)
                                    bAccuracy0=>Quality.Accuracy0,
                                    bAccuracy1=>Quality.Accuracy1,
                                    bAccuracy2=>Quality.Accuracy2,
                                    bAccuracy3=>Quality.Accuracy3,
                                    bAccuracy4=>Quality.Accuracy4 );
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|5' *)
IF fbIED.Relay.LLN0.Mod_.t.eAccuracy <> UtcTimeQualityAccuracy_To_Accuracy(Quality) THEN
    Quality := fbIED.Relay.LLN0.Mod_.t.Quality;
END_IF

Quality.LeapSecondsKnown := FALSE;
Quality.ClockFailure := FALSE;
Quality.ClockNotSynchronized := FALSE;
Accuracy_To_UtcTimeQualityAccuracy( E_UtcTimeAccuracy.Unspecified,(* Unspecified accuracy *)
                                    bAccuracy0=>Quality.Accuracy0,
                                    bAccuracy1=>Quality.Accuracy1,
                                    bAccuracy2=>Quality.Accuracy2,
                                    bAccuracy3=>Quality.Accuracy3,
                                    bAccuracy4=>Quality.Accuracy4 );
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|31' *)
IF fbIED.Relay.MMXU1.phV.phsA.t.eAccuracy <> UtcTimeQualityAccuracy_To_Accuracy(Quality) THEN
    Quality := fbIED.Relay.MMXU1.phV.phsA.t.Quality;
END_IF

UtcTimeQualityAccuracy_ToFrom_Accuracy := TRUE;
```

6.4.2 AcsiAnalogueValue

Initializes the data elements of the structure: ST_AcsiAnalogueValue.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiAnalogueValue : ST_AcsiAnalogueValue
VAR_INPUT
    i : DINT;
    f : REAL;
END_VAR
```

Inputs

Name	Type	Description
i	DINT	Data element "i" to be set.
f	REAL	Data element "f" to be set.

Return value

Name	Type	Description
AcsiAnalogueValue	ST_AcsiAnalogueValue [▶ 531]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiAnalogueValue : BOOL
VAR
    mag: ST_AcsiAnalogueValue;
END_VAR
```

```
mag:=AcsiAnalogueValue(i:=0, f:=12.45);
Sample_AcsiAnalogueValue:=TRUE;
```

6.4.3 AcsiCalendarTime

Initializes the data elements of the structure: ST_AcsiCalendarTime.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiCalendarTime : ST_AcsiCalendarTime
VAR_INPUT
    occ     : WORD;
    occType : E_AcsiOccType;
    occPer  : E_AcsiOccPer;
    weekDay : E_AcsiWeekDay;
    month   : E_AcsiMonth;
    day     : BYTE;
    hr      : BYTE;
    mn      : BYTE;
END_VAR
```

Inputs

Name	Type	Description
occ	WORD	Data element "occ" to be set.
occType	E_AcsiOccType [▶ 495]	Data element "occType" to be set.
occPer	E_AcsiOccPer [▶ 494]	Data element "occPer" to be set.
weekDay	E_AcsiWeekDay [▶ 518]	Data element "weekDay" to be set.
month	E_AcsiMonth [▶ 493]	Data element "month" to be set.
day	BYTE	Data element "day" to be set.
hr	BYTE	Data element "hr" to be set.
mn	BYTE	Data element "mn" to be set.

Return value

Name	Type	Description
AcsiCalendarTime	ST_AcsiCalendarTime [▶ 532]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiCalendarTime : BOOL
VAR
    calTm: ST_AcsiCalendarTime;
END_VAR
```

```
calTm:=AcsiCalendarTime(occ:=0, occType:=E_AcsiOccType.Time_,
    occPer:=E_AcsiOccPer.Hour, weekDay:=E_AcsiWeekDay.Friday,
    month:=E_AcsiMonth.April, day:=1, hr:=0, mn:=0);
Sample_AcsiCalendarTime:=TRUE;
```

6.4.4 AcsiCell

Initializes the data elements of the structure: ST_AcsiCell.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiCell : ST_AcsiCell
VAR_INPUT
    xStart : REAL;
    xEnd   : REAL;
    yStart : REAL;
    yEnd   : REAL;
END_VAR
```

Inputs

Name	Type	Description
xStart	REAL	Data element "xStart" to be set.
xEnd	REAL	Data element "xEnd" to be set.
yStart	REAL	Data element "yStart" to be set.
yEnd	REAL	Data element "yEnd" to be set.

Return value

Name	Type	Description
AcsiCell	ST_AcsiCell [▶ 532]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiCell : BOOL
VAR
    cell: ST_AcsiCell;
END_VAR
```

```
cell:=AcsiCell(xStart:=1.0, xEnd:=1.0, yStart:=1.0, yEnd:=1.0);
Sample_AcsiCell:=TRUE;
```

6.4.5 AcsiLogOptionalFields

Initializes the data elements of the structure: ST_AcsiLogOptionalFields.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiLogOptionalFields : ST_AcsiLogOptionalFields
VAR_INPUT
    ReasonForInclusion : BOOL;
END_VAR
```

Inputs

Name	Type	Description
ReasonForInclusion	BOOL	Data element "ReasonForInclusion" to be set.

Return value

Name	Type	Description
AcsiLogOptionalFields	ST_AcsiLogOptionalFields [▶ 536]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiLogOptionalFields : BOOL
VAR
    fields : ST_AcsiLogOptionalFields;
END_VAR
```

```
fields:=AcsiLogOptionalFields(ReasonForInclusion:=FALSE);
Sample_AcsiLogOptionalFields:=TRUE;
```

6.4.6 AcsiOptionalFields

Initializes the data elements of the structure: ST_AcsiOptionalFields.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiOptionalFields : ST_AcsiOptionalFields
VAR_INPUT
    SequenceNumber     : BOOL;
    ReportTimeStamp    : BOOL;
    ReasonForInclusion : BOOL;
    DataSetName        : BOOL;
    DataReference      : BOOL;
    BufferOverflow     : BOOL;
    EntryID            : BOOL;
    ConfRevision       : BOOL;
    Segmentation       : BOOL;
END_VAR
```

Inputs

Name	Type	Description
SequenceNumber	BOOL	Data element "SequenceNumber" to be set.
ReportTimeStamp	BOOL	Data element "ReportTimeStamp" to be set.
ReasonForInclusion	BOOL	Data element "ReasonForInclusion" to be set.
DataSetName	BOOL	Data element "DataSetName" that is to be set.
DataReference	BOOL	Data element "DataReference" to be set.
BufferOverflow	BOOL	Data element "BufferOverflow" to be set.
EntryID	BOOL	Data element "EntryID" to be set.
ConfRevision	BOOL	Data element "ConfRevision" to be set.
Segmentation	BOOL	Data element "Segmentation" to be set.

Return value

Name	Type	Description
AcsiOptionalFields	ST_AcsiOptionalFields [▶ 536]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiOptionalFields : BOOL
VAR
    fields : ST_AcsiOptionalFields;
END_VAR
```

```
fields:=AcsiOptionalFields(SequenceNumber:=FALSE, ReportTimeStamp:=FALSE,
    ReasonForInclusion:=FALSE, DataSetName:=FALSE, DataReference:=FALSE,
    BufferOverflow:=FALSE, EntryID:=FALSE, ConfRevision:=0, Segmentation:=FALSE);
Sample_AcsiOptionalFields:=TRUE;
```

6.4.7 AcsiOriginator

Initializes the data elements of the structure: ST_AcsiOriginator.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiOriginator : ST_AcsiOriginator
VAR_INPUT
    orCat   : E_AcsiOrCategory;
    orIdent : T_OCTET64;
END_VAR
```

Inputs

Name	Type	Description
orCat	E_AcsiOrCategory [▶ 496]	Data element "orCat" to be set.
orIdent	T_OCTET64 [▶ 559]	Data element "orIdent" to be set.

Return value

Name	Type	Description
AcsiOriginator	ST_AcsiOriginator [▶ 538]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiOriginator : BOOL
VAR
    org: ST_AcsiOriginator;
    orIdent : T_OCTET64:=[16#36,16#31, 16#38, 16#35, 16#30];
END_VAR
```

```
org:=AcsiOriginator(orCat:=E_AcsiOrCategory.StationControl, orIdent:=orIdent);
Sample_AcsiOriginator:=TRUE;
```

6.4.8 AcsiPhyComAddr

Initializes the data elements of the structure: ST_AcsiPhyComAddr.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiPhyComAddr : ST_AcsiPhyComAddr
VAR_INPUT
    Addr     : T_OCTET6;
    PRIORITY : BYTE;
    VID      : WORD;
    APPID    : WORD;
END_VAR
```

Inputs

Name	Type	Description
Addr	T_OCTET6 [▶ 558]	Data element "Addr" to be set.
PRIORITY	BYTE	Data element "PRIORITY" to be set.
VID	WORD	Data element "VID" to be set.
APPID	WORD	Data element "APPID" to be set.

Return value

Name	Type	Description
AcsiPhyComAddr	ST_AcsiPhyComAddr [▶ 538]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiPhyComAddr : BOOL
VAR
    addr: ST_AcsiPhyComAddr;
END_VAR
```

```
addr:=AcsiPhyComAddr(Addr:=LWORD_TO_OCTET6(16#AC3040506001), PRIORITY:=4, VID:=16#0001, APPID:=16#00
01);
Sample_AcsiPhyComAddr:=TRUE;
```

6.4.9 AcsiPoint

Initializes the data elements of the structure: ST_AcsiPoint.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiPoint : ST_AcsiPoint
VAR_INPUT
    xVal : REAL;
    yVal : REAL;
    zVal : REAL;
END_VAR
```

Inputs

Name	Type	Description
xVal	REAL	Data element "xVal" to be set.
yVal	REAL	Data element "yVal" to be set.
zVal	REAL	Data element "zVal" to be set.

Return value

Name	Type	Description
AcsiPoint	ST_AcsiPoint [▶ 538]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiPoint : BOOL
VAR
    point: ST_AcsiPoint;
END_VAR
```

```
point:=AcsiPoint(xVal:=1.0, yVal:=1.0, zVal:=1.0);
Sample_AcsiPoint:=TRUE;
```

6.4.10 AcsiPulseConfig

Initializes the data elements of the structure: ST_AcsiPulseConfig.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiPulseConfig : ST_AcsiPulseConfig
VAR_INPUT
    cmdQual : E_AcsiCmdQual;
    onDur   : DWORD;
    offDur  : DWORD;
    numPls  : DWORD;
END_VAR
```

Inputs

Name	Type	Description
cmdQual	E_AcsiCmdQual [▶ 482]	Data element "cmdQual" to be set.
onDur	DWORD	Data element "onDur" to be set.
offDur	DWORD	Data element "offDur" to be set.
numPls	DWORD	Data element "numPls" to be set.

Return value

Name	Type	Description
AcsiPulseConfig	ST_AcsiPulseConfig [▶ 540]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiPulseConfig : BOOL
VAR
    config: ST_AcsiPulseConfig;
END_VAR
```

```
config:=AcsiPulseConfig(cmdQual:=E_AcsiCmdQual.Pulse,
        onDur:=10, offDur:=10, numPls:=1);
Sample_AcsiPulseConfig:=TRUE;
```

6.4.11 AcsiQuality

Initializes the data elements of the structure: ST_AcsiQuality.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiQuality : ST_AcsiQuality
VAR_INPUT
    Validity0       : BOOL;
    Validity1       : BOOL;
    Overflow        : BOOL;
    OutOfRange      : BOOL;
    BadReference    : BOOL;
    Oscillatory     : BOOL;
    Failure         : BOOL;
    OldData         : BOOL;
    Inconsistent    : BOOL;
    Inaccurate      : BOOL;
    Source          : BOOL;
    Test            : BOOL;
    OperatorBlocked : BOOL;
END_VAR
```

Inputs

Name	Type	Description
Validity0	BOOL	Data element "Validity0" to be set.
Validity1	BOOL	Data element "Validity1" to be set.
Overflow	BOOL	Data element "Overflow" to be set.
OutOfRange	BOOL	Data element "OutOfRange" to be set.
BadReference	BOOL	Data element "BadReference" to be set.
Oscillatory	BOOL	Data element "Oscillatory" to be set.
Failure	BOOL	Data element "Failure" to be set.
OldData	BOOL	Data element "OldData" to be set.
Inconsistent	BOOL	Data element "Inconsistent" to be set.
Inaccurate	BOOL	Data element "Inaccurate" to be set.
Source	BOOL	Data element "Source" to be set.
Test	BOOL	Data element "Test" to be set.
OperatorBlocked	BOOL	Data element "OperatorBlocked" to be set.

Return value

Name	Type	Description
AcsiQuality	ST_AcsiQuality [▶ 541]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiQuality : BOOL
VAR
    q: ST_AcsiQuality;
END_VAR
```

```
q:=AcsiQuality(Validity0:=0, Validity1:=0, Overflow:=0, OutOfRange:=0,
               BadReference:=0, Oscillatory:=0, Failure:=0, OldData:=0,
               Inconsistent:=0, Inaccurate:=0, Source:=0, Test:=0,
               OperatorBlocked:=0);
Sample_AcsiQuality:=TRUE;
```

6.4.12 AcsiQuality_To_String

Converts the components of the structured type ST_AcsiQuality [▶ 541] into a formatted Quality string. The resulting string has the following structure: 'Q#vv|ORBSFDCA|s|TB'. It corresponds to the TwinCAT Quality-String format specification [▶ 778].

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiQuality_To_String : STRING(18)
VAR_INPUT
    in : ST_AcsiQuality;
END_VAR
```

Inputs

Name	Type	Description
in	ST_AcsiQuality [▶ 541]	The information to be converted as a Quality structure.

Return value

Name	Type	Description
AcsiQuality_To_String	STRING(18)	Converted information as a Quality string.

Sample

```
METHOD AcsiQuality_ToFrom_String : BOOL
VAR_INPUT
    fbIED  : REFERENCE TO FB_IED;
END_VAR
VAR
    qValue : ST_AcsiQuality;
    sValue : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

qValue := String_To_AcsiQuality('Q#00|00000000|0|00');
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#00|00000000|0|00' *)
IF fbIED.Relay.LLN0.Beh.q.sValue <> AcsiQuality_To_String(qValue) THEN
    qValue := fbIED.Relay.LLN0.Beh.q.qValue;
END_IF

qValue := String_To_AcsiQuality('Q#01|00110011|0|01');
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#01|00110011|0|01' *)
IF fbIED.Relay.LLN0.Health.q.sValue <> AcsiQuality_To_String(qValue) THEN
    qValue := fbIED.Relay.LLN0.Health.q.qValue;
END_IF

qValue := String_To_AcsiQuality('Q#10|11001100|1|10');
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#10|11001100|1|10' *)
IF fbIED.Relay.LLN0.Mod_.q.sValue <> AcsiQuality_To_String(qValue) THEN
    qValue := fbIED.Relay.LLN0.Mod_.q.qValue;
```

```
END_IF

AcsiQuality_ToFrom_String := TRUE;
```

6.4.13 AcsiQuality_To_WORD

Converts the components of the structured type: ST_AcsiQuality [▶ 541] into binary data of the type WORD. The coding of the components in the WORD corresponds to the TwinCAT Quality-WORD format specification [▶ 779].

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiQuality_To_Word : WORD(0..8191)
VAR_INPUT
    in : ST_AcsiQuality;
END_VAR
```

Inputs

Name	Type	Description
in	ST_AcsiQuality [▶ 541]	The information to be converted as a Quality structure.

Return value

Name	Type	Description
AcsiQuality_To_WORD	WORD(0..8191)	Converted information as Quality binary data.

Sample

```
METHOD AcsiQuality_ToFrom_WORD : BOOL
VAR_INPUT
    fbIED : REFERENCE TO FB_IED;
END_VAR
VAR
    qValue : ST_AcsiQuality;
    sValue : STRING;
    nValue : WORD;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

qValue := WORD_To_AcsiQuality(0);
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#00|00000000|0|00' *)
nValue := AcsiQuality_To_WORD(qValue);(* nValue := 2#00000000_00000000 *)
IF fbIED.Relay.LLN0.Beh.q.nValue <> nValue THEN
    qValue := fbIED.Relay.LLN0.Beh.q.qValue;
END_IF

qValue := String_To_AcsiQuality('Q#01|00110011|0|01');
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#01|00110011|0|01' *)
nValue := AcsiQuality_To_Word(qValue);(* nValue := 2#00010011_00110010 *)
IF fbIED.Relay.LLN0.Health.q.nValue <> nValue THEN
    qValue := fbIED.Relay.LLN0.Health.q.qValue;
END_IF

qValue := String_To_AcsiQuality('Q#10|11001100|1|10');
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#10|11001100|1|10' *)
nValue := AcsiQuality_To_Word(qValue);(* nValue := 2#00001100_11001101 *)
IF fbIED.Relay.LLN0.Mod_.q.nValue <> nValue THEN
    qValue := fbIED.Relay.LLN0.Mod_.q.qValue;
END_IF

AcsiQuality_ToFrom_WORD := TRUE;
```

6.4.14 AcsiQualitySource_To_Source

Converts the Source-Bit component of the structured type: ST_AcsiQuality [▶ 541] into the enumeration type: E_AcsiQualitySource [▶ 501]. Only the Source-Bit component is converted.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiQualitySource_To_Source : E_AcsiQualitySource
VAR_INPUT
    in : ST_AcsiQuality;
END_VAR
```

Inputs

Name	Type	Description
in	ST_AcsiQuality [▶ 541]	The information to be converted as a Quality-Source-Bit component.

Return value

Name	Type	Description
AcsiQualitySource_To_Source	E_AcsiQualitySource [▶ 501]	Converted information as a Quality-Source enumeration type.

Sample

```
METHOD AcsiQualitySource_ToFrom_Source : BOOL
VAR_INPUT
    fbIED  : REFERENCE TO FB_IED;
END_VAR
VAR
    qValue : ST_AcsiQuality;
    sValue : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

Validity_To_AcsiQualityValidity(E_AcsiQualityValidity.Good, bValidity0=>qValue.Validity0, bValidity1
=> qValue.Validity1 );
qValue.Overflow := FALSE;
qValue.OutOfRange := FALSE;
qValue.BadReference := FALSE;
qValue.Oscillatory := FALSE;
qValue.Failure := FALSE;
qValue.OldData := FALSE;
qValue.Inconsistent := FALSE;
qValue.Inaccurate := FALSE;
qValue.Source := Source_To_AcsiQualitySource(E_AcsiQualitySource.Substituted);
qValue.Test := TRUE;
qValue.OperatorBlocked := FALSE;
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#00|00000000|1|10' *)

IF fbIED.Relay.LLN0.Beh.q.eSource <> AcsiQualitySource_To_Source(qValue) THEN
     qValue := fbIED.Relay.LLN0.Beh.q.qValue;
END_IF

AcsiQualitySource_ToFrom_Source := TRUE;
```

6.4.15 AcsiQualityValidity_To_Validity

Converts the Validity0 and Validity1 bit components of the structured type: ST_AcsiQuality [▶ 541] into the enumeration type: E_AcsiQualityValidity [▶ 501]. Only the Validity0 and Validity1 bit components are converted. See also: Coding of the Quality-Validity0..1-bits [▶ 778].

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiQualityValidity_To_Validity : E_AcsiQualityValidity
VAR_INPUT
    in : ST_AcsiQuality;
END_VAR
```

Inputs

Name	Type	Description
in	ST_AcsiQuality [▶ 541]	The information to be converted as Quality-Validity bit components.

Return value

Name	Type	Description
AcsiQualityValidity_To_Validity	E_AcsiQualityValidity [▶ 501]	Converted information as Quality-Validity enumeration type.

Sample

```
METHOD AcsiQualityValidity_ToFrom_Validity : BOOL
VAR_INPUT
    fbIED  : REFERENCE TO FB_IED;
END_VAR
VAR
    qValue : ST_AcsiQuality;
    sValue : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF
Validity_To_AcsiQualityValidity(E_AcsiQualityValidity.Good, bValidity0=>qValue.Validity0, bValidity1
=>qValue.Validity1);
qValue.Overflow := FALSE;
qValue.OutOfRange := FALSE;
qValue.BadReference := FALSE;
qValue.Oscillatory := FALSE;
qValue.Failure := FALSE;
qValue.OldData := FALSE;
qValue.Inconsistent := FALSE;
qValue.Inaccurate := FALSE;
qValue.Source := Source_To_AcsiQualitySource(E_AcsiQualitySource.Process);
qValue.Test := FALSE;
qValue.OperatorBlocked := FALSE;
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#00|00000000|0|00' *)

IF fbIED.Relay.LLN0.Beh.q.eValidity <> AcsiQualityValidity_To_Validity(qValue) THEN
    qValue := fbIED.Relay.LLN0.Beh.q.qValue;
END_IF

AcsiQualityValidity_ToFrom_Validity := TRUE;
```

6.4.16 AcsiRangeConfig

Initializes the data elements of the structure: ST_AcsiRangeConfig.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiRangeConfig : ST_AcsiRangeConfig
VAR_INPUT
    hhLim : ST_AcsiAnalogueValue;
    hLim  : ST_AcsiAnalogueValue;
    lLim  : ST_AcsiAnalogueValue;
    llLim : ST_AcsiAnalogueValue;
    min_  : ST_AcsiAnalogueValue;
```

```
    max_   : ST_AcsiAnalogueValue;
    limDb  : DWORD;
END_VAR
```

Inputs

Name	Type	Description
hhLim	ST_AcsiAnalogueValue [▶ 531]	Data element "hhLim" to be set.
hLim	ST_AcsiAnalogueValue [▶ 531]	Data element "hLim" to be set.
lLim	ST_AcsiAnalogueValue [▶ 531]	Data element "lLim" to be set.
llLim	ST_AcsiAnalogueValue [▶ 531]	Data element "llLim" to be set.
min_	ST_AcsiAnalogueValue [▶ 531]	Data element "min_" to be set.
max_	ST_AcsiAnalogueValue [▶ 531]	Data element "max_" to be set.
limDb	DWORD	Data element "limDb" to be set.

Return value

Name	Type	Description
AcsiRangeConfig	ST_AcsiRangeConfig [▶ 542]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiRangeConfig : BOOL
VAR
    range: ST_AcsiRangeConfig;
END_VAR
```

```
range:=AcsiRangeConfig(hhLim:=AcsiAnalogueValue(0,0), hLim:=AcsiAnalogueValue(0,0), lLim:=AcsiAnalog
ueValue(0,0), llLim:=AcsiAnalogueValue(0,0), min_:=AcsiAnalogueValue(0,0), max_:=AcsiAnalogueValue(0
,0), limDb:=0);
Sample_AcsiRangeConfig:=TRUE;
```

6.4.17 AcsiReasonCode

Initializes the data elements of the structure: ST_AcsiReasonCode.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiReasonCode : ST_AcsiReasonCode
VAR_INPUT
    DataChange           : BOOL;
    QualityChange        : BOOL;
    DataUpdate           : BOOL;
    Integrity            : BOOL;
    GeneralInterrogation : BOOL;
    ApplicationTrigger   : BOOL;
END_VAR
```

Inputs

Name	Type	Description
DataChange	BOOL	Data element "DataChange" to be set.
QualityChange	BOOL	Data element "QualityChange" to be set.
DataUpdate	BOOL	Data element "DataUpdate" to be set.
Integrity	BOOL	Data element "Integrity" to be set.
GeneralInterrogation	BOOL	Data element "GeneralInterrogation" to be set.
ApplicationTrigger	BOOL	Data element "ApplicationTrigger" that is to be set.

Return value

Name	Type	Description
AcsiReasonCode	ST_AcsiReasonCode [▶ 542]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiReasonCode : BOOL
VAR
    reason : ST_AcsiReasonCode;
END_VAR
```

```
reason:=AcsiReasonCode(DataChange:=TRUE, QualityChange:=TRUE, DataUpdate:=TRUE,
    Integrity:=FALSE, GeneralInterrogation:=FALSE, ApplicationTrigger:=FALSE);
Sample_AcsiReasonCode:=TRUE;
```

6.4.18 AcsiScaledValueConfig

Initializes the data elements of the structure: ST_AcsiScaledValueConfig.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiScaledValueConfig : ST_AcsiScaledValueConfig
VAR_INPUT
    scaleFactor : REAL;
    offset      : REAL;
END_VAR
```

Inputs

Name	Type	Description
scaleFactor	REAL	Data element "scaleFactor" to be set.
offset	REAL	Data element "offset" to be set.

Return value

Name	Type	Description
AcsiScaledValueConfig	ST_AcsiScaledValueConfig [▶ 543]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiScaledValueConfig : BOOL
VAR
    config: ST_AcsiScaledValueConfig;
END_VAR
```

```
config:=AcsiScaledValueConfig(scaleFactor:=1.0, offset:=100);
Sample_AcsiScaledValueConfig:=TRUE;
```

6.4.19 AcsiSvOptionalFields

Initializes the data elements of the structure: ST_AcsiSvOptionalFields.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiSvOptionalFields : ST_AcsiSvOptionalFields
VAR_INPUT
    RefreshTime         : BOOL;
    SampleSynchronized  : BOOL;
    SampleRate          : BOOL;
    DataSetName         : BOOL;
    Security            : BOOL;
```

```
    SampleMode          : BOOL;
    SynchSourceIdentity : BOOL;
END_VAR
```

Inputs

Name	Type	Description
RefreshTime	BOOL	Data element "RefreshTime" to be set.
SampleSynchronized	BOOL	Data element "SampleSynchronized" to be set.
SampleRate	BOOL	Data element "SampleRate" to be set.
DataSetName	BOOL	Data element "DataSetName" that is to be set.
Security	BOOL	Data element "Security" to be set.
SampleMode	BOOL	Data element "SampleMode" to be set.
SynchSourceIdentity	BOOL	Data element "SynchSourceIdentity" to be set.

Return value

Name	Type	Description
AcsiSvOptionalFields	ST_AcsiSvOptionalFields [▶ 543]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiSvOptionalFields : BOOL
VAR
    fields : ST_AcsiSvOptionalFields;
END_VAR
```

```
fields:=AcsiSvOptionalFields(RefreshTime:=FALSE, SampleSynchronized:=FALSE,
        SampleRate:=FALSE, DataSetName:=TRUE, Security:=FALSE,
        SampleMode:=FALSE, SynchSourceIdentity:=FALSE);
Sample_AcsiSvOptionalFields:=TRUE;
```

6.4.20 AcsiTriggerConditions

Initializes the data elements of the structure: ST_AcsiTriggerConditions.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiTriggerConditions : ST_AcsiTriggerConditions
VAR_INPUT
    DataChange          : BOOL;
    QualityChange       : BOOL;
    DataUpdate          : BOOL;
    Integrity           : BOOL;
    GeneralInterrogation : BOOL;
END_VAR
```

Inputs

Name	Type	Description
DataChange	BOOL	Data element "DataChange" to be set.
QualityChange	BOOL	Data element "QualityChange" to be set.
DataUpdate	BOOL	Data element "DataUpdate" to be set.
Integrity	BOOL	Data element "Integrity" to be set.
GeneralInterrogation	BOOL	Data element "GeneralInterrogation" to be set.

Return value

Name	Type	Description
AcsiTriggerConditions	ST_AcsiTriggerConditions [▶ 544]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiTriggerConditions : BOOL
VAR
    trigger : ST_AcsiTriggerConditions;
END_VAR
```

```
trigger:=AcsiTriggerConditions(DataChange:=TRUE, QualityChange:=TRUE, DataUpdate:=TRUE, Integrity:=F
ALSE, GeneralInterrogation:=FALSE);
Sample_AcsiTriggerConditions:=TRUE;
```

6.4.21 AcsiUnit

Initializes the data elements of the structure: ST_AcsiUnit.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiUnit : ST_AcsiUnit
VAR_INPUT
    SIUnit     : E_AcsiSIUnit;
    multiplier : E_AcsiMultiplier;
END_VAR
```

Inputs

Name	Type	Description
SIUnit	E_AcsiSIUnit [▶ 508]	Data element "SIUnit" to be set.
multiplier	E_AcsiMultiplier [▶ 493]	Data element "multiplier" to be set.

Return value

Name	Type	Description
AcsiUnit	ST_AcsiUnit [▶ 544]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiUnit : BOOL
VAR
    unit: ST_AcsiUnit;
END_VAR
```

```
unit:=AcsiUnit(SIUnit:=E_AcsiSIUnit.Ampere, multiplier:=E_AcsiMultiplier.Milli);
Sample_AcsiUnit:=TRUE;
```

6.4.22 AcsiValWithTrans

Initializes the data elements of the structure: ST_AcsiValWithTrans.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiValWithTrans : ST_AcsiValWithTrans
VAR_INPUT
    posVal   : SINT;
    transInd : BOOL;
END_VAR
```

Inputs

Name	Type	Description
posVal	SINT	Data element "posVal" to be set.
transInd	BOOL	Data element "transInd" to be set.

Return value

Name	Type	Description
AcsiValWithTrans	ST_AcsiValWithTrans [▶ 545]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiValWithTrans : BOOL
VAR
    valT : ST_AcsiValWithTrans;
END_VAR
```

```
valT:=AcsiValWithTrans(posVal:=1, transInd:=FALSE);
Sample_AcsiValWithTrans:=TRUE;
```

6.4.23 AcsiVector

Initializes the data elements of the structure: ST_AcsiVector.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION AcsiVector : ST_AcsiVector
VAR_INPUT
    mag : ST_AcsiAnalogueValue;
    ang : ST_AcsiAnalogueValue;
END_VAR
```

Inputs

Name	Type	Description
mag	ST_AcsiAnalogueValue [▶ 531]	Data element "mag" to be set.
ang	ST_AcsiAnalogueValue [▶ 531]	Data element "ang" to be set.

Return value

Name	Type	Description
AcsiVector	ST_AcsiVector [▶ 545]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_AcsiVector : BOOL
VAR
    vector : ST_AcsiVector;
END_VAR
```

```
vector:=AcsiVector(mag:=AcsiAnalogueValue(0, 0), ang:=AcsiAnalogueValue(0, 0));
Sample_AcsiVector:=TRUE;
```

6.4.24 AnyBuffer_To_OctetString

Converts and copies the data of any variable into the octet string or one-dimensional byte array.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION AnyBuffer_To_OctetString : UDINT
VAR_IN_OUT
    target     : ARRAY[*] OF BYTE;
END_VAR
VAR_INPUT
    pSource    : PVOID;
    cbSource   : UDINT;
    bSwapBytes : BOOL;
END_VAR
VAR_OUTPUT
    bUnderflow : BOOL;
    bOverflow  : BOOL;
END_VAR
```

Inputs/Outputs

Name	Type	Description
target	ARRAY[*] OF BYTE	Target variable.

Inputs

Name	Type	Description
pSource	PVOID	Address pointer of the source variable.
cbSource	UDINT	Byte size of the source variable.
bSwapBytes	BOOL	With TRUE the data bytes are exchanged.

Outputs

Name	Type	Description
bUnderflow	BOOL	If TRUE then the source variable is smaller than the target variable.
bOverflow	BOOL	If TRUE then the source variable is greater than the target variable.

Return value

Name	Type	Description
AnyBuffer_To_OctetString	UDINT	Number of successfully copied data bytes.

Example

```
METHOD FINAL Sample_AnyBuffer_To_OctetString : BOOL
VAR
    sValue     : STRING:='0123456789';
    ui64       : LWORD:=16#3031323334353637;
    ui32       : UDINT:=16#30313233;
    o6         : T_OCTET6;
    o8         : T_OCTET8;
    o64        : T_OCTET64;
    size       : UDINT;
    bOverflow  : BOOL;
    bUnderflow : BOOL;
END_VAR
size:=AnyBuffer_To_OctetString(target:=o6, pSource:=ADR(sValue), cbSource:=SIZEOF(sValue), bSwapByte
s:=FALSE, bUnderflow=>bUnderflow, bOverflow=>bOverflow);(* size:=6 *)
size:=AnyBuffer_To_OctetString(target:=o8, pSource:=ADR(ui64), cbSource:=SIZEOF(ui64), bSwapBytes:=T
RUE, bUnderflow=>bUnderflow, bOverflow=>bOverflow);(* size:=8 *)
size:=AnyBuffer_To_OctetString(target:=o64, pSource:=ADR(ui32), cbSource:=SIZEOF(ui32), bSwapBytes:=
TRUE, bUnderflow=>bUnderflow, bOverflow=>bOverflow);(* size:=4 *)
Sample_AnyBuffer_To_OctetString:=TRUE;
```

6.4.25 BinaryTime

Initializes the data elements of the structure: T_BinaryTime.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION BinaryTime : T_BinaryTime
VAR_INPUT
    timeOfDay : TOD;
    day       : WORD;
END_VAR
```

Inputs

Name	Type	Description
timeOfDay	TOD	Number of milliseconds since midnight of the current day.
day	WORD	Number of days since January 1, 1984.

Return value

Name	Type	Description
BinaryTime	T_BinaryTime [▶ 556]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_BinaryTime : BOOL
VAR
    t: T_BinaryTime;
END_VAR
```

```
t:=BinaryTime(timeOfDay:=TOD#12:00:00.000, day:=DATE_To_BinaryTime6Day(in:=D#2023-01-01));
Sample_BinaryTime:=(t.timeOfDay = TOD#12:00:00.000) AND (t.day = 14245);
```

6.4.26 BinaryTime_To_SystemTime

Converts the time information of type T_BinaryTime [▶ 556] (binary time format) to the time information of type TIMESTRUCT [▶ 531] (system time format). In case of conversion error the output variable "bOverflow" has the value "TRUE" and the returned system time value has the date:1970-01-01 and the time: 0h0m0s0ms.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION BinaryTime_To_SystemTime : TIMESTRUCT
VAR_INPUT
    in        : T_BinaryTime;
END_VAR
VAR_OUTPUT
    bOverflow : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	T_BinaryTime [▶ 556]	The time information to be converted as a Binary-Time structure.

Outputs

Name	Type	Description
bOverflow	BOOL	Date overflow if "TRUE". In this case, the binary time to be converted has a date (T_BinaryTime [▶ 556].day member) greater than 2106-02-07.

Return value

Name	Type	Description
BinaryTime_To_SystemTime	TIMESTRUCT [▶ 531]	Converted time information as a System-Time structure.

Sample

```
PROGRAM MAIN
VAR
    tSystem   : TIMESTRUCT;
    tBinary   : T_BinaryTime;
    bOverflow : BOOL;
END_VAR
```

```
tSystem.wYear:=2022;
tSystem.wMonth:=9;
tSystem.wDay:=16;
tSystem.wHour:=12;
tSystem.wMinute:=52;
tSystem.wSecond:=4;
tSystem.wMilliseconds:=500;
tBinary:=SystemTime_To_BinaryTime(in:=tSystem, bOverflow=>bOverflow);
(* Returns tBinary.timeOfDay = TOD#12:52:04.500 and tBinary.day = 14138 *)

tBinary.day:=1;
tBinary.timeOfDay:=TOD#01:30:15.500;
tSystem:=BinaryTime_To_SystemTime(in:=tBinary, bOverflow=>bOverflow);(* Returns system-
time date: 1984-01-02 and time: 01h30m15s500ms*)
```

6.4.27 BinaryTime6_To_LWORD

Converts the components of the structured type T_BinaryTime [▶ 556] into binary data of the type LWORD. The coding of the components in LWORD corresponds to the TwinCAT Binary-Time LWORD format specification [▶ 772].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION BinaryTime6_To_LWORD : LWORD(0.. 16#FFFFFFFFFFFF);
VAR_INPUT
    in     : T_BinaryTime;
END_VAR
VAR_OUTPUT
    bError : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	T_BinaryTime [▶ 556]	The time information to be converted as a Binary-Time structure.

Outputs

Name	Type	Description
bError	BOOL	The output is "TRUE" if the value of the in.timeOfDay component is greater than "TOD#23:59:59,999". In this case the function returns the value "0".

Return value

Name	Type	Description
BinaryTime_To_LWORD	LWORD(0..16#FFFFFFFFFFFF);	Converted time information as Binary-Time binary data.

Sample

```
METHOD BinaryTime6_ToFrom_LWORD : BOOL
VAR_INPUT
    fbIED : REFERENCE TO FB_IED;
END_VAR
VAR
    tValue : T_BinaryTime;
    sValue : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

tValue := LWORD_To_BinaryTime6(0);
sValue := BinaryTime6_To_String(tValue);(* sValue := 'BT#1984-01-01-00:00:00' *)
IF fbIED.Relay.LLN0.brcb1.TimeOfEntry.nValue <> BinaryTime6_To_LWORD(tValue) THEN
    tValue := fbIED.Relay.LLN0.brcb1.TimeOfEntry.tValue;
END_IF

tValue := LWORD_To_BinaryTime6(1);
sValue := BinaryTime6_To_String(tValue);(* sValue := 'BT#1984-01-01-00:00:00.001'*)
IF fbIED.Relay.LLN0.brcb2.TimeOfEntry.nValue <> BinaryTime6_To_LWORD(tValue) THEN
    tValue := fbIED.Relay.LLN0.brcb2.TimeOfEntry.tValue;
END_IF

tValue := String_To_BinaryTime6('BT#2017-03-21-11:22:33.44');
sValue := BinaryTime6_To_String(tValue);(* sValue := 'BT#2017-03-21-11:22:33.440'*)
IF fbIED.Relay.LLN0.brcb3.TimeOfEntry.nValue <> BinaryTime6_To_LWORD(tValue) THEN
    tValue := fbIED.Relay.LLN0.brcb3.TimeOfEntry.tValue;
END_IF

BinaryTime6_ToFrom_LWORD := TRUE;
```

6.4.28 BinaryTime6_To_String

Converts the components of the structured type T_BinaryTime [▶ 556] into a formatted Binary-Time string. The resulting string has the following structure: 'BT#[YYYY-MM-DD-]hh:mm:ss[.nnn]'. It corresponds to the TwinCAT Binary-Time string format specification [▶ 772].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION BinaryTime6_To_String : STRING(26)
VAR_INPUT
    in     : T_BinaryTime;
END_VAR
VAR_OUTPUT
    bError : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	T_BinaryTime [▶ 556]	The time information to be converted as a Binary-Time structure.

Outputs

Name	Type	Description
bError	BOOL	The output is "TRUE" if the value of the in.day component is greater than "16#AE35" (this corresponds to February 7, 2106). In this case the function returns an empty string.

Return value

Name	Type	Description
BinaryTime6_To_String	STRING(26)	Converted time information as a Binary-Time string.

Sample

```
METHOD BinaryTime6_ToFrom_String : BOOL
VAR_INPUT
    fbIED  : REFERENCE TO FB_IED;
END_VAR
VAR
    tValue : T_BinaryTime;
    sValue : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

tValue := String_To_BinaryTime6('BT#01:02:03');
sValue := BinaryTime6_To_String(tValue); (* sValue := 'BT#1984-01-01-01:02:03' *)
IF fbIED.Relay.LLN0.brcb1.TimeOfEntry.sValue <> sValue THEN
    tValue := fbIED.Relay.LLN0.brcb1.TimeOfEntry.tValue;
END_IF

tValue := String_To_BinaryTime6('BT#04:05:06.7');
sValue := BinaryTime6_To_String(tValue);(* sValue := 'BT#1984-01-01-04:05:06.700' *)
IF fbIED.Relay.LLN0.brcb2.TimeOfEntry.sValue <> sValue THEN
    tValue := fbIED.Relay.LLN0.brcb2.TimeOfEntry.tValue;
END_IF

tValue := String_To_BinaryTime6('BT#2017-01-18-11:12:13');
sValue := BinaryTime6_To_String(tValue);(* sValue := 'BT#2017-01-18-11:12:13' *)
IF fbIED.Relay.LLN0.brcb3.TimeOfEntry.sValue <> sValue THEN
    tValue := fbIED.Relay.LLN0.brcb3.TimeOfEntry.tValue;
END_IF

BinaryTime6_ToFrom_String := TRUE;
```

6.4.29 BinaryTime6Day_To_Date

Converts the number of days since January 1, 1984 from type WORD to a date of type DATE. The function returns the max. representable DATE value "D#2106-02-07" if the day to be converted is greater than the value "16#AE35".

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION BinaryTime6Day_To_Date : DATE
VAR_INPUT
    in     : WORD(0..16#AE35);
END_VAR
VAR_OUTPUT
    bError : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	WORD(0..16#AE35)	The number of days since January 1, 1984 to be converted. This value corresponds to the value of the T_BinaryTime [▶ 556].day component.

Outputs

Name	Type	Description
bError	BOOL	The output is "TRUE" if the input value is greater than "16#AE35" (February 7, 2106). In this case the function returns the value "D#1970-01-01".

Return value

Name	Type	Description
BinaryTime6Day_To_Date	DATE	Date in the DATE format.

Sample

```
METHOD FINAL BinaryTime6Day_ToFrom_Date : BOOL
VAR
    tValue : T_BinaryTime;
    d      : DATE;
    sValue : STRING;
END_VAR
```

```
tValue.timeOfDay:=TOD#00:00:00.125;
tValue.day:=Date_To_BinaryTime6Day(D#1984-01-01);(* tValue.day:=16#0000 *)
sValue:=BinaryTime6_To_String(tValue);(* sValue:='BT#1984-01-01-00:00:00.125' *)
d:=BinaryTime6Day_To_Date(tValue.day);(* d:=D#1984-01-01 *)

tValue.timeOfDay:=TOD#01:02:03.125;
tValue.day:=Date_To_BinaryTime6Day(D#1984-01-02);(* tValue.day:=16#0001 *)
sValue:=BinaryTime6_To_String(tValue);(* sValue:='BT#1984-01-02-01:02:03.125' *)
d:=BinaryTime6Day_To_Date(tValue.day);(* d:=D#1984-01-02 *)

tValue.timeOfDay:=TOD#04:05:06.125;
tValue.day:=Date_To_BinaryTime6Day(D#2018-03-21);(* tValue.day:=16#30D2 *)
sValue:=BinaryTime6_To_String(tValue);(* sValue:='BT#2018-03-21-04:05:06.125' *)
d:=BinaryTime6Day_To_Date(tValue.day);(* d:=D#2018-03-21 *)

tValue.timeOfDay:=TOD#07:08:09;
tValue.day:=tValue.day + 7;(* add 7 days *)
sValue:=BinaryTime6_To_String(tValue);(* sValue:='BT#2018-03-28-07:08:09' *)
d:=BinaryTime6Day_To_Date(tValue.day);(* d:=D#2018-03-28 *)

BinaryTime6Day_ToFrom_Date:=TRUE;
```

6.4.30 BYTE_To_UtcTimeQuality

Converts the binary data of the type BYTE into the components of the structured type T_UtcTimeQuality [▶ 560]. The coding of the components in BYTE corresponds to the TwinCAT UTC-Time-Quality-BYTE format specification [▶ 775].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION BYTE_To_UtcTimeQuality : T_UtcTimeQuality
VAR_INPUT
    in    : BYTE;
END_VAR
```

Inputs

Name	Type	Description
in	BYTE	The information to be converted as UTC-Time-Quality binary data.

Return value

Name	Type	Description
BYTE_To_UtcTimeQuality	T_UtcTimeQuality [▶ 560]	Converted information as UTC-Time-Quality structure.

Sample

```
METHOD UtcTimeQuality_ToFrom_Byte : BOOL
VAR_INPUT
    fbIED    : REFERENCE TO FB_IED;
END_VAR
VAR
    Quality  : T_UtcTimeQuality;
    sQuality : STRING;
    u8       : BYTE;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

Quality := Byte_To_UtcTimeQuality(0);
u8 := UtcTimeQuality_To_Byte(Quality);(* u8 := 2#00000000 => 0 bits accuracy *)
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|00' *)
IF UtcTimeQuality_To_Byte(fbIED.Relay.LLN0.Beh.t.Quality) <> u8 THEN
    Quality := fbIED.Relay.LLN0.Beh.t.Quality;
END_IF

Quality := String_To_UtcTimeQuality('UQ#000|03');
u8 := UtcTimeQuality_To_Byte(Quality);(* u8 := 2#11000000 => 3 bits accuracy *)
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|03' *)
IF UtcTimeQuality_To_Byte(fbIED.Relay.LLN0.Health.t.Quality) <> u8 THEN
    Quality := fbIED.Relay.LLN0.Health.t.Quality;
END_IF

Quality := String_To_UtcTimeQuality('UQ#001|05');
u8 := UtcTimeQuality_To_Byte(Quality);
(* u8 := 2#10100100 => 5 bits accuracy, ClockNotSynchronized := 1 *)
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#001|05' *)
IF UtcTimeQuality_To_Byte(fbIED.Relay.LLN0.Mod_.t.Quality) <> u8 THEN
    Quality := fbIED.Relay.LLN0.Mod_.t.Quality;
END_IF

Quality := String_To_UtcTimeQuality('UQ#000|31');
u8 := UtcTimeQuality_To_Byte(Quality);(* u8 := 2#11111000 => unspecified accuracy *)
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|31' *)
IF UtcTimeQuality_To_Byte(fbIED.Relay.MMXU1.phV.phsA.t.Quality) <> u8 THEN
    Quality := fbIED.Relay.MMXU1.phV.phsA.t.Quality;
END_IF

UtcTimeQuality_ToFrom_Byte := TRUE;
```

6.4.31 CltsEveryBool

This function checks whether all elements of a boolean array have the same value as the bTest function input parameter.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION CltsEveryBool : BOOL
VAR_INPUT
    bTest: BOOL;
END_VAR
VAR_IN_OUT
```

```
    in: ARRAY[*] OF BOOL;
END_VAR
VAR_OUTPUT
    index: DINT;
END_VAR
```

Inputs

Name	Type	Description
bTest	BOOL	Test value that is compared with all array elements.

Inputs/outputs

Name	Type	Description
in	ARRAY[*] OF BOOL	The boolean array variable to check.

Outputs

Name	Type	Description
index	DINT	Index number of the first array element whose value is not equal to the bTest input parameter. This value is 0 if the function return parameter is "TRUE".

Return value

Name	Type	Description
CltsEveryBool	BOOL	"TRUE" if all array elements have the same value as the bTest function input parameter. "FALSE" if at least one array element has an unequal value like the bTest function input parameter.

Sample

```
PROGRAM MAIN
VAR
    a: ARRAY[1..5] OF BOOL;
    index: DINT;
    bResult: BOOL;
END_VAR
```

```
a[1]:=FALSE;
a[2]:=FALSE;
a[3]:=TRUE;
a[4]:=FALSE;
a[5]:=TRUE;
bResult:=CltsEveryBool(bTest:=TRUE, in:=a, index=>index);(* return value is FALSE, index = 1 *)
bResult:=CltsEveryBool(bTest:=FALSE, in:=a, index=>index);(* return value is FALSE, index = 3 *)
a[1]:=TRUE;
a[2]:=TRUE;
a[4]:=TRUE;
bResult:=CltsEveryBool(bTest:=TRUE, in:=a, index=>index);(* return value is TRUE, index = 0 *)
```

6.4.32 CltsSomeBool

This function checks whether at least one array element of a boolean array has the same value as the bTest function input parameter.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION CltsSomeBool : BOOL
VAR_INPUT
    bTest: BOOL;
END_VAR
VAR_IN_OUT
```

```
    in: ARRAY[*] OF BOOL;
END_VAR
VAR_OUTPUT
    index: DINT;
END_VAR
```

Inputs

Name	Type	Description
bTest	BOOL	Test value that is compared with all array elements.

Inputs/outputs

Name	Type	Description
in	ARRAY[*] OF BOOL	The boolean array variable to check.

Outputs

Name	Type	Description
index	DINT	Index of the first array element whose value is equal to the bTest function input parameter. This value is 0 if the function return value is "FALSE".

Return value

Name	Type	Description
CltsSomeBool	BOOL	"TRUE" if at least one array element has the same value as the bTest function input parameter. "FALSE" if none of the array elements has the same value as the bTest function input parameter.

Sample

```
PROGRAM MAIN
VAR
    a: ARRAY[1..5] OF BOOL;
    index: DINT;
    bResult: BOOL;
END_VAR
```

```
a[1]:=FALSE;
a[2]:=FALSE;
a[3]:=FALSE;
a[4]:=FALSE;
a[5]:=FALSE;
bResult:=CltsSomeBool(bTest:=TRUE, in:=a, index=>index);(* return value is FALSE, index = 0 *)
a[1]:=FALSE;
a[2]:=FALSE;
a[3]:=TRUE;
a[4]:=FALSE;
a[5]:=TRUE;
bResult:=CltsSomeBool(bTest:=TRUE, in:=a, index=>index);(* return value is TRUE, index = 3 *)
bResult:=CltsSomeBool(bTest:=FALSE, in:=a, index=>index);(* return value is TRUE, index = 1 *)
```

6.4.33 DATE_and_TOD_To_DT

Combines the date of type DATE and the time of type TIME_OF_DAY (TOD) to the time information of type DATE_AND_TIME (DT).

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION DATE_and_TOD_To_DT : DT
VAR_INPUT
    d             : DATE;
```

```
    t             : TOD;
END_VAR
VAR_OUTPUT
    wMilliseconds : WORD;
END_VAR
```

Inputs

Name	Type	Description
d	DATE	Date.
t	TOD	Time.

Outputs

Name	Type	Description
wMilliseconds	WORD	Millisecond rest of the TOD time which could not be converted to the DATE_AND_TIME type. The DATE_AND_TIME type has the max. resolution of one second.

Return value

Name	Type	Description
DATE_and_TOD_To_DT	DT	Combined time information of type DATE_AND_TIME.

Sample

```
PROGRAM MAIN
VAR
    d             : DATE:= D#2022-09-15;
    t             : TOD:=TOD#17:24:56.789;
    dateAndTime   : DT;
    wMilliseconds : WORD;
END_VAR
```

```
d:=D#2022-09-15;
t:=TOD#17:24:56.789;
dateAndTime:=DATE_and_TOD_To_DT(d:=d, t:=t, wMilliseconds=>wMilliseconds);
(* Result is: DT#2022-09-15-17:24:56, wMilliseconds = 789 *)

d:=D#2022-08-16;
t:=TOD#10:00:00.000;
dateAndTime:=DATE_and_TOD_To_DT(d:=d, t:=t, wMilliseconds=>wMilliseconds);
(* Result is: DT#2022-08-16-10:00:00, wMilliseconds = 0 *)
```

6.4.34 Date_To_BinaryTime6Day

Converts a date of type: DATE to the number of days since January 1, 1984 of type WORD. The function returns the value "0" if the date to be converted is before January 1, 1984.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION Date_To_BinaryTime6Day : WORD(0..16#AE35)
VAR_INPUT
    in     : DATE := D#1984-01-01;
END_VAR
VAR_OUTPUT
    bError : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	DATE	The date to be converted.

Outputs

Name	Type	Description
bError	BOOL	The output is "TRUE" if the input value is smaller than "D#1984-01-01". In this case the function returns the value "0".

Return value

Name	Type	Description
Date_To_BinaryTime6Day	WORD	Converted number of days since January 1, 1984. This value corresponds to the value of the T_BinaryTime [▶ 556].day component.

Sample

```
METHOD FINAL BinaryTime6Day_ToFrom_Date : BOOL
VAR
    tValue : T_BinaryTime;
    d      : DATE;
    sValue : STRING;
END_VAR
```

```
tValue.timeOfDay:=TOD#00:00:00.125;
tValue.day:=Date_To_BinaryTime6Day(D#1984-01-01);(* tValue.day:=16#0000 *)
sValue:=BinaryTime6_To_String(tValue);(* sValue:='BT#1984-01-01-00:00:00.125' *)
d:=BinaryTime6Day_To_Date(tValue.day);(* d:=D#1984-01-01 *)

tValue.timeOfDay:=TOD#01:02:03.125;
tValue.day:=Date_To_BinaryTime6Day(D#1984-01-02);(* tValue.day:=16#0001 *)
sValue:=BinaryTime6_To_String(tValue);(* sValue:='BT#1984-01-02-01:02:03.125' *)
d:=BinaryTime6Day_To_Date(tValue.day);(* d:=D#1984-01-02 *)

tValue.timeOfDay:=TOD#04:05:06.125;
tValue.day:=Date_To_BinaryTime6Day(D#2018-03-21);(* tValue.day:=16#30D2 *)
sValue:=BinaryTime6_To_String(tValue);(* sValue:='BT#2018-03-21-04:05:06.125' *)
d:=BinaryTime6Day_To_Date(tValue.day);(* d:=D#2018-03-21 *)

tValue.timeOfDay:=TOD#07:08:09;
tValue.day:=tValue.day + 7;(* add 7 days *)
sValue:=BinaryTime6_To_String(tValue);(* sValue:='BT#2018-03-28-07:08:09' *)
d:=BinaryTime6Day_To_Date(tValue.day);(* d:=D#2018-03-28 *)

BinaryTime6Day_ToFrom_Date:=TRUE;
```

6.4.35 FillOctetString

Overwrites all octets of an octet string with a given value.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION FillOctetString : UDINT
VAR_IN_OUT
    o         : ARRAY[*] OF BYTE;
END_VAR
VAR_INPUT
    fillOctet : BYTE;
END_VAR
```

/ Inputs/Outputs

Name	Type	Description
o	ARRAY[*] OF BYTE	Octet string to be overwritten.

Inputs

Name	Type	Description
fillOctet	BYTE	Value with which the octets are to be overwritten.

Return value

Name	Type	Description
FillOctetString	UDINT	Number of octets overwritten.

Example

```
METHOD FINAL Sample_FillOctetString : BOOL
VAR
    o6  : T_OCTET6:=[6(0)];
    o8  : T_OCTET8:=[8(0)];
    o64 : T_OCTET64:=[64(0)];
    size: UDINT;
END_VAR
```

```
size:=FillOctetString(o:=o6, fillOctet:=16#FF);(* size:=6, all octets: 16#FF *)
size:=FillOctetString(o:=o8, fillOctet:=16#BB);(* size:=8, all octets: 16#BB *)
size:=FillOctetString(o:=o64, fillOctet:=16#CC);(* size:=64, all octets: 16#CC *)
Sample_FillOctetString:=TRUE;
```

6.4.36 INT24

Initializes a signed 24-bit number of the type: T_INT24.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION INT24 : T_INT24
VAR_INPUT
    Hi : BYTE;
    Mi : BYTE;
    Lo : BYTE;
END_VAR
```

Inputs

Name	Type	Description
Hi	BYTE	Top 8 bits to be set.
Mi	BYTE	Middle 8 bits to be set.
Lo	BYTE	Lowest 8 bits to be set.

Return value

Name	Type	Description
INT24	T_INT24 [▶ 557]	24-bit number.

Example

```
METHOD FINAL Sample_INT24 : BOOL
VAR
    iValue: T_INT24;
END_VAR
```

```
iValue:=INT24(16#AB, 16#CD, 16#EF);(* iValue:=16#ABCDEF *)
Sample_INT24:=TRUE;
```

6.4.37 INT128

Initializes a signed 128-bit number of the type: T_INT128.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION INT128 : T_INT128
VAR_INPUT
    Hi : LWORD;
    Lo : LWORD;
END_VAR
```

Inputs

Name	Type	Description
Hi	LWORD	Upper 64 bits to be set.
Lo	LWORD	Lower 64 bits to be set.

Return value

Name	Type	Description
INT128	T_INT128 [▶ 557]	128-bit number.

Example

```
METHOD FINAL Sample_INT128 : BOOL
VAR
    iValue: T_INT128;
END_VAR
```

```
iValue:=INT128(16#0000000000ABCDEF, 16#1234567890000001);
(* iValue:=16#0000000000ABCDEF_1234567890000001 *)
Sample_INT128:=TRUE;
```

6.4.38 LTIME_To_UtcTimeFractionOfSecond

Converts the fraction of a second of the type LTIME into the type T_UINT24 [▶ 559]. The converted value corresponds to the value of the T_UtcTime [▶ 560].fractionOfSecond component. The LTIME value is coded in 24 bits in a special way. See: Coding of the UTC-Time.fractionOfSecond0..23-bits [▶ 773]. The input value must be <= LTIME#999ms999us9999ns. The maximum resolution for the conversion is approx. 60 ns.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION LTIME_To_UtcTimeFractionOfSecond : T_UINT24
VAR_INPUT
    in          : LTIME;
END_VAR
VAR_OUTPUT
    bOverflow   : BOOL;
    secondsOver : LWORD;
    nanosec     : LWORD;
    attosec     : LWORD;
END_VAR
```

Inputs

Name	Type	Description
in	LTIME	The fraction of a second to be converted.

Outputs

Name	Type	Description
bOverflow	BOOL	Additional output parameter, which provides information about the input value overflow. This value is "TRUE" if the input value is >= LTIME#1s, otherwise it is "FALSE".
secondsOver	LWORD	Additional output parameter that returns the number of overflow seconds.
nanosec	LWORD	Additional output parameter that returns the input value in nanoseconds.
attosec	LWORD	Additional output parameter that returns the input value in attoseconds.

Return value

Name	Type	Description
LTIME_To_UtcTimeFractionOfSecond	T_UINT24 [▶ 559]	Converted fraction of a second as binary data.

Sample

```
METHOD UtcTimeFractionOfSecond_ToFrom_LTIME : BOOL
VAR
    tValue    : T_UtcTime;
    sValue    : STRING;
    tFos      : LTIME;
    nanosec   : LWORD;
END_VAR
```

```
tValue.quality := String_To_UtcTimeQuality('UQ#000|24');
tValue.secondSinceEpoch := DT#2018-03-28-12:13:14;

tValue.fractionOfSecond := LTIME_To_UtcTimeFractionOfSecond(LTIME#0NS);
(* tValue.fractionOfSecond := [16#00, 16#00, 16#00], 0 significant bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2018-03-28-12:13:14.000000000|000|24' *)
tFos := UtcTimeFractionOfSecond_To_LTIME(tValue.fractionOfSecond, nanosec=>nanosec);
(* tFos := LTIME#0ns, nanosec := 0 *)

tValue.fractionOfSecond := LTIME_To_UtcTimeFractionOfSecond(LTIME#750MS);
(* tValue.fractionOfSecond := [16#03, 16#00, 16#00], 3 significant bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2018-03-28-12:13:14.750000000|000|24' *)
tFos := UtcTimeFractionOfSecond_To_LTIME(tValue.fractionOfSecond, nanosec=>nanosec);
(* tFos := LTIME#750ms, nanosec := 750000000 *)

tValue.fractionOfSecond := LTIME_To_UtcTimeFractionOfSecond(LTIME#968MS750US);
(* tValue.fractionOfSecond := [16#1F, 16#00, 16#00], 5 significant bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2018-03-28-12:13:14.968750000|000|24' *)
tFos := UtcTimeFractionOfSecond_To_LTIME(tValue.fractionOfSecond, nanosec=>nanosec);
(* tFos := LTIME#968ms750us, nanosec := 968750000 *)

tValue.fractionOfSecond := LTIME_To_UtcTimeFractionOfSecond(LTIME#998MS46US875NS);
(* tValue.fractionOfSecond := [16#FF, 16#01, 16#00], 9 significant bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2018-03-28-12:13:14.998046875|000|24' *)
tFos := UtcTimeFractionOfSecond_To_LTIME(tValue.fractionOfSecond, nanosec=>nanosec);
(* tFos := LTIME#998ms46us875ns, nanosec := 998046875 *)

tValue.fractionOfSecond := LTIME_To_UtcTimeFractionOfSecond(LTIME#999MS999US999NS);
(* tValue.fractionOfSecond := [16#FF, 16#FF, 16#FF], 24 significant bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2018-03-28-12:13:14.999999940|000|24' *)
tFos := UtcTimeFractionOfSecond_To_LTIME(tValue.fractionOfSecond, nanosec=>nanosec);
(* tFos := LTIME#999ms999us940ns, nanosec := 999999940 *)

UtcTimeFractionOfSecond_ToFrom_LTIME := TRUE;
```

6.4.39 LWORD_To_BinaryTime6

Converts the binary data of the type LWORD into the components of the structured type T_BinaryTime [▶ 556]. The coding of the components in LWORD corresponds to the TwinCAT Binary-Time LWORD format specification [▶ 772].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION LWORD_To_BinaryTime6 : T_BinaryTime
VAR_INPUT
    in      : LWORD(0..16#FFFFFFFFFFFF);
END_VAR
VAR_OUTPUT
    bError : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	LWORD(0..16#FFFFFFFFFFFF)	The time information to be converted as Binary-Time binary data.

Outputs

Name	Type	Description
bError	BOOL	The output is "TRUE" if the value of bits 0..31 (timeOfDay component) is greater than "TOD#23:59:59.999". In this case the function returns the value "STRUCT(timeOfDay := TOD#00:00:00, day := 0)".

Return value

Name	Type	Description
LWORD_To_BinaryTime6	T_BinaryTime [▶ 556]	Converted time information as Binary-Time structure.

Sample

```
METHOD BinaryTime6_ToFrom_LWORD : BOOL
VAR_INPUT
    fbIED : REFERENCE TO FB_IED;
END_VAR
VAR
    tValue : T_BinaryTime;
    sValue : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

tValue := LWORD_To_BinaryTime6(0);
sValue := BinaryTime6_To_String(tValue);(* sValue := 'BT#1984-01-01-00:00:00' *)
IF fbIED.Relay.LLN0.brcb1.TimeOfEntry.nValue <> BinaryTime6_To_LWORD(tValue) THEN
    tValue := fbIED.Relay.LLN0.brcb1.TimeOfEntry.tValue;
END_IF

tValue := LWORD_To_BinaryTime6(1);
sValue := BinaryTime6_To_String(tValue);(* sValue := 'BT#1984-01-01-00:00:00.001'*)
IF fbIED.Relay.LLN0.brcb2.TimeOfEntry.nValue <> BinaryTime6_To_LWORD(tValue) THEN
    tValue := fbIED.Relay.LLN0.brcb2.TimeOfEntry.tValue;
END_IF

tValue := String_To_BinaryTime6('BT#2017-03-21-11:22:33.44');
sValue := BinaryTime6_To_String(tValue);(* sValue := 'BT#2017-03-21-11:22:33.440'*)
IF fbIED.Relay.LLN0.brcb3.TimeOfEntry.nValue <> BinaryTime6_To_LWORD(tValue) THEN
    tValue := fbIED.Relay.LLN0.brcb3.TimeOfEntry.tValue;
```

```
END_IF

BinaryTime6_ToFrom_LWORD := TRUE;
```

6.4.40 LWORD_TO_MACADDR

Converts the binary data of type LWORD to type ETHERNET_ADDRESS. The LWORD bytes are converted and swapped during conversion according to the following scheme: The least significant byte becomes the last and the most significant converted byte becomes the first octet of the MAC address.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION LWORD_TO_MACADDR : ETHERNET_ADDRESS
VAR_INPUT
    in: LWORD(0..16#FFFFFFFFFFFF);
END_VAR
```

Inputs

Name	Type	Description
in	LWORD(0..16#FFFFFFFFFFFF)	The LWORD value to be converted.

Return value

Name	Type	Description
LWORD_TO_MACADDR	ETHERNET_ADDRESS [▶ 528]	Octet string with the maximum length of 8 octets.

Example

```
METHOD FINAL Sample_LWORD_TO_MACADDR : BOOL
VAR
    mac: ETHERNET_ADDRESS;
END_VAR
```

```
mac:=LWORD_TO_MACADDR(16#00B0D063C226);
(* mac:=00-B0-D0-63-C2-26 *)
Sample_LWORD_TO_MACADDR:=TRUE;
```

6.4.41 LWORD_TO_OCTET6

Converts the binary data of type LWORD to type T_OCTET6. The two most significant bytes are ignored and not converted. The remaining LWORD bytes are converted and swapped during conversion according to the following scheme: the least significant byte becomes the last and the most significant converted byte becomes the first octet of the octet string.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION LWORD_TO_OCTET6 : T_OCTET6
VAR_INPUT
    in: LWORD(0..16#FFFFFFFFFFFF);
END_VAR
```

Inputs

Name	Type	Description
in	LWORD(0..16#FFFFFFFFFFFF)	The LWORD value to be converted.

Return value

Name	Type	Description
LWORD_TO_OCTET6	T_OCTET6 [▶ 558]	Octet string with the maximum length of 6 octets.

Example

```
METHOD FINAL Sample_LWORD_TO_OCTET6 : BOOL
VAR
    o: T_OCTET6;
END_VAR
```

```
o:=LWORD_TO_OCTET6(16#010CCD010001);
(* o:=[16#01,16#0C,16#CD,16#01,16#00,16#01] *)
Sample_LWORD_TO_OCTET6:=TRUE;
```

6.4.42 LWORD_TO_OCTET8

Converts the binary data of type LWORD to type T_OCTET8. The LWORD bytes are converted and swapped during the conversion according to the following scheme: the least significant byte becomes the last and the most significant converted byte becomes the first octet of the octet string.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION LWORD_TO_OCTET8 : T_OCTET8
VAR_INPUT
    in: LWORD;
END_VAR
```

Inputs

Name	Type	Description
in	LWORD	The LWORD value to be converted.

Return value

Name	Type	Description
LWORD_TO_OCTET8	T_OCTET8 [▶ 558]	Octet string with the maximum length of 8 octets.

Example

```
METHOD FINAL Sample_LWORD_TO_OCTET8 : BOOL
VAR
    o: T_OCTET8;
END_VAR
```

```
o:=LWORD_TO_OCTET8(16#0102030405060708);
(* o:=[16#01,16#02,16#03,16#04,16#05,16#06,16#07,16#08] *)
Sample_LWORD_TO_OCTET8:=TRUE;
```

6.4.43 LWORD_To_UtcTime

Converts the binary data of the type LWORD into the components of the structured type T_UtcTime [▶ 560]. The coding of the components in LWORD corresponds to the TwinCAT UTC-Time-LWORD format specification [▶ 773].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION LWORD_To_UtcTime : T_UtcTime
VAR_INPUT
    in : LWORD;
END_VAR
```

Inputs

Name	Type	Description
in	LWORD	The time information to be converted as UTC-Time binary data.

Return value

Name	Type	Description
LWORD_To_UtcTime	T_UtcTime [▶ 560]	Converted time information as UTC-Time structure.

Sample

```
METHOD UtcTime_ToFrom_LWORD : BOOL
VAR
    tValue : T_UtcTime;
    sValue : STRING;
END_VAR
```

```
tValue := LWORD_To_UtcTime(0);
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#1970-01-01-00:00:00.000000000|000|0' *)

IF UtcTime_To_LWORD(tValue) = 0 THEN
    tValue.secondSinceEpoch := DT#2000-01-01-00:00:0;
    tValue.fractionOfSecond := LTIME_TO_UtcTimeFractionOfSecond( LTIME#0MS );
    tValue.quality.LeapSecondsKnown := FALSE;
    tValue.quality.ClockFailure := FALSE;
    tValue.quality.ClockNotSynchronized := TRUE;
    Accuracy_To_UtcTimeQualityAccuracy( E_UtcTimeAccuracy._03,
                                        bAccuracy0=>tValue.quality.Accuracy0,
                                        bAccuracy1=>tValue.quality.Accuracy1,
                                        bAccuracy2=>tValue.quality.Accuracy2,
                                        bAccuracy3=>tValue.quality.Accuracy3,
                                        bAccuracy4=>tValue.quality.Accuracy4 );
    sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2000-01-01-00:00:00.000000000|001|3' *)
END_IF


UtcTime_ToFrom_LWORD := TRUE;
```

6.4.44 OctetString_To_AnyBuffer

Converts and copies the data from an octet string or a one-dimensional byte array to any variable.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION OctetString_To_AnyBuffer : UDINT
VAR_IN_OUT
    source     : ARRAY[*] OF BYTE;
END_VAR
VAR_INPUT
    pTarget    : PVOID;
    cbTarget   :UDINT;
    bSwapBytes : BOOL;
END_VAR
VAR_OUTPUT
    bUnderflow : BOOL;
    bOverflow  : BOOL;
END_VAR
```

Inputs/Outputs

Name	Type	Description
source	ARRAY[*] OF BYTE	Source variable

Inputs

Name	Type	Description
pTarget	PVOID	Address pointer of the target variable.
cbTarget	UDINT	Byte size of the target variable.
bSwapBytes	BOOL	With TRUE the data bytes are exchanged.

Outputs

Name	Type	Description
bUnderflow	BOOL	If TRUE then the source variable is smaller than the target variable.
bOverflow	BOOL	If TRUE then the source variable is greater than the target variable.

Return value

Name	Type	Description
OctetString_To_AnyBuffer	UDINT	Number of successfully copied data bytes.

Example

```
METHOD FINAL Sample_OctetString_To_AnyBuffer : BOOL
VAR
    sValue      : STRING;
    ui64        : LWORD;
    ui32        : UDINT;
    o6          : T_OCTET6:=[16#30, 16#31, 16#32, 16#33, 16#34, 16#35];
    o8          : T_OCTET8:=[16#30, 16#31, 16#32, 16#33, 16#34, 16#35, 16#36, 16#37];
    o64         : T_OCTET64:=[16#30, 16#31, 16#32, 16#33, 16#34, 16#35, 16#36, 16#37, 16#38, 16#39];
    size        : UDINT;
    bOverflow   : BOOL;
    bUnderflow : BOOL;
END_VAR
size:=OctetString_To_AnyBuffer(source:=o6, pTarget:=ADR(sValue), cbTarget:=SIZEOF(sValue), bSwapByte
s:=FALSE, bUnderflow=>bUnderflow, bOverflow=>bOverflow);(* size:=6 *)
size:=OctetString_To_AnyBuffer(source:=o8, pTarget:=ADR(ui64), cbTarget:=SIZEOF(ui64), bSwapBytes:=T
RUE, bUnderflow=>bUnderflow, bOverflow=>bOverflow);(* size:=8 *)
size:=OctetString_To_AnyBuffer(source:=o64, pTarget:=ADR(ui32), cbTarget:=SIZEOF(ui32), bSwapBytes:=
TRUE, bUnderflow=>bUnderflow, bOverflow=>bOverflow);(* size:=4 *)
Sample_OctetString_To_AnyBuffer:=(sValue = '012345') AND (ui64 = 16#3031323334353637) AND (ui32 = 16
#30313233);
```

6.4.45 SizeOfOctetString

Determines the byte size of any octet string (1-dimensional byte array).

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION SizeOfOctetString : UDINT
VAR_IN_OUT
    o : ARRAY[*] OF BYTE;
END_VAR
```

Inputs/Outputs

Name	Type	Description
o	ARRAY[*] OF BYTE	Octet string whose size is to be determined.

Return value

Name	Type	Description
SizeOfOctetString	UDINT	Octet string byte size.

Example

```
METHOD FINAL Sample_SizeOfOctetString : BOOL
VAR
    o6  : T_OCTET6:=[6(0)];
    o8  : T_OCTET8:=[8(0)];
    o64 : T_OCTET64:=[64(0)];
    size: UDINT;
END_VAR
```

```
size:=SizeOfOctetString(o:=o6);(* size:=6 *)
size:=SizeOfOctetString(o:=o8);(* size:=8 *)
size:=SizeOfOctetString(o:=o64);(* size:=64 *)
Sample_SizeOfOctetString:=TRUE;
```

6.4.46 Source_To_AcsiQualitySource

Converts the value of the enumeration type E_AcsiQualitySource [▶ 501] into a Boolean value. This output value corresponds to the Source-Bit component of the structured type ST_AcsiQuality [▶ 541].

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION Source_To_AcsiQualitySource : BOOL
VAR_INPUT
    in : E_AcsiQualitySource;
END_VAR
```

Inputs

Name	Type	Description
in	E_AcsiQualitySource [▶ 501]	The information to be converted as an enumeration type.

Return value

Name	Type	Description
Source_To_AcsiQualitySource	BOOL	Converted information as an ST_AcsiQuality [▶ 541].Source-Bit component.

Sample

```
METHOD AcsiQualitySource_ToFrom_Source : BOOL
VAR_INPUT
    fbIED  : REFERENCE TO FB_IED;
END_VAR
VAR
    qValue : ST_AcsiQuality;
    sValue : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

Validity_To_AcsiQualityValidity(E_AcsiQualityValidity.Good, bValidity0=>qValue.Validity0, bValidity1
=> qValue.Validity1 );
qValue.Overflow := FALSE;
qValue.OutOfRange := FALSE;
qValue.BadReference := FALSE;
qValue.Oscillatory := FALSE;
qValue.Failure := FALSE;
qValue.OldData := FALSE;
qValue.Inconsistent := FALSE;
qValue.Inaccurate := FALSE;
qValue.Source := Source_To_AcsiQualitySource(E_AcsiQualitySource.Substituted);
qValue.Test := TRUE;
qValue.OperatorBlocked := FALSE;
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#00|00000000|1|10' *)

IF fbIED.Relay.LLN0.Beh.q.eSource <> AcsiQualitySource_To_Source(qValue) THEN
     qValue := fbIED.Relay.LLN0.Beh.q.qValue;
```

```
END_IF

AcsiQualitySource_ToFrom_Source := TRUE;
```

6.4.47 String_To_AcsiQuality

Converts the quality information formatted as a string into the components of the structured type ST_AcsiQuality [▶ 541]. The string to be converted must be specified and has the following structure: 'Q#vv|ORBSFDCA|s|TB'. It corresponds to the TwinCAT Quality-String format specification [▶ 778].

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION String_To_AcsiQuality : ST_AcsiQuality
VAR_INPUT
    in     : STRING(18) := 'Q#00|00000000|0|00';
END_VAR
VAR_OUTPUT
    bError : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	STRING(18)	The information to be converted as a Quality string.

Outputs

Name	Type	Description
bError	BOOL	This output is "TRUE" if the input string could not be converted without error.

Return value

Name	Type	Description
String_To_AcsiQuality	ST_AcsiQuality [▶ 541]	Converted information as a Quality structure.

Sample

```
METHOD AcsiQuality_ToFrom_String : BOOL
VAR_INPUT
    fbIED  : REFERENCE TO FB_IED;
END_VAR
VAR
    qValue : ST_AcsiQuality;
    sValue : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

qValue := String_To_AcsiQuality('Q#00|00000000|0|00');
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#00|00000000|0|00' *)
IF fbIED.Relay.LLN0.Beh.q.sValue <> AcsiQuality_To_String(qValue) THEN
    qValue := fbIED.Relay.LLN0.Beh.q.qValue;
END_IF

qValue := String_To_AcsiQuality('Q#01|00110011|0|01');
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#01|00110011|0|01' *)
IF fbIED.Relay.LLN0.Health.q.sValue <> AcsiQuality_To_String(qValue) THEN
    qValue := fbIED.Relay.LLN0.Health.q.qValue;
END_IF

qValue := String_To_AcsiQuality('Q#10|11001100|1|10');
```

```
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#10|11001100|1|10' *)
IF fbIED.Relay.LLN0.Mod_.q.sValue <> AcsiQuality_To_String(qValue) THEN
    qValue := fbIED.Relay.LLN0.Mod_.q.qValue;
END_IF

AcsiQuality_ToFrom_String := TRUE;
```

6.4.48 String_To_BinaryTime6

Converts the Binary-Time time information formatted as a string into the components of the structured type T_BinaryTime [▶ 556]. The string to be converted must be specified and has the following structure: 'BT#[YYYY-MM-DD-]hh:mm:ss[.n[n[n]]]'. It corresponds to the TwinCAT Binary-Time string format specification [▶ 772].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION String_To_BinaryTime6 : T_BinaryTime
VAR_INPUT
    in     : STRING(26);
END_VAR
VAR_OUTPUT
    bError : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	STRING(26)	The time information to be converted as a Binary-Time string.

Outputs

Name	Type	Description
bError	BOOL	This output is "TRUE" if the input string could not be converted without error. In this case the function returns the value "STRUCT(timeOfDay := TOD#00:00:00, day := 0)".

Return value

Name	Type	Description
String_To_BinaryTime6	T_BinaryTime [▶ 556]	Converted time information as Binary-Time structure.

Sample

```
METHOD BinaryTime6_ToFrom_String : BOOL
VAR_INPUT
    fbIED  : REFERENCE TO FB_IED;
END_VAR
VAR
    tValue : T_BinaryTime;
    sValue : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

tValue := String_To_BinaryTime6('BT#01:02:03');
sValue := BinaryTime6_To_String(tValue); (* sValue := 'BT#1984-01-01-01:02:03' *)
IF fbIED.Relay.LLN0.brcb1.TimeOfEntry.sValue <> sValue THEN
    tValue := fbIED.Relay.LLN0.brcb1.TimeOfEntry.tValue;
END_IF

tValue := String_To_BinaryTime6('BT#04:05:06.7');
sValue := BinaryTime6_To_String(tValue);(* sValue := 'BT#1984-01-01-04:05:06.700' *)
```

```
IF fbIED.Relay.LLN0.brcb2.TimeOfEntry.sValue <> sValue THEN
    tValue := fbIED.Relay.LLN0.brcb2.TimeOfEntry.tValue;
END_IF

tValue := String_To_BinaryTime6('BT#2017-01-18-11:12:13');
sValue := BinaryTime6_To_String(tValue);(* sValue := 'BT#2017-01-18-11:12:13' *)
IF fbIED.Relay.LLN0.brcb3.TimeOfEntry.sValue <> sValue THEN
    tValue := fbIED.Relay.LLN0.brcb3.TimeOfEntry.tValue;
END_IF

BinaryTime6_ToFrom_String := TRUE;
```

6.4.49 String_To_UtcTime

Converts the UTC-Time information formatted as a string into the components of the structured type T_UtcTime [▶ 560]. The string to be converted is specified and has the following structure: 'UT#YYYY-MM-DD-hh:mm:ss.nnnnnnnnn|LFC|A[A]'. It corresponds to the TwinCAT UTC-Time string format specification [▶ 777].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION String_To_UtcTime : T_UtcTime
VAR_INPUT
    in     : STRING(39) := 'UT#1970-01-01-00:00:00.000000000|000|0';
END_VAR
VAR_OUTPUT
    bError : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	STRING(39)	The time information to be converted as a UTC-Time string.

Outputs

Name	Type	Description
bError	BOOL	This output is "TRUE" if the input string could not be converted without error.

Return value

Name	Type	Description
String_To_UtcTime	T_UtcTime [▶ 560]	Converted time information as UTC-Time structure.

Sample

'UT#2017-01-16-11:22:33.750000000|001|24' means date "2017-01-16", time "11:22:33" and "750" ms, leap seconds unknown, timer has no error, time is not synchronized.

```
METHOD UtcTime_ToFrom_String : BOOL
VAR_INPUT
    fbIED : REFERENCE TO FB_IED;
END_VAR
VAR
    tValue : T_UtcTime;
    sValue : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

tValue := String_To_UtcTime('UT#1970-01-01-00:00:00.000000000|000|0');
```

```
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#1970-01-01-00:00:00.000000000|000|0' *)
IF fbIED.Relay.LLN0.Beh.t.sValue <> sValue THEN
    tValue := fbIED.Relay.LLN0.Beh.t.tValue;
END_IF

tValue := String_To_UtcTime('UT#2018-03-21-14:56:55.125000000|010|3');
(* ClockFailure := 1, accuracy := 3 bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2018-03-21-14:56:55.125000000|010|3' *)
IF fbIED.Relay.LLN0.Health.t.sValue <> sValue THEN
    tValue := fbIED.Relay.LLN0.Health.t.tValue;
END_IF

tValue := String_To_UtcTime('UT#2017-01-16-11:22:33.750000000|001|24');
(* ClockNotSynchronized := 1, accuracy := 24 bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2017-01-16-11:22:33.750000000|001|24' *)
IF fbIED.Relay.LLN0.Mod_.t.sValue <> sValue THEN
    tValue := fbIED.Relay.LLN0.Mod_.t.tValue;
END_IF

UtcTime_ToFrom_String := TRUE;
```

6.4.50 String_To_UtcTimeQuality

Converts the UTC-Time-Quality information formatted as a string into the components of the structured type T_UtcTimeQuality [▶ 560]. The string to be converted must be specified and has the following structure: 'UQ#LFC|A[A]'. It corresponds to the TwinCAT UTC-Time-Quality string format specification [▶ 776].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION String_To_UtcTimeQuality : T_UtcTimeQuality
VAR_INPUT
    in     : STRING(9) := 'UQ#000|0';
END_VAR
VAR_OUTPUT
    bError : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	STRING(9)	The information to be converted as a UTC-Time-Quality string.

Outputs

Name	Type	Description
bError	BOOL	This output is "TRUE" if the input string could not be converted without error.

Return value

Name	Type	Description
String_To_UtcTimeQuality	T_UtcTimeQuality [▶ 560]	Converted information as UTC-Time-Quality structure.

Sample

The return value: 'UQ#001|3' means that leap seconds are not known. Timer has no error and is not synchronized, the resolution of the time information of the seconds fraction is 3 bits.

```
METHOD UtcTimeQuality_ToFrom_String : BOOL
VAR_INPUT
    fbIED    : REFERENCE TO FB_IED;
END_VAR
VAR
    Quality  : T_UtcTimeQuality;
```

```
    sQuality : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

Quality := String_To_UtcTimeQuality('UQ#000|0');
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|0' *)
IF fbIED.Relay.LLN0.Beh.t.sQuality <> sQuality THEN
    Quality := fbIED.Relay.LLN0.Beh.t.Quality;
END_IF

Quality := String_To_UtcTimeQuality('UQ#001|3');
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#001|3' *)
IF fbIED.Relay.LLN0.Health.t.sQuality <> sQuality THEN
    Quality := fbIED.Relay.LLN0.Health.t.Quality;
END_IF

Quality := String_To_UtcTimeQuality('UQ#000|31');
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|31' *)
IF fbIED.Relay.LLN0.Mod_.t.sQuality <> sQuality THEN
    Quality := fbIED.Relay.LLN0.Mod_.t.Quality;
END_IF

UtcTimeQuality_ToFrom_String := TRUE;
```

6.4.51 String_To_UtcTimeStamp

Converts the UTC timestamp information formatted as a string into the components of the structured type T_UtcTime [▶ 560]. The string to be converted is specified and has the following structure: 'US#YYY-MM-DD-hh:mm:ss.nnnnnnn' where:
YYYY:=year
MM:=month
DD:=day
hh:=hour
mm:=minutes
ss:=seconds
nnnnnnn:=nanoseconds (9 decimal places). The T_UtcTime [▶ 560].Quality component is not set during conversion (all bits have the value 0).

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION String_To_UtcTimeStamp : T_UtcTime
VAR_INPUT
    in               : STRING(32):='US#1970-01-01-00:00:00.000000000';
END_VAR
VAR_OUTPUT
    bError           : BOOL;
    secondSinceEpoch : DT;
    fractionOfSecond : T_UINT24;
END_VAR
```

Inputs

Name	Type	Description
in	STRING(32)	The time information to convert as a formatted UTC timestamp string.

Outputs

Name	Type	Description
bError	BOOL	The output is "TRUE" if the input value could not be converted without error.
secondSinceEpoch	DT	Value of converted T_UtcTime [▶ 560].secondSinceEpoch component.
fractionOfSecond	T_UINT24 [▶ 559]	Value of converted T_UtcTime [▶ 560].fractionofSecond component.

Return value

Name	Type	Description
String_To_UtcTimeStamp	T_UtcTime [▶ 560]	Converted timestamp information as UTC time structure.

Sample

```
PROGRAM MAIN
VAR
    tUtc             : T_UtcTime;
    sUtc             : STRING;
    bError           : BOOL;
    secondSinceEpoch : DT;
    fractionOfSecond : T_UINT24;
END_VAR
```

```
tUtc:=String_To_UtcTimeStamp(in:='US#2022-09-15-16:12:34.750000000' , bError=>bError, secondSinceEpo
ch=>secondSinceEpoch, fractionOfSecond=>fractionOfSecond);
sUtc:=UtcTimeStamp_To_String(in:=tUtc, bError=>bError);
(* => sUtc = 'US#2022-09-15-16:12:34.750000000' *)

tUtc.secondSinceEpoch:=DT#2022-09-15-00:00:00;
tUtc.fractionOfSecond[0]:=0;
tUtc.fractionOfSecond[1]:=0;
tUtc.fractionOfSecond[2]:=0;
sUtc:=UtcTimeStamp_To_String(in:=tUtc, bError=>bError);
(* => sUtc ='US#2022-09-15-00:00:00.000000000' *)
```

6.4.52 SystemTime_To_BinaryTime

Converts the time information of type TIMESTRUCT [▶ 531] (system time format) to the time information of type T_BinaryTime [▶ 556] (binary time format). In case of conversion error the output variable "bOverflow" has the value "TRUE" and the returned binary time value has the date:1984-01-01 and the time: 0h0m0s0ms.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION SystemTime_To_BinaryTime : T_BinaryTime
VAR_INPUT
    in        : TIMESTRUCT;
END_VAR
VAR_OUTPUT
    bOverflow : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	TIMESTRUCT [▶ 531]	The time information to be converted as a System-Time structure.

Outputs

Name	Type	Description
bOverflow	BOOL	Date overflow if "TRUE". In this case the system time to be converted has a date greater than 2106-02-07.

Return value

Name	Type	Description
SystemTime_To_BinaryTime	T_BinaryTime [▶ 556]	Converted time information as Binary-Time structure.

Sample

```
PROGRAM MAIN
VAR
    tSystem   : TIMESTRUCT;
    tBinary   : T_BinaryTime;
    bOverflow : BOOL;
END_VAR
```

```
tSystem.wYear:=2022;
tSystem.wMonth:=9;
tSystem.wDay:=16;
tSystem.wHour:=12;
tSystem.wMinute:=52;
tSystem.wSecond:=4;
tSystem.wMilliseconds:=500;
tBinary:=SystemTime_To_BinaryTime(in:=tSystem, bOverflow=>bOverflow);
(* Returns tBinary.timeOfDay = TOD#12:52:04.500 and tBinary.day = 14138 *)

tBinary.day:=1;
tBinary.timeOfDay:=TOD#01:30:15.500;
tSystem:=BinaryTime_To_SystemTime(in:=tBinary, bOverflow=>bOverflow);(* Returns system-
time date: 1984-01-02 and time: 01h30m15s500ms*)
```

6.4.53 SystemTime_To_UtcTime

Converts the time information from type TIMESTRUCT [▶ 531] (system time format) to type T_UtcTime [▶ 560].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION SystemTime_To_UtcTime : T_UtcTime
VAR_INPUT
    in                    : TIMESTRUCT;
    wMicroseconds         : WORD(0..999);
    wNanoseconds          : WORD(0..999);
    bLeapSecondsKnown     : BOOL;
    bClockFailure         : BOOL;
    bClockNotSynchronized : BOOL;
    eAccuracy             : E_UtcTimeAccuracy;
END_VAR
```

Inputs

Name	Type	Description
in	TIMESTRUCT [▶ 531]	The time information to be converted as a System-Time structure.
wMicroseconds	WORD(0..999)	Fraction of the millisecond in microseconds: „0..999“.
wNanoseconds	WORD(0..999)	Fraction of the microsecond in nanoseconds: „0..999“.
bLeapSecondsKnown	BOOL	Quality of the leap seconds: "FALSE" := unknown; "TRUE" := known;
bClockFailure	BOOL	Quality of the timer: "FALSE" := good; "TRUE" := erroneous;
bClockNotSynchronized	BOOL	Quality of the time synchronization: "FALSE" := synchronized; "TRUE" := not synchronized;
eAccuracy	E_UtcTimeAccuracy [▶ 527]	Accuracy of the fraction of the second as an enumeration type.

Return value

Name	Type	Description
SystemTime_To_UtcTime	T_UtcTime [▶ 560]	Converted time information as UTC-Time structure.

Sample

The time information: 2018-03-20-11:33:05.125000000 is converted with the max. accuracy of 24 bits (max. resolution ~60ns). The components of the system time have the following values:

wYear := 2018, wMonth := 3, wDay := 20, wHour := 11, wMinute := 33, wSecond := 5, wMilliseconds := 125, wDayOfWeek := 2.

```
METHOD UtcTime_ToFrom_SystemTime : BOOL
VAR
    tSystem       : TIMESTRUCT;
    sSystem       : STRING;
    tUtc          : T_UtcTime;
    sUtc          : STRING;
    wMicroseconds : WORD;
    wNanoseconds  : WORD;
END_VAR
```

```
tSystem.wYear := 2000;
tSystem.wMonth := 1;
tSystem.wDay := 1;
tSystem.wHour := 0;
tSystem.wMinute := 0;
tSystem.wSecond := 0;
tSystem.wMilliseconds := 0;
tSystem.wDayOfWeek := 0;
sSystem := SystemTime_To_String(tSystem);(* sSystem := '2000-01-01-00:00:00:000' *)
tUtc := SystemTime_To_UtcTime(tSystem, 0, 0, FALSE, FALSE, TRUE, E_UtcTimeAccuracy.Unspecified);
sUtc := UtcTime_To_String(tUtc);(* sUtc := 'UT#2000-01-01-00:00:00:000000000|001|31' *)

IF tUtc.quality.ClockNotSynchronized THEN
    tUtc.secondSinceEpoch := DT#2018-03-20-11:33:05;
    tUtc.fractionOfSecond := LTIME_TO_UtcTimeFractionOfSecond( LTIME#125MS0US0NS );
    tUtc.quality.LeapSecondsKnown := FALSE;
    tUtc.quality.ClockFailure := FALSE;
    tUtc.quality.ClockNotSynchronized := FALSE;
    Accuracy_To_UtcTimeQualityAccuracy( E_UtcTimeAccuracy._03,
                                        bAccuracy0=>tUtc.quality.Accuracy0,
                                        bAccuracy1=>tUtc.quality.Accuracy1,
                                        bAccuracy2=>tUtc.quality.Accuracy2,
```

```
                                        bAccuracy3=>tUtc.quality.Accuracy3,
                                        bAccuracy4=>tUtc.quality.Accuracy4 );

    sUtc := UtcTime_To_String(tUtc);(* sUtc := 'UT#2018-03-20-11:33:05.125000000|000|3' *)
    tSystem := UtcTime_To_SystemTime(tUtc, wMicroseconds=>wMicroseconds, wNanoseconds=>wNanoseconds)
;
    sSystem := SystemTime_To_String(tSystem);(* sSystem := '2018-03-20-11:33:05.125' *)
END_IF


UtcTime_ToFrom_SystemTime := TRUE;
```

6.4.54 UINT24

Initializes an unsigned 24-bit number of type: T_UINT24.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION UINT24 : T_UINT24
VAR_INPUT
    Hi : BYTE;
    Mi : BYTE;
    Lo : BYTE;
END_VAR
```

Inputs

Name	Type	Description
Hi	BYTE	Top 8 bits to be set.
Mi	BYTE	Middle 8 bits to be set.
Lo	BYTE	Lowest 8 bits to be set.

Return value

Name	Type	Description
UINT24	T_UINT24 [▶ 559]	24-bit number.

Example

```
METHOD FINAL Sample_UINT24 : BOOL
VAR
    nValue: T_UINT24;
END_VAR
```

```
nValue:=UINT24(16#12, 16#34, 16#56);(* nValue:=16#123456 *)
Sample_UINT24:=TRUE;
```

6.4.55 UINT128

Initializes an unsigned 128-bit number of type: T_UINT128.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION UINT128 : T_UINT128
VAR_INPUT
    Hi : LWORD;
    Lo : LWORD;
END_VAR
```

Inputs

Name	Type	Description
Hi	LWORD	Top 64 bits to be set.
Lo	LWORD	Lowest 64 bits to be set.

Return value

Name	Type	Description
UINT128	T_UINT128 [▶ 559]	128-bit number.

Example

```
METHOD FINAL Sample_UINT128 : BOOL
VAR
    nValue: T_UINT128;
END_VAR
```

```
nValue:=UINT128(16#1122334455667788, 16#0000000000000001);
(* nValue:=16#1122334455667788_0000000000000001 *)
Sample_UINT128:=TRUE;
```

6.4.56 UtcTime

Initializes the data elements of the structure: T_UtcTime.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION UtcTime : T_UtcTime
VAR_INPUT
    secondSinceEpoch : DT;
    fractionOfSecond : T_UINT24;
    quality          : T_UtcTimeQuality;
END_VAR
```

Inputs

Name	Type	Description
secondSinceEpoch	DT	Number of whole seconds since January 1, 1970.
fractionOfSecond	T_UINT24 [▶ 559]	Seconds fraction encoded as 24-bit number.
quality	T_UtcTimeQuality [▶ 560]	Additional information about the quality and accuracy of the time information.

Return value

Name	Type	Description
UtcTime	T_UtcTime [▶ 560]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_UtcTime : BOOL
VAR
    t: T_UtcTime;
END_VAR
```

```
t:=UtcTime(secondSinceEpoch:=DT#2023-01-01-00:00:00.000, fractionOfSecond:=UINT24(0, 0, 0), quality:
=UtcTimeQuality(FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE));
Sample_UtcTime:=TRUE;
```

6.4.57 UtcTime_To_LWORD

Converts the components of the structured type T_UtcTime [▶ 560] into the binary data of the type LWORD. The coding of the components in LWORD corresponds to the TwinCAT UTC-Time LWORD format specification [▶ 773].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION UtcTime_To_LWORD : LWORD
VAR_INPUT
    in : T_UtcTime;
END_VAR
```

Inputs

Name	Type	Description
in	T_UtcTime [▶ 560]	The time information to be converted as a UTC-Time structure.

Return value

Name	Type	Description
UtcTime_To_LWORD	LWORD	Converted time information as UTC-Time binary data.

Sample

```
METHOD UtcTime_ToFrom_LWORD : BOOL
VAR
    tValue : T_UtcTime;
    sValue : STRING;
END_VAR
```

```
tValue := LWORD_To_UtcTime(0);
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#1970-01-01-00:00:00.000000000|000|0' *)

IF UtcTime_To_LWORD(tValue) = 0 THEN
    tValue.secondSinceEpoch := DT#2000-01-01-00:00:0;
    tValue.fractionOfSecond := LTIME_TO_UtcTimeFractionOfSecond( LTIME#0MS );
    tValue.quality.LeapSecondsKnown := FALSE;
    tValue.quality.ClockFailure := FALSE;
    tValue.quality.ClockNotSynchronized := TRUE;
    Accuracy_To_UtcTimeQualityAccuracy( E_UtcTimeAccuracy._03,
                                        bAccuracy0=>tValue.quality.Accuracy0,
                                        bAccuracy1=>tValue.quality.Accuracy1,
                                        bAccuracy2=>tValue.quality.Accuracy2,
                                        bAccuracy3=>tValue.quality.Accuracy3,
                                        bAccuracy4=>tValue.quality.Accuracy4 );
    sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2000-01-01-00:00:00.000000000|001|3' *)
END_IF


UtcTime_ToFrom_LWORD := TRUE;
```

6.4.58 UtcTime_To_String

Converts the components of the structured type T_UtcTime [▶ 560] into a formatted UTC-Time string. The resulting string has the following structure: 'UT#YYYY-MM-DD-hh:mm:ss.nnnnnnnnn|LFC|A[A]'. It corresponds to the TwinCAT UTC-Time string format specification [▶ 777].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION UtcTime_To_String : STRING(39)
VAR_INPUT
    in     : T_UtcTime;
END_VAR
```

```
VAR_OUTPUT
    bError : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	T_UtcTime [▶ 560]	The time information to be converted as a UTC-Time structure.

Outputs

Name	Type	Description
bError	BOOL	The output is "TRUE" if the input value could not be converted without error.

Return value

Name	Type	Description
UtcTime_To_String	STRING(39)	Converted time information as a UTC-Time string.

Sample

'UT#2017-01-16-11:22:33.750000000|001|24' means date "2017-01-16", time "11:22:33" and "750" ms, leap seconds unknown, timer has no error, time is not synchronized.

```
METHOD UtcTime_ToFrom_String : BOOL
VAR_INPUT
    fbIED : REFERENCE TO FB_IED;
END_VAR
VAR
    tValue : T_UtcTime;
    sValue : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

tValue := String_To_UtcTime('UT#1970-01-01-00:00:00.000000000|000|0');
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#1970-01-01-00:00:00.000000000|000|0' *)
IF fbIED.Relay.LLN0.Beh.t.sValue <> sValue THEN
    tValue := fbIED.Relay.LLN0.Beh.t.tValue;
END_IF

tValue := String_To_UtcTime('UT#2018-03-21-14:56:55.125000000|010|3');
(* ClockFailure := 1, accuracy := 3 bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2018-03-21-14:56:55.125000000|010|3' *)
IF fbIED.Relay.LLN0.Health.t.sValue <> sValue THEN
    tValue := fbIED.Relay.LLN0.Health.t.tValue;
END_IF

tValue := String_To_UtcTime('UT#2017-01-16-11:22:33.750000000|001|24');
(* ClockNotSynchronized := 1, accuracy := 24 bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2017-01-16-11:22:33.750000000|001|24' *)
IF fbIED.Relay.LLN0.Mod_.t.sValue <> sValue THEN
    tValue := fbIED.Relay.LLN0.Mod_.t.tValue;
END_IF

UtcTime_ToFrom_String := TRUE;
```

6.4.59 UtcTime_To_SystemTime

Converts the time information of type T_UtcTime [▶ 560] to the time information of type TIMESTRUCT [▶ 531] (system time format).

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION UtcTime_To_SystemTime : TIMESTRUCT
VAR_INPUT
    in                    : T_UtcTime;
END_VAR
VAR_OUTPUT
    wMicroseconds         : WORD(0..999);
    wNanoseconds          : WORD(0..999);
    bLeapSecondsKnown     : BOOL;
    bClockFailure         : BOOL;
    bClockNotSynchronized : BOOL;
    eAccuracy             : E_UtcTimeAccuracy;
END_VAR
```

Inputs

Name	Type	Description
in	T_UtcTime [▶ 560]	The time information to be converted as a UTC-Time structure.

Outputs

Name	Type	Description
wMicroseconds	WORD(0..999)	Optional output parameter. Fraction of the millisecond in microseconds.
wNanoseconds	WORD(0..999)	Optional output parameter. Fraction of the microsecond in nanoseconds.
bLeapSecondsKnown	BOOL	Optional output parameter. Quality of the leap seconds.
bClockFailure	BOOL	Optional output parameter. Quality of the timer.
bClockNotSynchronized	BOOL	Optional output parameter. Quality of the time synchronization.
eAccuracy	E_UtcTimeAccuracy [▶ 527]	Accuracy of the seconds fraction. This value determines the max. number of significant bits of the T_UtcTime [▶ 560].fractionOfSecond component.

Return value

Name	Type	Description
UtcTime_To_SystemTime	TIMESTRUCT [▶ 531]	Converted time information as a System-Time structure.

Sample

The time information: 2018-03-20-11:33:05.125000000 is converted with the max. accuracy of 24 bits (max. resolution ~60ns). The components of the system time have the following values:

wYear := 2018, wMonth := 3, wDay := 20, wHour := 11, wMinute := 33, wSecond := 5, wMilliseconds := 125, wDayOfWeek := 2.

```
METHOD UtcTime_ToFrom_SystemTime : BOOL
VAR
    tSystem       : TIMESTRUCT;
    sSystem       : STRING;
    tUtc          : T_UtcTime;
    sUtc          : STRING;
    wMicroseconds : WORD;
    wNanoseconds  : WORD;
END_VAR
```

```
tSystem.wYear := 2000;
tSystem.wMonth := 1;
tSystem.wDay := 1;
tSystem.wHour := 0;
tSystem.wMinute := 0;
tSystem.wSecond := 0;
```

```
tSystem.wMilliseconds := 0;
tSystem.wDayOfWeek := 0;
sSystem := SystemTime_To_String(tSystem);(* sSystem := '2000-01-01-00:00:00:000' *)
tUtc := SystemTime_To_UtcTime(tSystem, 0, 0, FALSE, FALSE, TRUE, E_UtcTimeAccuracy.Unspecified);
sUtc := UtcTime_To_String(tUtc);(* sUtc := 'UT#2000-01-01-00:00:00:000000000|001|31' *)

IF tUtc.quality.ClockNotSynchronized THEN
    tUtc.secondSinceEpoch := DT#2018-03-20-11:33:05;
    tUtc.fractionOfSecond := LTIME_TO_UtcTimeFractionOfSecond( LTIME#125MS0US0NS );
    tUtc.quality.LeapSecondsKnown := FALSE;
    tUtc.quality.ClockFailure := FALSE;
    tUtc.quality.ClockNotSynchronized := FALSE;
    Accuracy_To_UtcTimeQualityAccuracy( E_UtcTimeAccuracy._03,
                                        bAccuracy0=>tUtc.quality.Accuracy0,
                                        bAccuracy1=>tUtc.quality.Accuracy1,
                                        bAccuracy2=>tUtc.quality.Accuracy2,
                                        bAccuracy3=>tUtc.quality.Accuracy3,
                                        bAccuracy4=>tUtc.quality.Accuracy4 );

    sUtc := UtcTime_To_String(tUtc);(* sUtc := 'UT#2018-03-20-11:33:05.125000000|000|3' *)
    tSystem := UtcTime_To_SystemTime(tUtc, wMicroseconds=>wMicroseconds, wNanoseconds=>wNanoseconds)
;
    sSystem := SystemTime_To_String(tSystem);(* sSystem := '2018-03-20-11:33:05.125' *)
END_IF


UtcTime_ToFrom_SystemTime := TRUE;
```

6.4.60 UtcTimeFractionOfSecond_To_LTIME

Converts the fraction of a second of the type T_UINT24 [▶ 559] into the type LTIME. The binary data to be converted correspond to the type of the T_UtcTime [▶ 560].fractionOfSecond component. The 24 bits must be coded in a special way. See: Coding of the UTC-Time.fractionOfSecond0..23-bits [▶ 773]. The T_Utctime [▶ 560].Quality.Accuracy0..4-bit components determine the max. number of significant fractionOfSecond bits and the max. resolution than can be attained when converting (~60 ns).

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION UtcTimeFractionOfSecond_To_LTIME : LTIME
VAR_INPUT
    in      : T_UINT24;
END_VAR
VAR_OUTPUT
    nanosec : LWORD;
    attosec : LWORD;
END_VAR
```

Inputs

Name	Type	Description
in	T_UINT24 [▶ 559]	The fraction of the second to be converted as binary data.

Outputs

Name	Type	Description
nanosec	LWORD	Optional output parameter. Fraction of the second in nanoseconds.
attosec	LWORD	Optional output parameter. Fraction of the second in attoseconds.

Return value

Name	Type	Description
UtcTimeFractionOfSecond_To_LTIME	LTIME	Converted fraction of a second in nanoseconds.

Sample

```
METHOD UtcTimeFractionOfSecond_ToFrom_LTIME : BOOL
VAR
    tValue    : T_UtcTime;
    sValue    : STRING;
    tFos      : LTIME;
    nanosec   : LWORD;
END_VAR
```

```
tValue.quality := String_To_UtcTimeQuality('UQ#000|24');
tValue.secondSinceEpoch := DT#2018-03-28-12:13:14;

tValue.fractionOfSecond := LTIME_To_UtcTimeFractionOfSecond(LTIME#0NS);
(* tValue.fractionOfSecond := [16#00, 16#00, 16#00], 0 significant bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2018-03-28-12:13:14.000000000|000|24' *)
tFos := UtcTimeFractionOfSecond_To_LTIME(tValue.fractionOfSecond, nanosec=>nanosec);
(* tFos := LTIME#0ns, nanosec := 0 *)

tValue.fractionOfSecond := LTIME_To_UtcTimeFractionOfSecond(LTIME#750MS);
(* tValue.fractionOfSecond := [16#03, 16#00, 16#00], 3 significant bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2018-03-28-12:13:14.750000000|000|24' *)
tFos := UtcTimeFractionOfSecond_To_LTIME(tValue.fractionOfSecond, nanosec=>nanosec);
(* tFos := LTIME#750ms, nanosec := 750000000 *)

tValue.fractionOfSecond := LTIME_To_UtcTimeFractionOfSecond(LTIME#968MS750US);
(* tValue.fractionOfSecond := [16#1F, 16#00, 16#00], 5 significant bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2018-03-28-12:13:14.968750000|000|24' *)
tFos := UtcTimeFractionOfSecond_To_LTIME(tValue.fractionOfSecond, nanosec=>nanosec);
(* tFos := LTIME#968ms750us, nanosec := 968750000 *)

tValue.fractionOfSecond := LTIME_To_UtcTimeFractionOfSecond(LTIME#998MS46US875NS);
(* tValue.fractionOfSecond := [16#FF, 16#01, 16#00], 9 significant bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2018-03-28-12:13:14.998046875|000|24' *)
tFos := UtcTimeFractionOfSecond_To_LTIME(tValue.fractionOfSecond, nanosec=>nanosec);
(* tFos := LTIME#998ms46us875ns, nanosec := 998046875 *)

tValue.fractionOfSecond := LTIME_To_UtcTimeFractionOfSecond(LTIME#999MS999US999NS);
(* tValue.fractionOfSecond := [16#FF, 16#FF, 16#FF], 24 significant bits *)
sValue := UtcTime_To_String(tValue);(* sValue := 'UT#2018-03-28-12:13:14.999999940|000|24' *)
tFos := UtcTimeFractionOfSecond_To_LTIME(tValue.fractionOfSecond, nanosec=>nanosec);
(* tFos := LTIME#999ms999us940ns, nanosec := 999999940 *)

UtcTimeFractionOfSecond_ToFrom_LTIME := TRUE;
```

6.4.61 UtcTimeIsNull

This function returns "TRUE" if all components of the structured type T_UtcTime [▶ 560] are "0".

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION UtcTimeIsNull : BOOL
VAR_INPUT
    in : T_UtcTime;
END_VAR
```

Inputs

Name	Type	Description
in	T_UtcTime [▶ 560]	UTC-Time value to be checked.

Return value

Name	Type	Description
UtcTimeIsNull	BOOL	TRUE if all components of UTC-Time are "0".

Sample

```
PROGRAM P_UtcTimeIsNull
VAR
    t     : T_UtcTime;
    bNull : BOOL;
END_VAR
```

```
t     := SEL( bNull, Null_TimeStamp, String_To_UtcTime('UT#2018-03-21-15:58:03.125000000|000|3') );
bNull := UtcTimeIsNull(t);
```

6.4.62 UtcTimeQuality

Initializes the data elements of the structure: T_UtcTimeQuality.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION UtcTimeQuality : T_UtcTimeQuality
VAR_INPUT
    LeapSecondsKnown     : BOOL;
    ClockFailure         : BOOL;
    ClockNotSynchronized : BOOL;
    Accuracy0            : BOOL;
    Accuracy1            : BOOL;
    Accuracy2            : BOOL;
    Accuracy3            : BOOL;
    Accuracy4            : BOOL;
END_VAR
```

Inputs

Name	Type	Description
LeapSecondsKnown	BOOL	Data element "LeapSecondsKnown" to be set.
ClockFailure	BOOL	Data element "ClockFailure" to be set.
ClockNotSynchronized	BOOL	Data element "ClockNotSynchronized" to be set.
Accuracy0	BOOL	Data element "Accuracy0" to be set.
Accuracy1	BOOL	Data element "Accuracy1" to be set.
Accuracy2	BOOL	Data element "Accuracy2" to be set.
Accuracy3	BOOL	Data element "Accuracy3" to be set.
Accuracy4	BOOL	Data element "Accuracy4" to be set.

Return value

Name	Type	Description
UtcTimeQuality	T_UtcTimeQuality [▶ 560]	Structure with initialized data elements.

Example

```
METHOD FINAL Sample_UtcTimeQuality : BOOL
VAR
    q: T_UtcTimeQuality;
END_VAR
```

```
q:=UtcTimeQuality(FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE);
(* q.ClockNotSynchronized:=TRUE, all other elements: FALSE *)
Sample_UtcTimeQuality:=TRUE;
```

6.4.63 UtcTimeQuality_To_BYTE

Converts the components of the structured type T_UtcTimeQuality [▶ 560] into the binary data of the type BYTE. The coding of the components in BYTE corresponds to the TwinCAT UTC-Time-Quality-BYTE format specification [▶ 775].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION UtcTimeQuality_To_BYTE : BYTE
VAR_INPUT
    in : T_UtcTimeQuality;
END_VAR
```

Inputs

Name	Type	Description
in	T_UtcTimeQuality [▶ 560]	The information to be converted as UTC-Time-Quality structure.

Return value

Name	Type	Description
UtcTimeQuality_To_BYTE	BYTE	Converted information as UTC-Time-Quality binary data.

Sample

```
METHOD UtcTimeQuality_ToFrom_Byte : BOOL
VAR_INPUT
    fbIED    : REFERENCE TO FB_IED;
END_VAR
VAR
    Quality  : T_UtcTimeQuality;
    sQuality : STRING;
    u8       : BYTE;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

Quality := Byte_To_UtcTimeQuality(0);
u8 := UtcTimeQuality_To_Byte(Quality);(* u8 := 2#00000000 => 0 bits accuracy *)
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|00' *)
IF UtcTimeQuality_To_Byte(fbIED.Relay.LLN0.Beh.t.Quality) <> u8 THEN
    Quality := fbIED.Relay.LLN0.Beh.t.Quality;
END_IF

Quality := String_To_UtcTimeQuality('UQ#000|03');
u8 := UtcTimeQuality_To_Byte(Quality);(* u8 := 2#11000000 => 3 bits accuracy *)
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|03' *)
IF UtcTimeQuality_To_Byte(fbIED.Relay.LLN0.Health.t.Quality) <> u8 THEN
    Quality := fbIED.Relay.LLN0.Health.t.Quality;
END_IF

Quality := String_To_UtcTimeQuality('UQ#001|05');
u8 := UtcTimeQuality_To_Byte(Quality);
(* u8 := 2#10100100 => 5 bits accuracy, ClockNotSynchronized := 1 *)
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#001|05' *)
IF UtcTimeQuality_To_Byte(fbIED.Relay.LLN0.Mod_.t.Quality) <> u8 THEN
    Quality := fbIED.Relay.LLN0.Mod_.t.Quality;
END_IF

Quality := String_To_UtcTimeQuality('UQ#000|31');
u8 := UtcTimeQuality_To_Byte(Quality);(* u8 := 2#11111000 => unspecified accuracy *)
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|31' *)
IF UtcTimeQuality_To_Byte(fbIED.Relay.MMXU1.phV.phsA.t.Quality) <> u8 THEN
    Quality := fbIED.Relay.MMXU1.phV.phsA.t.Quality;
END_IF

UtcTimeQuality_ToFrom_Byte := TRUE;
```

6.4.64 UtcTimeQuality_To_String

Converts the components of the structured type T_UtcTimeQuality [▶ 560] into a formatted UTC-Time-Quality string. The resulting string has the following structure: 'UQ#LFC|A[A]'. It corresponds to the TwinCAT UTC-Time-Quality string format specification [▶ 776].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION UtcTimeQuality_To_String : STRING(9)
VAR_INPUT
    in     : T_UtcTimeQuality;
END_VAR
VAR_OUTPUT
    bError : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	T_UtcTimeQuality [▶ 560]	The information to be converted as UTC-Time-Quality structure.

Outputs

Name	Type	Description
bError	BOOL	The output is "TRUE" if the input value could not be converted without error.

Return value

Name	Type	Description
UtcTimeQuality_To_String	STRING(9)	Converted information as a UTC-Time-Quality string.

Sample

The return value: 'UQ#001|3' means that leap seconds are not known. Timer has no error and is not synchronized, the resolution of the time information of the seconds fraction is 3 bits.

```
METHOD UtcTimeQuality_ToFrom_String : BOOL
VAR_INPUT
    fbIED    : REFERENCE TO FB_IED;
END_VAR
VAR
    Quality  : T_UtcTimeQuality;
    sQuality : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

Quality := String_To_UtcTimeQuality('UQ#000|0');
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|0' *)
IF fbIED.Relay.LLN0.Beh.t.sQuality <> sQuality THEN
    Quality := fbIED.Relay.LLN0.Beh.t.Quality;
END_IF

Quality := String_To_UtcTimeQuality('UQ#001|3');
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#001|3' *)
IF fbIED.Relay.LLN0.Health.t.sQuality <> sQuality THEN
    Quality := fbIED.Relay.LLN0.Health.t.Quality;
END_IF

Quality := String_To_UtcTimeQuality('UQ#000|31');
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|31' *)
IF fbIED.Relay.LLN0.Mod_.t.sQuality <> sQuality THEN
    Quality := fbIED.Relay.LLN0.Mod_.t.Quality;
END_IF

UtcTimeQuality_ToFrom_String := TRUE;
```

6.4.65 UtcTimeQualityAccuracy_To_Accuracy

Converts the accuracy of the time information of the structured type T_UtcTimeQuality [▶ 560] to the enumeration type E_UtcTimeAccuracy [▶ 527]. Only the Accuracy0..4-bit components are converted. These variables determined the max. number of significant bits in the T_UtcTime [▶ 560].fractionOfSecond component.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION UtcTimeQualityAccuracy_To_Accuracy : E_UtcTimeAccuracy
VAR_INPUT
    in : T_UtcTimeQuality;
END_VAR
```

Inputs

Name	Type	Description
in	T_UtcTimeQuality [▶ 560]	The information to be converted as UTC-Time-Quality.Accuracy0..4-bit components. See: Coding of the Accuracy0..4-bits [▶ 774].

Return value

Name	Type	Description
UtcTimeQualityAccuracy_To_Accuracy	E_UtcTimeAccuracy [▶ 527]	Converted information as Accuracy enumerator type.

Sample

```
METHOD UtcTimeQualityAccuracy_ToFrom_Accuracy : BOOL
VAR_INPUT
    fbIED    : REFERENCE TO FB_IED;
END_VAR
VAR
    Quality  : T_UtcTimeQuality;
    sQuality : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

Quality.LeapSecondsKnown := FALSE;
Quality.ClockFailure := FALSE;
Quality.ClockNotSynchronized := FALSE;
Accuracy_To_UtcTimeQualityAccuracy( E_UtcTimeAccuracy.Null,(* Null bits accuracy *)
                                    bAccuracy0=>Quality.Accuracy0,
                                    bAccuracy1=>Quality.Accuracy1,
                                    bAccuracy2=>Quality.Accuracy2,
                                    bAccuracy3=>Quality.Accuracy3,
                                    bAccuracy4=>Quality.Accuracy4 );
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|0' *)
IF fbIED.Relay.LLN0.Beh.t.eAccuracy <> UtcTimeQualityAccuracy_To_Accuracy(Quality) THEN
    Quality := fbIED.Relay.LLN0.Beh.t.Quality;
END_IF

Quality.LeapSecondsKnown := FALSE;
Quality.ClockFailure := FALSE;
Quality.ClockNotSynchronized := FALSE;
Accuracy_To_UtcTimeQualityAccuracy( E_UtcTimeAccuracy._03,(* 3 bits accuracy *)
                                    bAccuracy0=>Quality.Accuracy0,
                                    bAccuracy1=>Quality.Accuracy1,
                                    bAccuracy2=>Quality.Accuracy2,
                                    bAccuracy3=>Quality.Accuracy3,
                                    bAccuracy4=>Quality.Accuracy4 );
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|3' *)
IF fbIED.Relay.LLN0.Health.t.eAccuracy <> UtcTimeQualityAccuracy_To_Accuracy(Quality) THEN
    Quality := fbIED.Relay.LLN0.Health.t.Quality;
END_IF
```

```
Quality.LeapSecondsKnown := FALSE;
Quality.ClockFailure := FALSE;
Quality.ClockNotSynchronized := FALSE;
Accuracy_To_UtcTimeQualityAccuracy( E_UtcTimeAccuracy._05,(* 5 bits accuracy *)
                                    bAccuracy0=>Quality.Accuracy0,
                                    bAccuracy1=>Quality.Accuracy1,
                                    bAccuracy2=>Quality.Accuracy2,
                                    bAccuracy3=>Quality.Accuracy3,
                                    bAccuracy4=>Quality.Accuracy4 );
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|5' *)
IF fbIED.Relay.LLN0.Mod_.t.eAccuracy <> UtcTimeQualityAccuracy_To_Accuracy(Quality) THEN
    Quality := fbIED.Relay.LLN0.Mod_.t.Quality;
END_IF

Quality.LeapSecondsKnown := FALSE;
Quality.ClockFailure := FALSE;
Quality.ClockNotSynchronized := FALSE;
Accuracy_To_UtcTimeQualityAccuracy( E_UtcTimeAccuracy.Unspecified,(* Unspecified accuracy *)
                                    bAccuracy0=>Quality.Accuracy0,
                                    bAccuracy1=>Quality.Accuracy1,
                                    bAccuracy2=>Quality.Accuracy2,
                                    bAccuracy3=>Quality.Accuracy3,
                                    bAccuracy4=>Quality.Accuracy4 );
sQuality := UtcTimeQuality_To_String(Quality);(* sQuality := 'UQ#000|31' *)
IF fbIED.Relay.MMXU1.phV.phsA.t.eAccuracy <> UtcTimeQualityAccuracy_To_Accuracy(Quality) THEN
    Quality := fbIED.Relay.MMXU1.phV.phsA.t.Quality;
END_IF

UtcTimeQualityAccuracy_ToFrom_Accuracy := TRUE;
```

6.4.66 UtcTimeStamp_To_String

Converts the components of the structured type T_UtcTime [▶ 560] into a formatted UTC timestamp string.
The resulting string has the following structure: 'US#YYY-MM-DD-hh:mm:ss.nnnnnnn':
YYYY:=year
MM:=month
DD:=day
hh:=hour
mm:=minutes
ss:=seconds
nnnnnnn:=nanoseconds (9 decimal places).
The T_UtcTime [▶ 560].Quality component is not converted to the string.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION UtcTimeStamp_To_String : STRING(32)
VAR_INPUT
    in     : T_UtcTime;
END_VAR
VAR_OUTPUT
    bError : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	T_UtcTime [▶ 560]	The time information to be converted as a UTC-Time structure.

Outputs

Name	Type	Description
bError	BOOL	The output is "TRUE" if the input value could not be converted without error.

Return value

Name	Type	Description
UtcTimeStamp_To_String	STRING(32)	Converted time information as a formatted UTC timestamp string.

Sample

```
PROGRAM MAIN
VAR
    tUtc             : T_UtcTime;
    sUtc             : STRING;
    bError           : BOOL;
    secondSinceEpoch : DT;
    fractionOfSecond : T_UINT24;
END_VAR
```

```
tUtc:=String_To_UtcTimeStamp(in:='US#2022-09-15-16:12:34.750000000' , bError=>bError, secondSinceEpo
ch=>secondSinceEpoch, fractionOfSecond=>fractionOfSecond);
sUtc:=UtcTimeStamp_To_String(in:=tUtc, bError=>bError);
(* => sUtc = 'US#2022-09-15-16:12:34.750000000' *)

tUtc.secondSinceEpoch:=DT#2022-09-15-00:00:00;
tUtc.fractionOfSecond[0]:=0;
tUtc.fractionOfSecond[1]:=0;
tUtc.fractionOfSecond[2]:=0;
sUtc:=UtcTimeStamp_To_String(in:=tUtc, bError=>bError);
(* => sUtc ='US#2022-09-15-00:00:00.000000000' *)
```

6.4.67 Validity_To_AcsiQualityValidity

Converts the value of the enumeration type E_AcsiQualityValidity [▶ 501] into two Boolean values. These two output values correspond to the Validity0 and Validity1 bit components of the structured type ST_AcsiQuality [▶ 541]. See also: Coding of the Validity0..1-bits [▶ 778].

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION Validity_To_AcsiQualityValidity
VAR_INPUT
    in         : E_AcsiQualityValidity;
END_VAR
VAR_OUTPUT
    bValidity0 : BOOL;
    bValidity1 : BOOL;
END_VAR
```

Inputs

Name	Type	Description
in	E_AcsiQualityValidity [▶ 501]	The information to be converted as an enumeration type.

Outputs

Converted information as ST_AcsiQuality [▶ 541].Validity0..1-bit components.

Name	Type	Description
bValidity0	BOOL	Validity MSB bit.
bValidity1	BOOL	Validity LSB bit.

Return value

None.

Sample

```
METHOD AcsiQualityValidity_ToFrom_Validity : BOOL
VAR_INPUT
    fbIED  : REFERENCE TO FB_IED;
END_VAR
VAR
    qValue : ST_AcsiQuality;
    sValue : STRING;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF
Validity_To_AcsiQualityValidity(E_AcsiQualityValidity.Good, bValidity0=>qValue.Validity0, bValidity1
=>qValue.Validity1);
qValue.Overflow := FALSE;
qValue.OutOfRange := FALSE;
qValue.BadReference := FALSE;
qValue.Oscillatory := FALSE;
qValue.Failure := FALSE;
qValue.OldData := FALSE;
qValue.Inconsistent := FALSE;
qValue.Inaccurate := FALSE;
qValue.Source := Source_To_AcsiQualitySource(E_AcsiQualitySource.Process);
qValue.Test := FALSE;
qValue.OperatorBlocked := FALSE;
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#00|00000000|0|00' *)

IF fbIED.Relay.LLN0.Beh.q.eValidity <> AcsiQualityValidity_To_Validity(qValue) THEN
    qValue := fbIED.Relay.LLN0.Beh.q.qValue;
END_IF

AcsiQualityValidity_ToFrom_Validity := TRUE;
```

6.4.68 WORD_To_AcsiQuality

Converts the binary data of the type WORD into the components of the structured type ST_AcsiQuality [▶ 541]. The coding of the components in the WORD corresponds to the TwinCAT Quality-WORD format specification [▶ 779].

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
FUNCTION Word_To_AcsiQuality : ST_AcsiQuality
VAR_INPUT
    in : WORD(0..8191);
END_VAR
```

Inputs

Name	Type	Description
in	WORD(0..8191)	The information to be converted as Quality binary data.

Return value

Name	Type	Description
WORD_To_AcsiQuality	ST_AcsiQuality [▶ 541]	Converted information as a Quality structure.

Sample

```
METHOD AcsiQuality_ToFrom_WORD : BOOL
VAR_INPUT
    fbIED : REFERENCE TO FB_IED;
END_VAR
VAR
    qValue : ST_AcsiQuality;
    sValue : STRING;
    nValue : WORD;
END_VAR
```

```
IF NOT __ISVALIDREF(fbIED) THEN
    RETURN;
END_IF

qValue := WORD_To_AcsiQuality(0);
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#00|00000000|0|00' *)
nValue := AcsiQuality_To_WORD(qValue);(* nValue := 2#00000000_00000000 *)
IF fbIED.Relay.LLN0.Beh.q.nValue <> nValue THEN
    qValue := fbIED.Relay.LLN0.Beh.q.qValue;
END_IF

qValue := String_To_AcsiQuality('Q#01|00110011|0|01');
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#01|00110011|0|01' *)
nValue := AcsiQuality_To_Word(qValue);(* nValue := 2#00010011_00110010 *)
IF fbIED.Relay.LLN0.Health.q.nValue <> nValue THEN
    qValue := fbIED.Relay.LLN0.Health.q.qValue;
END_IF

qValue := String_To_AcsiQuality('Q#10|11001100|1|10');
sValue := AcsiQuality_To_String(qValue);(* sValue := 'Q#10|11001100|1|10' *)
nValue := AcsiQuality_To_Word(qValue);(* nValue := 2#00001100_11001101 *)
IF fbIED.Relay.LLN0.Mod_.q.nValue <> nValue THEN
    qValue := fbIED.Relay.LLN0.Mod_.q.qValue;
END_IF

AcsiQuality_ToFrom_WORD := TRUE;
```

6.4.69 ZeroOctetString

Overwrites all octets of an octet string with the value: 0.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
FUNCTION ZeroOctetString : UDINT
VAR_IN_OUT
    o : ARRAY[*] OF BYTE;
END_VAR
```

Inputs/Outputs

Name	Type	Description
o	ARRAY[*] OF BYTE	Octet string whose octets are to be overwritten with 0.

Return value

Name	Type	Description
ZeroOctetString	UDINT	Number of octets overwritten.

Example

```
METHOD FINAL Sample_ZeroOctetString : BOOL
VAR
    o6  : T_OCTET6:=[6(16#FF)];
    o8  : T_OCTET8:=[8(16#FF)];
    o64 : T_OCTET64:=[64(16#FF)];
    size: UDINT;
END_VAR
```

```
size:=ZeroOctetString(o:=o6);(* size:=6, all octets: 0 *)
size:=ZeroOctetString(o:=o8);(* size:=8, all octets: 0 *)
size:=ZeroOctetString(o:=o64);(* size:=64, all octets: 0 *)
Sample_ZeroOctetString:=TRUE;
```

6.5 Interfaces

6.5.1 Data model

6.5.1.1 Data attributes

6.5.1.1.1 I_AcsiDATypeScaledValueConfig

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AsciCommonAttributeClass [▶ 362] -> I_AcsiDATypeScaledValueConfig

```
INTERFACE I_AcsiDATypeScaledValueConfig EXTENDS I_AcsiCommonAttributeClass
```

Methods

Name	Description
ScaleAToP [▶ 361]	Scales an attribute value into a process value
ScalePToA [▶ 361]	Scales a process value into an attribute value

Properties

Name	Type	Access	Description
fOffset	REAL	Get,Set	Attribute value: Scaling offset
fScaleFactor	REAL	Get,Set	Attribute value: Scaling factor

6.5.1.1.1.1 ScaleAToP

Scales an attribute value of type DINT (attribute i of AnalogValue) to a process value of type LREAL. The conversion formula is:

$$ScaleAToP := (iValue * scaleFactor) + offset$$

```
METHOD ScaleAToP : LREAL
VAR_INPUT
    iValue : DINT;
END_VAR
```

Return parameter: Scaled process value

6.5.1.1.1.2 ScalePToA

Scales a process value of type LREAL to an attribute value of type DINT (attribute i of AnalogValue). The conversion formula is:

$$ScalePToA := \frac{fProcess - offset}{scaleFactor}$$

```
METHOD ScalePToA : DINT
VAR_INPUT
    fProcess : LREAL;
END_VAR
```

Return parameter: Scaled attribute value

6.5.1.1.2 I_AcsiDATypeUnit

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AsciCommonAttributeClass [▶ 362] -> I_AcsiDATypeUnit

```
INTERFACE I_AcsiDATypeUnit EXTENDS I_AcsiCommonDataClass, I_AcsiCommonAttributeClass
```

Methods

Name	Description
ScaleAToP [▶ 362]	Scales an attribute value into a process value
ScalePToA [▶ 362]	Scales a process value into an attribute value

Properties

Name	Type	Access	Description
eMultiplier	E_AcsiMultiplier [▶ 493]	Get,Set	Attribute value: Multiplier
eSIUnit	E_AcsiSIUnit [▶ 508]	Get,Set	Attribute value: SI unit

6.5.1.1.2.1 ScaleAToP

Scales an attribute value of type REAL (attribute f of AnalogValue) to a process value of type LREAL. The conversion formula is:

$$ScaleAToP := fValue * 10^{units.multiplier}$$

```
METHOD ScaleAToP : LREAL
VAR_INPUT
    fValue : REAL;
END_VAR
```

Return parameter: Scaled process value

6.5.1.1.2.2 ScalePToA

Scales a process value of type LREAL to an attribute value of type REAL (attribute f of AnalogValue). The conversion formula is:

$$ScalePToA := \frac{fProcess}{10^{units.multiplier}}$$

```
METHOD ScalePToA : REAL
VAR_INPUT
    fProcess : LREAL;
END_VAR
```

Return parameter: Scaled attribute value

6.5.1.2 I_AcsiCommonAttributeClass

Objects that implement this interface are at the fifth hierarchical level (or higher) of the IEC 61850 data model and are referred to as data attributes (DA). Data attributes are the smallest object units. They contain detailed information such as data object values. Multiple data attributes are often grouped into larger units (common data classes, CDC), for example binary counter reading, BCR. There are also attributes that contain further attributes (e.g. AnalogValue, RangeConfig or Unit). Such attributes are referred to as

"constructed attributes". This can result in further, deeper hierarchical levels. The TwinCAT Telecontrol Configurator generates function blocks for these levels, which are derived/extended from FB_AcsiCommonAttributeClass [▶ 152]. All function blocks derived from FB_AcsiCommonAttributeClass [▶ 152] automatically have the interface referred to above implemented by default.

For objects that implement this interface, the same services can be used as for the I_AcsiCommonDataClass [▶ 364] interface (e.g. GetDataValues, SetDataValues).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AcsiCommonAttributeClass

```
INTERFACE I_AcsiCommonAttributeClass EXTENDS I_AcsiCommonDataClass
```

Methods

Name	Description
AddAttributeToContainer [▶ 363]	Adds a data attribute object to the container of another data attribute object

6.5.1.2.1 AddAttributeToContainer

This method is used to add a data attribute object (DA) to the container of another data attribute object (DA).

```
METHOD AddAttributeToContainer : BOOL
VAR_INPUT
    ipAttribute : I_AcsiCommonAttributeClass;
END_VAR
```

ipAttribute: Interface pointer of type I_AcsiCommonAttributeClass [▶ 362] of the data attribute object, which is to be added to the container.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.3 I_AcsiCommonBufferedReportControlBlockClass

Objects that implement this interface are report control block instances for buffered reports (buffered report control blocks, BRCBs). The TwinCAT Telecontrol Configurator generates report control block instances, which are derived/extended from FB_AcsiCommonBufferedReportControlBlockClass [▶ 152]. All function blocks derived from FB_AcsiCommonBufferedReportControlBlockClass [▶ 152] automatically have the interface referred to above implemented by default.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AcsiCommonControlBlockClass [▶ 363] -> I_AcsiCommonBufferedReportControlBlockClass

```
INTERFACE I_AcsiCommonBufferedReportControlBlockClass EXTENDS I_AcsiCommonControlBlockClass
```

6.5.1.4 I_AcsiCommonControlBlockClass

All objects that are control block instances implement an interface that inherits from this interface. This interface serves as the base interface for all control blocks.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AcsiCommonControlBlockClass

```
INTERFACE I_AcsiCommonControlBlockClass EXTENDS I_AcsiCommonDataClass
```

6.5.1.5 I_AcsiCommonControllableDataClass

Objects that implement this interface can be used to run services such as Control, Operate, Cancel, Select or SelectWithValue. The TwinCAT Telecontrol Configurator generates function blocks for this purpose, which are derived/extended from FB_AcsiCommonControllableDataClass [▶ 153]. All function blocks derived from FB_AcsiCommonControllableDataClass [▶ 153] automatically have the interface referred to above implemented by default.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -
> I_AcsiCommonControllableDataClass

```
INTERFACE I_AcsiCommonControllableDataClass EXTENDS I_AcsiCommonDataClass
```

6.5.1.6 I_AcsiCommonDataClass

Objects that implement this interface are at the fourth hierarchical level of the IEC 61850 data model and are referred to as data objects (DO). IEC 61850-7-4 defines the scope of these data objects. Some data objects are mandatory, others are optional. Data objects can also serve as containers for other data objects. The TwinCAT Telecontrol Configurator generates function blocks for these levels, which are derived/extended from FB_AcsiCommonDataClass [▶ 154]. All function blocks derived from FB_AcsiCommonDataClass [▶ 154] automatically have the interface referred to above implemented by default. Services such as GetDataValues, SetDataValues can be applied to objects that implement this interface.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass

```
INTERFACE I_AcsiCommonDataClass EXTENDS I_AcsiCommonNodeClass
```

Methods

Name	Description
AddAttributeToContainer [▶ 364]	Adds a data attribute object to the container of another data object
AddDataToContainer [▶ 365]	Adds a data object to the container of another data object
AddToDataSet [▶ 365]	Adds a data object to the member container of the data set
DeleteFromDataSet [▶ 365]	Removes a data object from the member container of the data set

Properties

Name	Type	Access	Description
bPresence [▶ 366]	BOOL	Get,Set	Attribute visibility
Fc [▶ 366]	DWORD	Get,Set	Functional group
TrgOp [▶ 366]	DWORD	Get,Set	Report trigger

6.5.1.6.1 AddAttributeToContainer

This method is used to add a data attribute object (DA) to the container of a data object (DO).

```
METHOD AddAttributeToContainer : BOOL
VAR_INPUT
    ipAttribute : I_AcsiCommonAttributeClass;
END_VAR
```

ipAttribute: Interface pointer of type I_AcsiCommonAttributeClass [▶ 362] of the data attribute object, which is to be added to the container.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.6.2 AddDataToContainer

This method is used to add a data object (DO) to the container of another data object (DO).

```
METHOD FINAL AddDataToContainer : BOOL
VAR_INPUT
    ipData  : I_AcsiCommonDataClass;
END_VAR
```

ipData: Interface pointer of type I_AcsiCommonAttributeClass [▶ 364] of the data object, which is to be added to the container.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.6.3 AddToDataSet

This method is used to add a data object (DO) to the member container of the data set.

```
METHOD FINAL AddToDataSet : BOOL
VAR_INPUT
    ipDataSet : I_AcsiCommonDataSetClass;
    eFc       : E_AcsiFc;
END_VAR
VAR_OUTPUT
    bMatched  : BOOL;
    ipDirNode : I_AcsiDataSetMemberClass;
END_VAR
```

ipDataSet: Interface pointer of type I_AcsiCommonDataSetClass [▶ 366] of the data set in whose member container the data object is to be added.

eFc: Functional group of the data object of type E_AcsiFc [▶ 487].

bMatched: If TRUE, the data object is already in the member container.

ipDirNode: interface pointer of the new container list item.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.6.4 DeleteFromDataSet

This method is used to remove a data object (DO) from the member container of the data set.

```
METHOD FINAL DeleteFromDataSet : BOOL
VAR_INPUT
    ipDataSet : I_AcsiCommonDataSetClass;
    eFc       : E_AcsiFc;
END_VAR
```

ipDataSet: Interface pointer of type I_AcsiCommonDataSetClass [▶ 366] of the data set from whose member container the data object is to be removed.

eFc: Functional group of the data object of type E_AcsiFc [▶ 487].

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.6.5 bPresence

This property determines whether the instance of a data object or data attribute is mandatory or optional. Setting the visibility of a constructed data attribute also sets the visibility of the lower-level data attribute to the same value.

```
PROPERTY bPresence : BOOL {GET,SET}
```

bPresence: Visibility/existence:

- TRUE : = mandatory;
- FALSE : = optional;

6.5.1.6.6 TrgOp

This property specifies the trigger of a report (spontaneous transfer of data object or data attributes information).

```
PROPERTY TrgOp : DWORD {GET,SET}
```

TrgOp: Report trigger. One or more values can be ORed.

6.5.1.6.7 Fc

This property specifies the functional group to which a data object or data attribute belongs. Setting the functional group of a constructed data attribute also sets the visibility of the lower-level data attribute to the same value.

```
PROPERTY Fc : DWORD {GET,SET}
```

Fc: Functional group. One or more values can be ORed.

6.5.1.7 I_AcsiCommonDataSetClass

Objects that implement this interface are data set instances. TwinCAT Telecontrol Configurator generates data set instances, which are derived/extended from FB_AcsiCommonDataSetClass [▶ 154]. All function blocks derived from FB_AcsiCommonDataSetClass [▶ 154] automatically have the interface referred to above implemented by default.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataSetClass

```
INTERFACE I_AcsiCommonDataSetClass EXTENDS I_AcsiCommonNodeClass
```

Methods

Name	Description
AddMember [▶ 367]	Adds a new data object to the member container
DeleteMember [▶ 367]	Removes a data object from the member container
DeleteAllMembers [▶ 367]	Removes all data objects from the member container

Methods for event handling

Name	Description
OnInit [▶ 367]	Initializes/configures the data objects in the member container

Properties

Name	Type	Access	Description
nMembers	UDINT	Get	Number of data objects in the member container

6.5.1.7.1 AddMember

This method is used to add a data object (DO) to the member container of a data set.

```
METHOD FINAL AddMember : BOOL
VAR_INPUT
    ipData   : I_AcsiCommonDataClass;
    eFc      : E_AcsiFc;
END_VAR
VAR_OUTPUT
    bMatched : BOOL;
    ipDirNode: I_AcsiDataSetMemberClass;
END_VAR
```

ipData: Interface pointer of type I_AcsiCommonDataClass [▶ 364] of the data object, which is to be added to the member container.

eFc: Functional group of the data object of type E_AcsiFc [▶ 487].

bMatched: If TRUE, the data object is already in the member container.

ipDirNode: interface pointer of the new container list item.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.7.2 DeleteMember

This method is used to remove a data object (DO) from the member container of the data set.

```
METHOD FINAL DeleteMember : BOOL
VAR_INPUT
    ipData  : I_AcsiCommonDataClass;
    eFc     : E_AcsiFc;
END_VAR
```

ipData: Interface pointer of type I_AcsiCommonDataClass [▶ 364] of the data object, which is to be removed from the member container.

eFc: Functional group of the data object of type E_AcsiFc [▶ 487].

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.7.3 DeleteAllMembers

This method is used to remove all data objects (DO) and data attributes (DA) from the member container of the data set.

```
METHOD FINAL DeleteAllMembers : BOOL
VAR_OUTPUT
    nDeleted : UDINT;
END_VAR
```

nDeleted: Number of deleted data object and data attributes members.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.7.4 OnInit

This method can, but does not have to, be overloaded by the application. It can be used to fill the member container of the data set with the data object and data attribute references.

```
METHOD OnInit : BOOL
```

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.8 I_AcsiCommonGooseControlBlockClass

Objects implementing this interface are control block instances for Generic Object Oriented Substation Events (GOOSE). The TwinCAT Telecontrol Configurator generates GOOSE control block instances that are derived/extended from FB_AcsiCommonGooseControlBlockClass [▶ 155]. All function blocks derived from FB_AcsiCommonGooseControlBlockClass [▶ 155] automatically have a default implementation of the above interface.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AcsiCommonControlBlockClass [▶ 363] -> I_AcsiCommonGooseControlBlockClass

```
INTERFACE I_AcsiCommonGooseControlBlockClass EXTENDS I_AcsiCommonControlBlockClass
```

6.5.1.9 I_AcsiCommonIEDGroupClass

Objects implementing this interface represent a group of multiple IEDs. All IEDs in this group use the same "Real-Time Ethernet Adapter (Multiple Protocol Handler)" for communication. These objects serve as containers for multiple server instances (e.g. multiple GOOSE Subscribers or Publishers).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

__System.IQueryInterface -> I_AcsiCommonIEDGroupClass

```
INTERFACE I_AcsiCommonIEDGroupClass EXTENDS __System.IQueryInterface
```

Properties

Name	Type	Access	Description
nIEDs	UDINT	Get	Number of IEDs in the container.
sObjectName	T_AcsiObjectName [▶ 555]	Get	Object name.
Day	T_AcsiTag [▶ 556]	Get	Additional object information.

6.5.1.10 I_AcsiCommonIntelligentElectronicDeviceClass

Objects that implement this interface are located in the top hierarchical level of the IEC 61850 data model. This level can also be referred to as a server instance. The TwinCAT Telecontrol Configurator generates function blocks for this level, which are derived/extended from FB_AcsiCommonIntelligentElectronicDeviceClass [▶ 157]. All function blocks derived from FB_AcsiCommonIntelligentElectronicDeviceClass [▶ 157] automatically have the interface referred to above implemented by default.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonIntelligentElectronicDeviceClass

```
INTERFACE I_AcsiCommonIntelligentElectronicDeviceClass EXTENDS I_AcsiCommonNodeClass
```

Methods

Name	Description
AddLogicalDeviceToContainer [▶ 369]	Adds a new logical device to the server container

Properties

Name	Type	Access	Description
nLogicalDevices	UDINT	Get	Number of logical devices in the server container

6.5.1.10.1 AddLogicalDeviceToContainer

This method is used to add an object instance of the logical device (LD) to the server container (intelligent logical device, IED).

```
METHOD FINAL AddLogicalDeviceToContainer : BOOL
VAR_INPUT
    ipLogicalDevice : I_AcsiCommonLogicalDeviceClass;
END_VAR
VAR_OUTPUT
    bMatched        : BOOL;
    ipDirNode       : I_AcsiLogicalDeviceContainerClass;
END_VAR
```

ipLogicalDevice: Interface pointer of type I_AcsiCommonLogicalDeviceClass [▶ 369] of the logical device, which is to be added to the container.

bMatched: If TRUE, the object already exists in the container.

ipDirNode: interface pointer of the new container list item.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.11 I_AcsiCommonLogicalDeviceClass

Objects that implement this interface are located in the second hierarchical level of the IEC 61850 data model. This level is referred to a logical device (LD). The TwinCAT Telecontrol Configurator generates function blocks for this level, which are derived/extended from FB_AcsiCommonLogicalDeviceClass [▶ 157]. All function blocks derived from FB_AcsiCommonLogicalDeviceClass [▶ 157] automatically have the interface referred to above implemented by default.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonLogicalNodeClass

```
INTERFACE I_AcsiCommonLogicalDeviceClass EXTENDS I_AcsiCommonNodeClass
```

Methods

Name	Description
AddLogicalNodeToContainer [▶ 370]	Adds a new logical node

Properties

Name	Type	Access	Description
nLogicalNodes	UDINT	Get	Number of logical node in the container of the logical device

6.5.1.11.1 AddLogicalNodeToContainer

This method is used to add an object instance of the logical node (LN) to the container of the logical device (LD).

```
METHOD FINAL AddLogicalNodeToContainer : BOOL
VAR_INPUT
    ipLogicalNode : I_AcsiCommonLogicalNodeClass;
END_VAR
VAR_OUTPUT
    bMatched      : BOOL;
    ipDirNode     : I_AcsiLogicalNodeContainerClass;
END_VAR
```

ipLogicalNode: Interface pointer of type I_AcsiCommonLogicalNodeClass [▶ 370] of the logical node, which is to be added to the container.

bMatched: If TRUE, the object already exists in the container.

ipDirNode: interface pointer of the new container list item.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.12 I_AcsiCommonLogicalNodeClass

Objects that implement this interface are located in the third hierarchical level of the IEC 61850 data model. This level is referred to as logical node (LN). The TwinCAT Telecontrol Configurator generates function blocks for this level, which are derived/extended from FB_AcsiCommonLogicalNodeClass [▶ 158]. All function blocks derived from FB_AcsiCommonLogicalNodeClass [▶ 158] automatically have the interface referred to above implemented by default.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonLogicalNodeClass

```
INTERFACE I_AcsiCommonLogicalNodeClass EXTENDS I_AcsiCommonNodeClass
```

Methods

Name	Description
AddDataToContainer [▶ 371]	Adds a new data object
AddDataSetToContainer [▶ 371]	Adds a new data set
AddUnbufferedReportControlBlockToContainer [▶ 371]	Adds a new report control block instance for unbuffered reports
AddBufferedReportControlBlockToContainer [▶ 372]	Adds a new report control block instance for buffered reports
AddGooseControlBlockToContainer [▶ 372]	Adds a new goose control block instance

Properties

Name	Type	Access	Description
nDataSets	UDINT	Get	Number of data sets in the container of the logical node
nBufferedReportControlBlocks	UDINT	Get	Number of report control block instances for unbuffered reports
nUnbufferedReportControlBlocks	UDINT	Get	Number of report control block instances for buffered reports
nGooseControlBlocks	UDINT	Get	Number of goose control block instances

6.5.1.12.1 AddDataToContainer

This method is used to add a data object (DO) to the container of the logical node (LN).

```
METHOD FINAL AddDataToContainer : BOOL
VAR_INPUT
    ipData    : I_AcsiCommonDataClass;
END_VAR
VAR_OUTPUT
    bMatched  : BOOL;
    ipDirNode : I_AcsiDataContainerClass;
END_VAR
```

ipData: Interface pointer of type I_AcsiCommonAttributeClass [▶ 364] of the data object, which is to be added to the container.

bMatched: If TRUE, the object already exists in the container.

ipDirNode: interface pointer of the new container list item.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.12.2 AddDataSetToContainer

This method is used to add an object instance of the data set to the container of the logical node (LN).

```
METHOD FINAL AddDataSetToContainer : BOOL
VAR_INPUT
    ipDataSet : I_AcsiCommonDataSetClass;
END_VAR
VAR_OUTPUT
    bMatched  : BOOL;
    ipDirNode : I_AcsiDataSetContainerClass;
END_VAR
```

ipDataSet: Interface pointer of type I_AcsiCommonDataSetClass [▶ 366] of the data set, which is to be added to the container.

bMatched: If TRUE, the object already exists in the container.

ipDirNode: interface pointer of the new container list item.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.12.3 AddUnbufferedReportControlBlockToContainer

This method is used to add a report control block instance for unbuffered reports (unbuffered report control block, URCB) to the container of the logical node (LN).

```
METHOD FINAL AddUnbufferedReportControlBlockToContainer : BOOL
VAR_INPUT
    ipUnbufferedReportControlBlock : I_AcsiCommonUnbufferedReportControlBlockClass;
END_VAR
VAR_OUTPUT
    bMatched                       : BOOL;
    ipDirNode                      : I_AcsiUrCBContainerClass;
END_VAR
```

ipUnbufferedReportControlBlock: Interface pointer of type I_AcsiUnbufferedReportControlBlockClass [▶ 374] of the report control block instance, which is to be added to the container.

bMatched: If TRUE, the object already exists in the container.

ipDirNode: interface pointer of the new container list item.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.12.4 AddBufferedReportControlBlockToContainer

This method is used to add a report control block instance for buffered reports (Buffered Report Control Block, BRCB) to the container of the logical node (LN).

```
METHOD FINAL AddBufferedReportControlBlockToContainer : BOOL
VAR_INPUT
    ipBufferedReportControlBlock : I_AcsiCommonBufferedReportControlBlockClass;
END_VAR
VAR_OUTPUT
    bMatched                     : BOOL;
    ipDirNode                    : I_AcsiBrCBContainerClass;
END_VAR
```

ipBufferedReportControlBlock: Interface pointer of type I_AcsiBufferedReportControlBlockClass [▶ 363] of the report control block instance, which is to be added to the container.

bMatched: If TRUE, the object already exists in the container.

ipDirNode: interface pointer of the new container list item.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.12.5 AddGooseControlBlockToContainer

This method is used to add a goose control block instance (generic object oriented substation event control block, GCB) to the container of the logical node (LN).

```
METHOD FINAL AddGooseControlBlockToContainer : BOOL
VAR_INPUT
    ipGooseControlBlock : I_AcsiCommonGooseControlBlockClass;
END_VAR
VAR_OUTPUT
    bMatched            : BOOL;
    ipDirNode           : I_AcsiGoCBContainerClass;
END_VAR
```

ipGooseControlBlock: Interface pointer of type I_AcsiCommonGooseControlBlockClass of the goose control block instance, which is to be added to the container.

bMatched: If TRUE, the object already exists in the container.

ipDirNode: interface pointer of the new container list item.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.13 I_AcsiCommonMulticastSampledValueControlBlockClass

Objects that implement this interface are control block instances for Sampled Values (SV). The TwinCAT Telecontrol Configurator generates SV control block instances that are derived/extended from FB_AcsiCommonMulticastSampledValueControlBlockClass [▶ 159]. All function blocks derived from FB_AcsiCommonMulticastSampledValueControlBlockClass [▶ 159] automatically have a default implementation of the above interface.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AcsiCommonControlBlockClass [▶ 363] -> I_AcsiCommonMulticastSampledValueControlBlockClass

```
INTERFACE I_AcsiCommonMulticastSampledValueControlBlockClass EXTENDS I_AcsiCommonControlBlockClass
```

6.5.1.14 I_AcsiCommonNodeClass

Objects that implement this interface form the basis of the TwinCAT IEC 61850 data model. All function blocks generated by the TwinCAT Telecontrol Configurator implement this interface or are derived from the function block FB_AcsiCommonNodeClass [▶ 160], which implements this interface.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass

```
INTERFACE I_AcsiCommonNodeClass
```

Methods

Name	Description
GetObjectReference [▶ 373]	Returns the reference path of an object instance
GetServerObject [▶ 374]	Provides the interface pointer of the object instance of the server

Properties

Name	Type	Access	Description
sObjectName [▶ 374]	T_AcsiObjectName [▶ 555]	Get,Set	Object name

6.5.1.14.1 GetObjectReference

This method can be used to determine the reference path of an object instance of the TwinCAT IEC 61850 data model. The query of the server object instance (IEDs) returns the name of the server. In all other object instances above the server, the server name is not included in the reference path. The name of the functional group is not included in the reference path of the object instances above the logical nodes (LN).

```
METHOD GetObjectReference : T_AcsiObjectReference
```

Return parameter: String containing the reference path of the object instance.

Example:

Extract from a fictitious TwinCAT IEC 61850 data model:

```
MA2C03                      {IED}
    -MA2C03Relay            {LD}
         -LLN0              {LN}
              -Mod          {DO}
                   stVal    {DA}
                   q        {DA}
                   t        {DA}
              +Beh          {DO}
              +Health       {DO}
              +NamPlt       {DO}
         +LPHD1             {LN}
...
```

Declaration part:

```
PROGRAM MAIN
VAR
    sRef : T_AcsiObjectReference;
END_VAR
```

Implementation:

```
sRef := MA2C03.GetObjectReference();                                (*'MA2C03'*)
sRef := MA2C03.MA2C03Relay.GetObjectReference();                    (*'MA2C03Relay'*)
sRef := MA2C03.MA2C03Relay.LLN0.GetObjectReference();               (*'MA2C03Relay/LLN0'*)
sRef := MA2C03.MA2C03Relay.LLN0.Mod_.GetObjectReference();          (*'MA2C03Relay/LLN0.Mod'*)
sRef := MA2C03.MA2C03Relay.LLN0.Mod_.stVal.GetObjectReference();(*'MA2C03Relay/LLN0.Mod.stVal'*)
sRef := MA2C03.MA2C03Relay.LLN0.Mod_.q.GetObjectReference();        (*'MA2C03Relay/LLN0.Mod.q'*)
sRef := MA2C03.MA2C03Relay.LLN0.Mod_.t.GetObjectReference();        (*'MA2C03Relay/LLN0.Mod.t'*)
sRef := MA2C03.MA2C03Relay.LLN0.Beh.GetObjectReference();           (*'MA2C03Relay/LLN0.Beh'*)
sRef := MA2C03.MA2C03Relay.LLN0.Health.GetObjectReference();        (*'MA2C03Relay/LLN0.Health'*)
sRef := MA2C03.MA2C03Relay.LLN0.NamPlt.GetObjectReference();        (*'MA2C03Relay/LLN0.NamPlt'*)
sRef := MA2C03.MA2C03Relay.LPHD1.GetObjectReference();              (*'MA2C03Relay/LPHD1'*)
```

6.5.1.14.2 GetServerObject

This method can be used to determine the interface pointer of the server object instance (top level of the IEC 61850 data model, IED).

```
METHOD GetServerObject : BOOL
VAR_OUTPUT
    ipServer : I_AcsiCommonIntelligentElectronicDeviceClass;
END_VAR
```

ipServer: Interface pointer of type I_AcsiCommonIntelligentElectronicDeviceClass [▶ 368] of the server object instance.

Return parameter: Positive message (TRUE) on success, negative message (FALSE) on error.

6.5.1.14.3 sObjectName

```
PROPERTY sObjectName : T_AcsiObjectName (GET)
```

sObjectName: Object name as string of type T_AcsiObjectName [▶ 555].

6.5.1.15 I_AcsiCommonUnbufferedReportControlBlockClass

Objects that implement this interface are report control block instances for unbuffered reports (unbuffered report control blocks, UBRCBs). The TwinCAT Telecontrol Configurator generates report control block instances, which are derived/extended from FB_AcsiCommonUnbufferedReportControlBlockClass [▶ 160]. All function blocks derived from FB_AcsiCommonUnbufferedReportControlBlockClass [▶ 160] automatically have the interface referred to above implemented by default.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AcsiCommonControlBlockClass [▶ 363] -> I_AcsiCommonUnbufferedReportControlBlockClass

```
INTERFACE I_AcsiCommonUnbufferedReportControlBlockClass EXTENDS I_AcsiCommonControlBlockClass
```

6.5.1.16 I_AcsiDATypeLastApplError

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AsciCommonAttributeClass [▶ 362] -> I_AcsiDATypeLastApplError

```
INTERFACE I_AcsiDATypeLastApplError EXTENDS I_AcsiCommonAttributeClass
```

Properties

Name	Type	Access	Description
sCntrlObj	STRING(129)	Get,Set	Attribute value
eError	E_AcsiApplError [▶ 478]	Get,Set	Attribute value
stOrigin	ST_AcsiOriginator [▶ 538]	Get,Set	Attribute value
nCtlNum	BYTE	Get,Set	Attribute value
eAddCause	E_AcsiApplAddCause [▶ 476]	Get,Set	Attribute value

6.5.2 Protocol settings

6.5.2.1 I_GseAdapterSettingsClass

Objects that implement this interface are used to configure a network adapter that can be used for GSE communication.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

I_GseAdapterSettingsClass

```
INTERFACE I_GseAdapterSettingsClass
```

Properties

Name	Type	Access	Description
bExecGoCBPubs	BOOL	Get	Enables/disables automatic calling of the "Execute" method of the Publisher GOOSE control blocks by the adapter.
bExecGoCBSubs	BOOL	Get	Enables/disables automatic calling of the "Execute" method of the Subscriber GOOSE control blocks by the adapter.
bSimulation	BOOL	Get	Enables/disables the GOOSE simulation bit field.
eAdapterPrio	TcEthernetAdapterPriority [▶ 530]	Get	Priority of the network adapter.
eDispatchMode	E_GseDispatchMode [▶ 521]	Get	Setting whether received Ethernet telegrams are forwarded to the operating system.
macAddr	ETHERNET_ADDRESS [▶ 528]	Get Set	MAC address of the network adapter.
multicastAddr	ETHERNET_ADDRESS [▶ 528]	Get	Selected single subscriber multicast address of the network adapter. If set, the network adapter block instance receives only the subscriber messages sent to this exact address. In this case, the address is used as a filter for the subscriber messages. All other messages are forwarded either to the operating system or (if available) to further instances of the network adapter block. **INFO:** **From TwinCAT TF6510 IEC 61850 Telecontrol v3.1.98.0 and higher: If this address has not been set ('00-00-00-00-00-00'), then all multicast addresses and APPIDs configured on the subscriber control block instances are used as filters for the subscriber messages. In this case, different multicast addresses can be configured in the GOOSE control block instances, for example.** The configured Subscriber APPID's (if used > 0), must correspond to the APPID's on the Publisher page. All received messages that cannot be assigned to a control block are forwarded to the operating system or to further instances of the network adapter block.
oid	OTCID [▶ 529]	Get	TwinCAT object ID of the network adapter.
sMacAddr	STRING(17)	Get	Text representation of the MAC address of the network adapter.
sMulticastAddr	STRING(17)	Get Set	Text representation of the multicast address of the network adapter.

6.5.2.2 I_ScsmClientSettingsClass

Objects that implement this interface are used for configuring the SCSM communication layer (specific communication service mapping).

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_SocketClientSettingsClass [▶ 391] -> I_TpktClientSettingsClass [▶ 390] -> I_Rfc1006ClientSettingsClass [▶ 388] -> I_UlosiClientSettingsClass [▶ 384] -> I_MmsClientSettingsClass [▶ 378] -> I_ScsmClientSettingsClass

```
INTERFACE I_ScsmClientSettingsClass EXTENDS I_MmsClientSettingsClass
```

Methods

Name	Description
Defaults [▶ 377]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
bGetDataSetDirectoryMembersReload	BOOL	Get	Enables/disables the automatic initialization of the DataSets/DataSet member.
eScsmEdition	E_ScsmEdition [▶ 525]	Get	IEC 61850 data model edition (1 or 2)
tScsmTimeout	TIME	Get	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceScsm	T_TraceLevel [▶ 526]	Get	Configures the priority level for the logging of the error messages and log messages.

6.5.2.2.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.3 I_ScsmConnectionSettingsClass

Objects that implement this interface are used for configuring the SCSM communication layer (specific communication service mapping). The configuration settings only apply to a selected server session function block instance.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_SocketConnectionSettingsClass [▶ 392] -> I_TpktConnectionSettingsClass [▶ 390] -
> I_Rfc1006ConnectionSettingsClass [▶ 388] -> I_UlosiConnectionSettingsClass [▶ 386] -
> I_MmsConnectionSettingsClass [▶ 382] -> I_ScsmConnectionSettingsClass

```
INTERFACE I_ScsmConnectionSettingsClass EXTENDS I_MmsConnectionSettingsClass
```

Methods

Name	Description
Defaults [▶ 377]	Resets the configuration settings to the default values.

6.5.2.3.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.4 I_ScsmServerSettingsClass

Objects that implement this interface are used for configuring the SCSM communication layer (specific communication service mapping).

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_SocketServerSettingsClass [▶ 393] -> I_TpktServerSettingsClass [▶ 391] -> I_Rfc1006ServerSettingsClass [▶ 389] -> I_UlosiServerSettingsClass [▶ 386] -> I_MmsServerSettingsClass [▶ 382] -
> I_ScsmServerSettingsClass

```
INTERFACE I_ScsmServerSettingsClass EXTENDS I_MmsServerSettingsClass
```

Methods

Name	Description
Defaults [▶ 378]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
eScsmEdition	E_ScsmEdition [▶ 525]	Get	IEC 61850 data model edition (1 or 2)
tScsmTimeout	TIME	Get	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceScsm	T_TraceLevel [▶ 526]	Get	Configures the priority level for the logging of the error messages and log messages.

6.5.2.4.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.5 I_MmsClientSettingsClass

Objects that implement this interface are used to configure the MMS communication layer.

Namespace: Tc3_Mms [▶ 102]
Library: Tc3_Mms (Tc3_Mms.compiled-library)

Inheritance hierarchy

I_SocketClientSettingsClass [▶ 391] -> I_TpktClientSettingsClass [▶ 390] -> I_Rfc1006ClientSettingsClass [▶ 388] -> I_UlosiClientSettingsClass [▶ 384] -> I_MmsClientSettingsClass

```
INTERFACE I_MmsClientSettingsClass EXTENDS I_UlosiClientSettingsClass
```

Methods

Name	Description
Defaults [▶ 381]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
nLocalDetail	T_MmsInteger32 [▶ 557]	Get	MMS Initiate-RequestPDU parameter: localDetailCalling.
nMaxServOutCalling	T_MmsInteger16 [▶ 557]	Get	MMS Initiate-RequestPDU parameter: proposedMaxServOutstandingCalling
nMaxServOutCalled	T_MmsInteger16 [▶ 557]	Get	MMS Initiate-RequestPDU parameter: proposedMaxServOutstandingCalled.
nNestingLevel	T_MmsInteger8 [▶ 557]	Get	MMS Initiate-RequestPDU parameter: proposedDataStructureNestingLevel.
nMmsVersionNumber	T_MmsInteger16 [▶ 557]	Get	MMS initRequestDetail parameter: proposedVersionNumber.
ipParameters	I_MmsParameterSupportOptions [▶ 380]	Get	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: proposedParameterCBB.
ipServices	I_MmsServiceSupportOptions [▶ 380]	Get	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: servicesSupportedCalling.
ipExtendedServices	I_MmsAdditionalSupportOptions [▶ 381]	Get	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: additionalSupportedCalling.
ipExtendedParameters	I_MmsAdditionalCBBOptions [▶ 381]	Get	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: additionalCbbSupportedCalling.
sProvilegeClassIdentity	T_MmsVisibleString [▶ 558]	Get	MMS-initRequestDetail parameter: privilegeClassIdentityCalled.
tMmsTimeout	TIME	Get	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceMms	E_TraceLevel [▶ 526]	Get	Configures the priority level for the logging of the error messages and log messages.

6.5.2.5.1 I_MmsServiceSupportOptions

Configures the supported services of the MMS communication layer.

Inheritance hierarchy

I_MmsServiceSupportOptions

```
INTERFACE I_MmsServiceSupportOptions
```

Properties

Name	Type	Access	Description
bCancel	BOOL	Get,Set	
bConclude	BOOL	Get,Set	
bDefineNamedType	BOOL	Get,Set	
bDefineNamedVariable	BOOL	Get,Set	
bDefineNamedVariableList	BOOL	Get,Set	
bDefineScatteredAccess	BOOL	Get,Set	
bDeleteNamedType	BOOL	Get,Set	
bDeleteNamedVariableList	BOOL	Get,Set	
bDeleteVariableAccess	BOOL	Get,Set	
bGetDomainAttributes	BOOL	Get,Set	
bGetNamedTypeAttributes	BOOL	Get,Set	
bGetNamedVariableListAttributes	BOOL	Get,Set	
bGetNameList	BOOL	Get,Set	
bGetScatteredAccessAttributes	BOOL	Get,Set	
bGetVariableAccessAttributes	BOOL	Get,Set	
bIdentify	BOOL	Get,Set	
bInformationReport	BOOL	Get,Set	
bRead	BOOL	Get,Set	
bRename	BOOL	Get,Set	
bStatus	BOOL	Get,Set	
bWrite	BOOL	Get,Set	

6.5.2.5.2 I_MmsParameterSupportOptions

Configures the supported parameters of the MMS communication layer.

Inheritance hierarchy

I_MmsParameterSupportOptions

```
INTERFACE I_MmsParameterSupportOptions
```

Properties

Name	Type	Access	Description
bCspi	BOOL	Get,Set	
bCsr	BOOL	Get,Set	
bStr1	BOOL	Get,Set	Array support
bStr2	BOOL	Get,Set	Structure support
bVadr	BOOL	Get,Set	
bValt	BOOL	Get,Set	Support for index and index range values (alternate access)
bVlis	BOOL	Get,Set	Support for namedVariableList
bVnam	BOOL	Get,Set	Support for namedVariables

6.5.2.5.3 I_MmsAdditionalSupportOptions

Configures the additionally supported MMS services.

Inheritance hierarchy

I_MmsAdditionalSupportOptions

```
INTERFACE I_MmsAdditionalSupportOptions
```

Properties

Name	Type	Access	Description
bVmdReset	BOOL	Get,Set	
bVmdStop	BOOL	Get,Set	
bSelect	BOOL	Get, Set	

6.5.2.5.4 I_MmsAdditionalCBBOptions

Configures the additionally supported MMS CBB parameters.

Inheritance hierarchy

I_MmsAdditionalCBBOptions

```
INTERFACE I_MmsAdditionalCBBOptions
```

Properties

Name	Type	Access	Description
bDei	BOOL	Get,Set	
bDes	BOOL	Get,Set	
bRed	BOOL	Get, Set	

6.5.2.5.5 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.6 I_MmsConnectionSettingsClass

Objects that implement this interface are used to configure the MMS communication layer. The configuration settings only apply to a selected server session function block instance.

Namespace: Tc3_Mms [▶ 102]
Library: Tc3_Mms (Tc3_Mms.compiled-library)

Inheritance hierarchy

I_SocketConnectionSettingsClass [▶ 392] -> I_TpktConnectionSettingsClass [▶ 390] -
> I_Rfc1006ConnectionSettingsClass [▶ 388] -> I_UlosiConnectionSettingsClass [▶ 386] -
> I_MmsConnectionSettingsClass

```
INTERFACE I_MmsConnectionSettingsClass EXTENDS I_UlosiConnectionSettingsClass
```

Methods

Name	Description
Defaults [▶ 382]	Resets the configuration settings to the default values.

6.5.2.6.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.7 I_MmsServerSettingsClass

Objects that implement this interface are used to configure the MMS communication layer.

Namespace: Tc3_Mms [▶ 102]
Library: Tc3_Mms (Tc3_Mms.compiled-library)

Inheritance hierarchy

I_SocketServerSettingsClass [▶ 393] -> I_TpktServerSettingsClass [▶ 391] -> I_Rfc1006ServerSettingsClass [▶ 389] -> I_UlosiServerSettingsClass [▶ 386] -> I_MmsServerSettingsClass

```
INTERFACE I_MmsServerSettingsClass EXTENDS I_UlosiServerSettingsClass
```

Methods

Name	Description
Defaults [▶ 383]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
nLocalDetail	T_MmsInteger32 [▶ 557]	Get	MMS Initiate-RequestPDU parameter: localDetailCalling.
nMaxServOutCalling	T_MmsInteger16 [▶ 557]	Get	MMS Initiate-RequestPDU parameter: proposedMaxServOutstandingCalling
nMaxServOutCalled	T_MmsInteger16 [▶ 557]	Get	MMS Initiate-RequestPDU parameter: proposedMaxServOutstandingCalled.
nNestingLevel	T_MmsInteger8 [▶ 557]	Get	MMS Initiate-RequestPDU parameter: proposedDataStructureNestingLevel.
nMmsVersionNumber	T_MmsInteger16 [▶ 557]	Get	MMS initRequestDetail parameter: proposedVersionNumber.
ipParameters	I_MmsParameterSupportOptions [▶ 380]	Get	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: proposedParameterCBB.
ipServices	I_MmsServiceSupportOptions [▶ 380]	Get	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: servicesSupportedCalling.
ipExtendedServices	I_MmsAdditionalSupportOptions [▶ 381]	Get	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: additionalSupportedCalling.
ipExtendedParameters	I_MmsAdditionalCBBOptions [▶ 381]	Get	Interface pointer of the object for the configuration of the MMS initRequestDetail parameter: additionalCbbSupportedCalling.
sProvilegeClassIdentity	T_MmsVisibleString [▶ 558]	Get	MMS-initRequestDetail parameter: privilegeClassIdentityCalled.
tMmsTimeout	TIME	Get	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceMms	E_TraceLevel [▶ 526]	Get	Configures the priority level for the logging of the error messages and log messages.

6.5.2.7.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.8 I_UlosiClientSettingsClass

Objects that implement this interface are used to configure the ULOSI communication layer (upper layer OSI).

Namespace: Tc3_Ulosi [▶ 102]
Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Inheritance hierarchy

I_SocketClientSettingsClass [▶ 391] -> I_TpktClientSettingsClass [▶ 390] -> I_Rfc1006ClientSettingsClass [▶ 388] -> I_UlosiClientSettingsClass

```
INTERFACE I_UlosiClientSettingsClass EXTENDS I_Rfc1006ClientSettingsClass
```

Methods

Name	Description
Defaults [▶ 385]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
sCalling_S_Selector	STRING	Get	Calling-S-selector.
sCalled_S_Selector	STRING	Get	Called-S-selector.
nSessionRequirement	WORD	Get	Session requirements.
sCalling_P_Selector	STRING	Get	Calling-P-selector.
sCalled_P_Selector	STRING	Get	Called-P-selector.
nPresentRequirements	BYTE	Get	Presentation requirements.
nPresentProtocolVersion	BYTE	Get	Presentation protocol version.
nProtocol_Version	BYTE	Get	Association Control Service protocol version.
sContext_Name	STRING	Get	Application context name (object identifier, dotted presentation), MMS.
sCalled_AP_Title	STRING	Get	Called-AP-title.
nCalled_AE_Qualifier	UDINT	Get	Called-AE-qualifier.
nCalled_AP_InvID	UDINT	Get	Called-AP-invocation-identifier.
nCalled_AE_InvID	UDINT	Get	Called-AE-invocation-identifier.
sCalling_AP_Title	STRING	Get	Calling-AP-title.
nCalling_AE_Qualifier	UDINT	Get	Calling-AE-qualifier.
nCalling_AP_InvID	UDINT	Get	Calling-AP-invocation-identifier.
nCalling_AE_InvID	UDINT	Get	Calling-AE-invocation-identifier.
nRequirements	BYTE	Get	Association Control Service requirements.
sMechanism_Name	STRING	Get	Authentication mechanism name.
sAuthent_Value	STRING	Get	Authentication value (password).
sResponding_Value	STRING	Get	Responding authentication value (password).
eAcse_Authent	E_AcseAuthentication [▶ 474]	Get	Authentication mode.
tUlosiTimeout	TIME	Get	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceUlosi	E_TraceLevel [▶ 526]	Get	Configures the priority level for the logging of the error messages and log messages.

6.5.2.8.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.9 I_UlosiConnectionSettingsClass

Objects that implement this interface are used to configure the ULOSI communication layer (upper layer OSI). The configuration settings only apply to a selected server session function block instance.

Namespace: Tc3_Ulosi [▶ 102]
Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Inheritance hierarchy

I_SocketConnectionSettingsClass [▶ 392] -> I_TpktConnectionSettingsClass [▶ 390] -> I_Rfc1006ConnectionSettingsClass [▶ 388] -> I_UlosiConnectionSettingsClass

```
INTERFACE I_UlosiConnectionSettingsClass EXTENDS I_Rfc1006ConnectionSettingsClass
```

Methods

Name	Description
Defaults [▶ 386]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
sAuthent_Value	STRING	Get	Connection-specific authentication password. Remote communication partner (client) connected to the selected server session function block instance can successfully authenticate with the server using this password.

6.5.2.9.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.10 I_UlosiServerSettingsClass

Objects that implement this interface are used to configure the ULOSI communication layer (upper layer OSI).

Namespace: Tc3_Ulosi [▶ 102]
Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Inheritance hierarchy

I_SocketServerSettingsClass [▶ 393] -> I_TpktServerSettingsClass [▶ 391] -> I_Rfc1006ServerSettingsClass [▶ 389] -> I_UlosiServerSettingsClass

```
INTERFACE I_UlosiServerSettingsClass EXTENDS I_Rfc1006ServerSettingsClass
```

Methods

Name	Description
Defaults [▶ 387]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
sCalling_S_Selector	STRING	Get	Calling-S-selector.
sCalled_S_Selector	STRING	Get	Called-S-selector.
nSessionRequirement	WORD	Get	Session requirements.
sCalling_P_Selector	STRING	Get	Calling-P-selector.
sCalled_P_Selector	STRING	Get	Called-P-selector.
nPresentRequirements	BYTE	Get	Presentation requirements.
nPresentProtocolVersion	BYTE	Get	Presentation protocol version.
nProtocol_Version	BYTE	Get	Association Control Service protocol version.
sContext_Name	STRING	Get	Application context name (object identifier, dotted presentation), MMS.
sCalled_AP_Title	STRING	Get	Called-AP-title.
nCalled_AE_Qualifier	UDINT	Get	Called-AE-qualifier.
nCalled_AP_InvID	UDINT	Get	Called-AP-invocation-identifier.
nCalled_AE_InvID	UDINT	Get	Called-AE-invocation-identifier.
sCalling_AP_Title	STRING	Get	Calling-AP-title.
nCalling_AE_Qualifier	UDINT	Get	Calling-AE-qualifier.
nCalling_AP_InvID	UDINT	Get	Calling-AP-invocation-identifier.
nCalling_AE_InvID	UDINT	Get	Calling-AE-invocation-identifier.
nRequirements	BYTE	Get	Association Control Service requirements.
sMechanism_Name	STRING	Get	Authentication mechanism name.
sAuthent_Value	STRING	Get	Authentication value (password).
sResponding_Value	STRING	Get	Responding authentication value (password).
eAcse_Authent	E_AcseAuthentication [▶ 474]	Get	Authentication mode.
tUlosiTimeout	TIME	Get	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceUlosi	E_TraceLevel [▶ 526]	Get	Configures the priority level for the logging of the error messages and log messages.

6.5.2.10.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.11 I_Rfc1006ClientSettingsClass

Objects that implement this interface are used to configure the RFC1006 communication layer.

Namespace: Tc3_Rfc1006 [▶ 102]
Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Inheritance hierarchy

I_SocketClientSettingsClass [▶ 391] -> I_TpktClientSettingsClass [▶ 390] -> I_Rfc1006ClientSettingsClass

```
INTERFACE I_Rfc1006ClientSettingsClass EXTENDS I_TpktClientSettingsClass
```

Methods

Name	Description
Defaults [▶ 388]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
sCalling_T_Selector	STRING	Get	RFC 1006 parameter: Calling transport address selector (source).
sCalled_T_Selector	STRING	Get	RFC 1006 parameter: Called transport address selector (destination).
eTpduSize	E_Rfc1006TpduSize [▶ 524]	Get	Maximum TPDU byte length.
tRfc1006Timeout	TIME	Get	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceRfc1006	E_TraceLevel [▶ 526]	Get	Configures the priority level for the logging of the error messages and log messages.

6.5.2.11.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.12 I_Rfc1006ConnectionSettingsClass

Objects that implement this interface are used to configure the RFC1006 communication layer. The configuration settings only apply to a selected server session function block instance.

Namespace: Tc3_Rfc1006 [▶ 102]
Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Inheritance hierarchy

I_SocketConnectionSettingsClass [▶ 392] -> I_TpktConnectionSettingsClass [▶ 390] -
> I_Rfc1006ConnectionSettingsClass

```
INTERFACE I_Rfc1006ConnectionSettingsClass EXTENDS I_TpktConnectionSettingsClass
```

Methods

Name	Description
Defaults [▶ 389]	Resets the configuration settings to the default values.

6.5.2.12.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.13 I_Rfc1006ServerSettingsClass

Objects that implement this interface are used to configure the RFC1006 communication layer.

Namespace: Tc3_Rfc1006 [▶ 102]
Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Inheritance hierarchy

I_SocketServerSettingsClass [▶ 393] -> I_TpktServerSettingsClass [▶ 391] -> I_Rfc1006ServerSettingsClass

```
INTERFACE I_Rfc1006ServerSettingsClass EXTENDS I_TpktServerSettingsClass
```

Methods

Name	Description
Defaults [▶ 389]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
sCalling_T_Selector	STRING	Get	RFC 1006 parameter: Calling transport address selector (source).
sCalled_T_Selector	STRING	Get	RFC 1006 parameter: Called transport address selector (destination).
eTpduSize	E_Rfc1006TpduSize [▶ 524]	Get	Maximum TPDU byte length.
tRfc1006Timeout	TIME	Get	Maximum time that may not be exceeded during the execution of the confirmed services.

6.5.2.13.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.14 I_TpktClientSettingsClass

Objects that implement this interface are used to configure the TPKT communication layer (ISO transport services on top of the TCP).

Namespace: Tc3_Tpkt [▶ 102]
Library: Tc3_Tpkt (Tc3_Tpkt.compiled-library)

Inheritance hierarchy

I_SocketClientSettingsClass [▶ 391] -> I_TpktClientSettingsClass

```
INTERFACE I_TpktClientSettingsClass EXTENDS I_SocketClientSettingsClass
```

Methods

Name	Description
Defaults [▶ 390]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
tTpktTimeout	TIME	Get	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceTpkt	E_TraceLevel [▶ 526]	Get	Configures the priority level for the logging of the error messages and log messages.

6.5.2.14.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.15 I_TpktConnectionSettingsClass

Objects that implement this interface are used to configure the TPKT communication layer (ISO transport services on top of the TCP). The configuration settings only apply to a selected server session function block instance.

Namespace: Tc3_Tpkt [▶ 102]
Library: Tc3_Tpkt (Tc3_Tpkt.compiled-library)

Inheritance hierarchy

I_SocketConnectionSettingsClass [▶ 392] -> I_TpktConnectionSettingsClass

```
INTERFACE I_TpktConnectionSettingsClass EXTENDS I_SocketConnectionSettingsClass
```

Methods

Name	Description
Defaults [▶ 391]	Resets the configuration settings to the default values.

6.5.2.15.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.16 I_TpktServerSettingsClass

Objects that implement this interface are used to configure the TPKT communication layer (ISO transport services on top of the TCP).

Namespace: Tc3_Tpkt [▶ 102]
Library: Tc3_Tpkt (Tc3_Tpkt.compiled-library)

Inheritance hierarchy

I_SocketServerSetttngsClass [▶ 393] -> I_TpktServerSettingsClass

```
INTERFACE I_TpktServerSettingsClass EXTENDS I_SocketServerSettingsClass
```

Methods

Name	Description
Defaults [▶ 391]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
tTpktTimeout	TIME	Get	Maximum time that may not be exceeded during the execution of the confirmed services.
eTraceTpkt	E_TraceLevel [▶ 526]	Get	Configures the priority level for the logging of the error messages and log messages.

6.5.2.16.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.17 I_SocketClientSettingsClass

Objects that implement this interface are used to configure the TCP/IP communication layer.

Namespace: Tc3_Sockets [▶ 102]
Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Inheritance hierarchy

I_SocketClientSettingsClass -> __System.IQueryInterface

```
INTERFACE I_SocketClientSettingsClass EXTENDS __System.IQueryInterface
```

Methods

Name	Description
Defaults [▶ 392]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
bAutoCleanup	BOOL	Get	Enables/disables the automatic clean-up/ closure of the unused sockets (e.g. after program download or PLC reset).
sSrvNetID	T_AmsNetID [▶ 529]	Get	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
nRemotePort	UDINT	Get	TCP/IP port number of the server.
sRemoteHost	T_Ipv4Addr [▶ 529]	Get	IPv4 network address of the server.
tConnect	TIME	Get	Earliest time on expiry of which a new connection can be established.
tAdsTimeout	TIME	Get	Maximum time that must not be exceeded during the execution of the ADS services.
tSocketTimeout	TIME	Get	Maximum time that may not be exceeded during the execution of the confirmed services.
bThrottleMode	BOOL	Get	Enables/disables the receive data throttle polling mode.
throttleTimes	T_ThrottleTimes [▶ 530]	Get	Configuration of the cycle times for the receive data throttle polling mode.
eTraceSocket	E_TraceLevel [▶ 526]	Get	Configures the priority level for the logging of the error messages and log messages.
ipTLS	I_SocketTlsSettingsClass [▶ 394]	Get	Configuration parameters of the TLS (Transport Layer Security) protocol layer.
bEvaLinkStatusErr	BOOL	Get	Configures the evaluation mode for the link status error of the network adapter.

6.5.2.17.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.18 I_SocketConnectionSettingsClass

Objects that implement this interface are used to configure the TCP/IP communication layer. The configuration settings only apply to a selected server session function block instance.

Namespace: Tc3_Sockets [▶ 102]
Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Inheritance hierarchy

I_SocketConnectionSettingsClass -> __System.IQueryInterface

```
INTERFACE I_SocketConnectionSettingsClass EXTENDS __System.IQueryInterface
```

Methods

Name	Description
Defaults [▶ 393]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
bEnable	BOOL	Get	Reads the configured activation mode for incoming connections.
bTrace	BOOL	Get	Reads the configured activation mode for trace log messages.
eAccept	E_SocketAcceptMode	Get	Reads the configured acceptance mode for incoming connections.
nAcceptPort	UDINT	Get	Reads the configured TCP/IP port number to be accepted.
sAcceptHost	T_IPv4Addr [▶ 529]	Get	Reads out the configured IPv4 network address to be accepted.

6.5.2.18.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.19 I_SocketServerSettingsClass

Objects that implement this interface are used to configure the TCP/IP communication layer.

Namespace: Tc3_Sockets [▶ 102]
Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Inheritance hierarchy

I_SocketServerSettingsClass -> __System.IQueryInterface

```
INTERFACE I_SocketServerSettingsClass EXTENDS __System.IQueryInterface
```

Methods

Name	Description
Defaults [▶ 394]	Resets the configuration settings to the default values.

Properties

Name	Type	Access	Description
bAutoCleanup	BOOL	Get	Enables/disables the automatic clean-up/closure of the unused sockets (e.g. after program download or PLC reset).
bEnable	BOOL	Get	Enables/disables establishing the server connections. "TRUE" enables the Listen command (listening and incoming connections are accepted) and "FALSE" disables it.
sSrvNetID	T_AmsNetID [▶ 529]	Get	TwinCAT network address of the TwinCAT TCP/IP server (TF6310). With the empty string, the network address of the local TwinCAT system is used.
nLocalPort	UDINT	Get	TCP/IP port number of the server.
sLocalHost	T_Ipv4Addr [▶ 529]	Get	IPv4 network address of the server.
tListen	TIME	Get	Retry cycle time for the internal TCP/IP socket Listen command.
tAccept	TIME	Get	Retry cycle time for the internal TCP/IP socket Accept commands.
tAdsTimeout	TIME	Get	Maximum time that must not be exceeded during the execution of the ADS services.
tSocketTimeout	TIME	Get	Maximum time that may not be exceeded during the execution of the confirmed services.
bThrottleMode	BOOL	Get	Enables/disables the receive data throttle polling mode.
throttleTimes	T_ThrottleTimes [▶ 530]	Get	Configuration of the cycle times for the receive data throttle polling mode.
eTraceSocket	E_TraceLevel [▶ 526]	Get	Configures the priority level for the logging of the error messages and log messages.
ipTLS	I_SocketTlsSettingsClass [▶ 394]	Get	Configuration parameters of the TLS (Transport Layer Security) protocol layer.
bEvaLinkStatusErr	BOOL	Get	Configures the evaluation mode for the link status error of the network adapter.

6.5.2.19.1 Defaults

This method resets the configuration settings to the default values.

```
METHOD Defaults
VAR_INPUT
    bAllOf : BOOL;
END_VAR
```

bAllOf: The value "FALSE" only resets the settings of the derived (current) class. If the value is "TRUE", the inherited settings of all base classes are also reset.

6.5.2.20 I_SocketTlsSettingsClass

Objects that implement this interface are used to configure the TLS (Transport Layer Security) communication layer.

Namespace: Tc3_Sockets [▶ 102]
Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Inheritance hierarchy

I_SocketTlsSettingsClass -> __System.IQueryInterface

```
INTERFACE I_SocketTlsSettingsClass EXTENDS __System.IQueryInterface
```

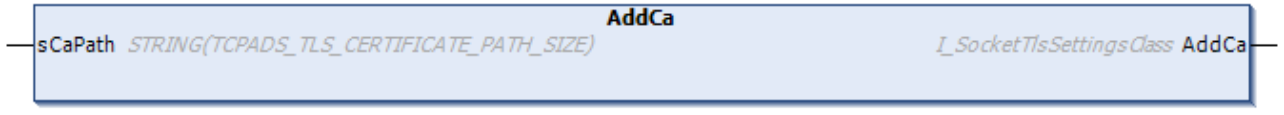
Methods

Name	Description
AddCa [▶ 395]	Saves the file path of the CA certificate in the TLS configuration settings.
AddCert [▶ 396]	Saves the file path of the client or server certificate in the TLS configuration settings.
AddCrl [▶ 396]	Saves the file path of the Certificate Revocation List (CRL) in the TLS configuration settings.
AddPsk [▶ 396]	Saves the PSK key and PSK identity string in the TLS configuration settings.
Reset [▶ 397]	Resets all TLS configuration settings.
SetConnectFlags [▶ 397]	Saves additional, optional TLS configuration settings of the client.
SetListenFlags [▶ 397]	Saves additional, optional TLS configuration settings of the server.

Properties

Name	Type	Access	Description
nSecurePort	UDINT	Get	Secure ISO TP0 (IANA) port number for TLS communication.
connectFlags	ST_TlsConnectFlags	Get	Additional TLS configuration settings of the client.
listenFlags	ST_TlsListenFlags	Get	Additional TLS configuration settings of the server.
pskKey	PVOID	Get	Points to the first byte of the PSK key.
pskKeyLen	UDINT(0..TCPADS_TLS_MAX_PSK_KEY_SIZE)	Get	Byte length of the PSK key.
sCaPath	STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE)	Get	File path of the CA certificate.
sCertPath	STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE)	Get	File path of the client or server certificate.
sCrlPath	STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE)	Get	File path from the certificate revocation list (CRL).
sIdentity	STRING(TCPADS_TLS_PSK_IDENTITY_SIZE)	Get	Identity string of the PSK key.
sKeyPath	STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE)	Get	File path of the private key.
sKeyPwd	STRING(TCPADS_TLS_KEY_PASSWORD_SIZE)	Get	Password string of the certificate.

6.5.2.20.1 AddCa



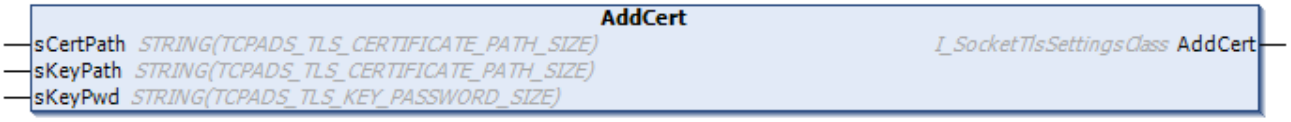
This method saves the file path of the CA certificate in the TLS configuration settings.

```
METHOD FINAL AddCa : I_SocketTlsSettingsClass
VAR_INPUT
    sCaPath : STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE);
END_VAR
```

sCaPath: File path of the CA certificate.

Return parameter: Own interface pointer of type I_SocketTlsSettingsClass [▶ 394].

6.5.2.20.2 AddCert



This method saves the file path of the server or client certificate in the TLS configuration settings.

```
METHOD FINAL AddCert : I_SocketTlsSettingsClass
VAR_INPUT
    sCertPath : STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE);
    sKeyPath  : STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE);
    sKeyPwd   : STRING(TCPADS_TLS_KEY_PASSWORD_SIZE);
END_VAR
```

sCertPath: File path of the server or client certificate.

sKeyPath: File path of the private key.

sKeyPwd: Password string of the certificate.

Return parameter: Own interface pointer of type I_SocketTlsSettingsClass [▶ 394].

6.5.2.20.3 AddCrl



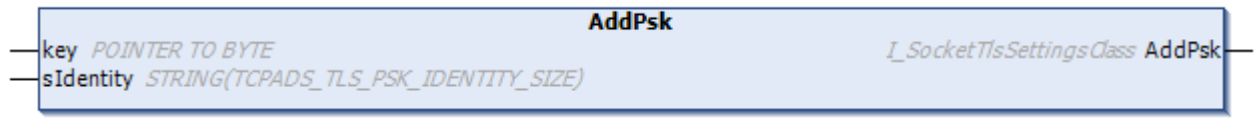
This method saves the file path of the certificate revocation list (CRL) in the TLS configuration settings.

```
METHOD FINAL AddCrl : I_SocketTlsSettingsClass
VAR_INPUT
    sCrlPath : STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE);
END_VAR
```

sCrlPath: File path of the certificate revocation list (CRL).

Return parameter: Own interface pointer of type I_SocketTlsSettingsClass [▶ 394].

6.5.2.20.4 AddPsk



This method saves the PSK (pre-shared key) and the PSK identity string in the TLS configuration settings.

```
METHOD FINAL AddPsk : I_SocketTlsSettingsClass
VAR_IN_OUT
    key       : ARRAY[*] OF BYTE;
END_VAR
VAR_INPUT
    sIdentity : STRING(TCPADS_TLS_PSK_IDENTITY_SIZE);
END_VAR
```

key: Data buffer with the PSK key (variable length array).

sIdentity: PSK Identity string.

Return parameter: Own interface pointer of type I_SocketTlsSettingsClass [▶ 394].

6.5.2.20.5 Reset

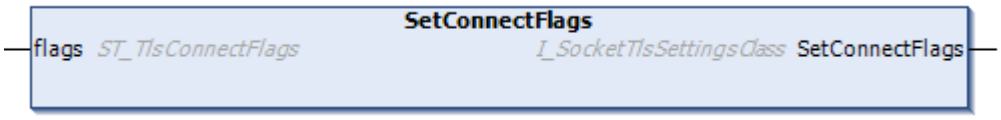


This method resets all TLS configuration settings.

```
METHOD FINAL Reset : I_SocketTlsSettingsClass
```

Return parameter: Own interface pointer of type I_SocketTlsSettingsClass [▶ 394].

6.5.2.20.6 SetConnectFlags



This method saves additional, optional TLS configuration settings of the client.

```
METHOD FINAL SetConnectFlags : I_SocketTlsSettingsClass
VAR_INPUT
    flags : ST_TlsConnectFlags;
END_VAR
```

flags: Parameter of type ST_TlsConnectFlags with additional, optional TLS configuration settings of the client. The value of the "flags" member variable "bNoserverCertCheck" determines whether the server certificate is checked (FALSE, default) or not checked (TRUE). The value of the "flags" member variable "bIgnoreCnMismatch" determines whether a mismatch in the "CommonName" of the certificate is ignored (TRUE) or reported as an error (FALSE, default).

Return parameter: Own interface pointer of type I_SocketTlsSettingsClass [▶ 394].

6.5.2.20.7 SetListenFlags



This method saves additional, optional TLS configuration settings of the server.

```
METHOD FINAL SetListenFlags : I_SocketTlsSettingsClass
VAR_INPUT
    flags : ST_TlsListenFlags;
END_VAR
```

flags: parameter of type ST_TlsListenFlags with additional, optional TLS configuration settings of the server. The value of the "flags" member variable "bNoClientCert" determines whether the configuration of the client certificate is required (FALSE, default) or not required (TRUE).

Return parameter: Own interface pointer of type I_SocketTlsSettingsClass [▶ 394].

6.5.3 Communication

6.5.3.1 GOOSE

6.5.3.1.1 I_GseAdapterClass

Interface pointers of this type identify network adapters usable for GSE communication.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

I_GseAdapterClass

```
INTERFACE I_GseAdapterClass
```

Properties

Name	Type	Access	Description
ipSettings	I_GseAdapterSettingsClass [▶ 375]	Get	Protocol settings of the GSE adapter.

6.5.3.1.2 I_GseGoCBImplClass

Objects implementing this interface are control block instances for Generic Object Oriented Substation Events (GOOSE). For each configured GOOSE control block, the TwinCAT Telecontrol Configurator generates an instance of the function block FB_GseGoCBImplClass [▶ 275], which implements this interface.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AcsiCommonControlBlockClass [▶ 363] -> I_AcsiCommonGooseControlBlockClass [▶ 368] -> I_ScsmGoCBImplClass [▶ 458] -> I_GseGoCBImplClass

```
INTERFACE I_GseGoCBImplClass EXTENDS I_ScsmGoCBImplClass
```

Properties

Name	Type	Access	Description
ipPublisher	I_GseGoCBPublisherClass [▶ 398]	Get	Implementation of the Publisher.
ipSubscriber	I_GseGoCBSubscriberClass [▶ 400]	Get	Implementation of the Subscriber.

6.5.3.1.3 I_GseGoCBPublisherClass

Objects that implement this interface control the behavior of a GOOSE Publisher.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

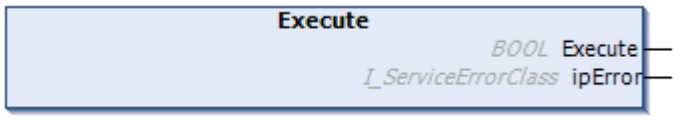
I_ServiceErrorClass [▶ 468] -> I_GseGoCBPublisherClass

```
INTERFACE I_GseGoCBPublisherClass EXTENDS I_ServiceErrorClass
```

Methods

Name	Description
Execute [▶ 399]	Executes the Publisher state machine.
Start [▶ 399]	Starts sending GOOSE messages.
Stop [▶ 399]	Stops sending GOOSE messages.
Update [▶ 399]	Triggers an immediate sending of a GOOSE message.

6.5.3.1.3.1 Execute



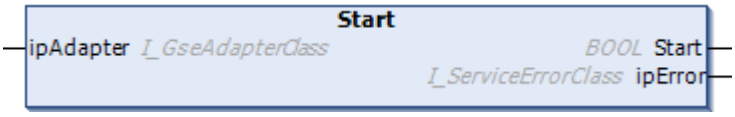
Executes the Publisher state machine.

```
METHOD Execute : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.1.3.2 Start



Starts sending GOOSE messages.

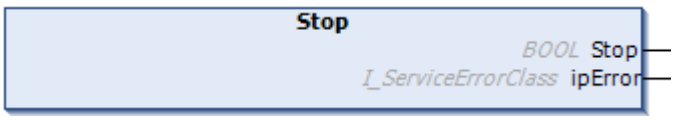
```
METHOD Start : BOOL
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
END_VAR
VAR_OUTPUT
    ipError   : I_ServiceErrorClass;
END_VAR
```

ipAdapter: Interface pointer of type I_GseAdapterClass [▶ 397]. This pointer identifies the network adapter to be used for sending/receiving GOOSE/SV messages.

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.1.3.3 Stop



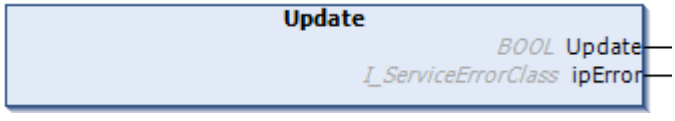
Stops sending GOOSE messages.

```
METHOD Stop : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.1.3.4 Update



Triggers an immediate sending of a new GOOSE message. New messages are sent when the value of GOOSE dataset members changes. Message replays are sent automatically, in the background, triggered by the Execute method of the Publisher state machine.

```
METHOD Update : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.1.4 I_GseGoCBSubscriberClass

Objects that implement this interface control the behavior of a GOOSE Subscriber.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

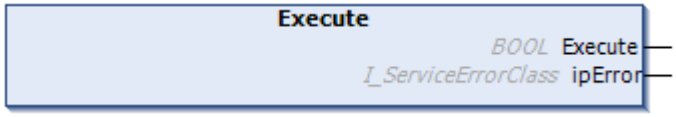
I_ServiceErrorClass [▶ 468] -> I_GseGoCBSubscriberClass

```
INTERFACE I_GseGoCBSubscriberClass EXTENDS I_ServiceErrorClass
```

Methods

Name	Description
Execute [▶ 400]	Executes the Subscriber state machine.
Enable [▶ 400]	Starts receiving GOOSE messages.
Disable [▶ 401]	Stops receiving GOOSE messages.

6.5.3.1.4.1 Execute



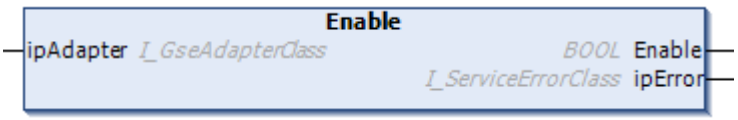
Executes the Subscriber state machine.

```
METHOD Execute : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.1.4.2 Enable



Starts receiving GOOSE messages.

```
METHOD Enable : BOOL
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
END_VAR
```

```
VAR_OUTPUT
    ipError   : I_ServiceErrorClass;
END_VAR
```

ipAdapter: Interface pointer of type I_GseAdapterClass [▶ 397]. This pointer identifies the network adapter to be used for sending/receiving GOOSE/SV messages.

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.1.4.3 Disable



Stops receiving GOOSE messages.

```
METHOD Disable : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.1.5 I_GseLinkStatusEventSink

Objects that implement this interface have a user-defined "LinkStatusChange" service event handling routine.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

I_GseLinkStatusEventSink

```
INTERFACE I_GseLinkStatusEventSink EXTENDS __System.IQueryInterface
```

Methods for event handling

Name	Description
OnLinkStatusChange [▶ 401]	User-defined "LinkStatusChange" service event handling routine

6.5.3.1.5.1 OnLinkStatusChange

```
METHOD OnLinkStatusChange
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    eStatus   : E_GseLinkStatus;
END_VAR
```

ipAdapter: interface pointer of type: I_GseAdapterClass [▶ 397]. This pointer indicates the instance whose status has changed.

eStatus: the current status of type E_GseLinkStatus [▶ 522].

6.5.3.1.6 I_GseSystemClockEventSink

Objects that implement this interface have a user-defined system clock service event handling routine for access to an external time source.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

I_GseSystemClockEventSink

```
INTERFACE I_GseSystemClockEventSink EXTENDS __System.IQueryInterface
```

Methods for event handling

Name	Description
OnGetSystemTime [▶ 402]	User-defined system clock service event handling routine

6.5.3.1.6.1 OnGetSystemTime

```
METHOD OnGetSystemTime : BOOL
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
END_VAR
VAR_OUTPUT
    tT        : T_UtcTime;
END_VAR
```

ipAdapter: Interface pointer of type: I_GseAdapterClass [▶ 397]. This pointer specifies the instance that queries the current system time.

tT: current UTC time information of type T_UtcTime [▶ 560].

6.5.3.2 Sampled Values

6.5.3.2.1 I_GseMsvCBImplClass

Objects that implement this interface are control block instances for SV (Sampled Values). For each configured SV control block, the TwinCAT Telecontrol Configurator generates an instance of the FB_GseMsvCBImplClass function block, which implements this interface.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AcsiCommonControlBlockClass [▶ 363] -> I_AcsiCommonMulticastSampledValueControlBlockClass [▶ 372]-> I_ScsmMsvCBImplClass [▶ 462] -> I_GseMsvCBImplClass

```
INTERFACE I_GseMsvCBImplClass EXTENDS I_ScsmMsvCBImplClass
```

Properties

Name	Type	Access	Description
ipPublisher	I_GseMsvCBPublisherClass [▶ 403]	Get	Implementation of the Publisher.
ipSubscriber	I_GseMsvCBSubscriberClass [▶ 404]	Get	Implementation of the Subscriber.

6.5.3.2.2 I_GseMsvCBPublisherClass

Objects that implement this interface control the behavior of an SV Publisher.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

I_ServiceErrorClass [▶ 468] -> I_GseMsvCBPublisherClass

```
INTERFACE I_GseMsvCBPublisherClass EXTENDS I_ServiceErrorClass
```

Methods

Name	Description
Execute [▶ 403]	Executes the Publisher state machine.
Start [▶ 403]	Starts sending SV messages.
Stop [▶ 404]	Stops sending SV messages.
Update [▶ 404]	Triggers an immediate sending of an SV message.

6.5.3.2.2.1 Execute



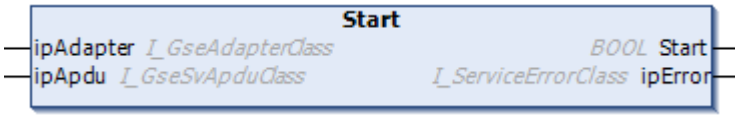
Executes the SV Publisher state machine.

```
METHOD Execute : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.2.2.2 Start



Starts sending SV messages.

```
METHOD Start : BOOL
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    ipApdu    : I_GseSvApduClass;
END_VAR
VAR_OUTPUT
    ipError   : I_ServiceErrorClass;
END_VAR
```

ipAdapter: Interface pointer of type I_GseAdapterClass [▶ 397]. This pointer identifies the network adapter to be used for sending/receiving GOOSE/SV messages.

ipApdu: Interface pointer of type I_GseSvApduClass [▶ 405]. This pointer identifies the object instance of the SV APDU (Sampled Values Application Protocol Data Unit) data.

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.2.2.3 Stop



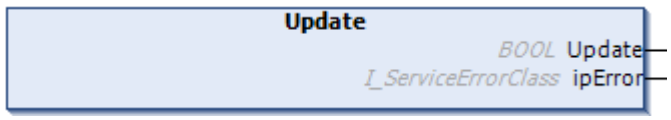
Stops sending SV messages

```
METHOD Stop : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.2.2.4 Update



Triggers an immediate sending of a new SV message.

```
METHOD Update : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.2.3 I_GseMsvCBSubscriberClass

Objects that implement this interface control the behavior of an SV Subscriber.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

I_ServiceErrorClass [▶ 468] -> I_GseMsvCBSubscriberClass

```
INTERFACE I_GseMsvCBSubscriberClass EXTENDS I_ServiceErrorClass
```

Methods

Name	Description
Execute [▶ 404]	Executes the Subscriber state machine.
Enable [▶ 405]	Starts receiving SV messages.
Disable [▶ 405]	Stops receiving SV messages.

6.5.3.2.3.1 Execute



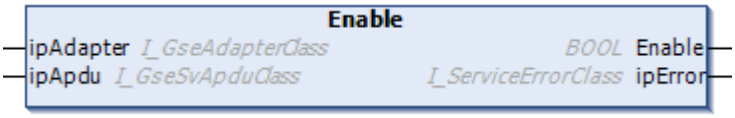
Executes the SV Subscriber state machine.

```
METHOD Execute : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.2.3.2 Enable



Starts receiving SV messages.

```
METHOD Enable : BOOL
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    ipApdu    : I_GseSvApduClass;
END_VAR
VAR_OUTPUT
    ipError   : I_ServiceErrorClass;
END_VAR
```

ipAdapter: Interface pointer of type I_GseAdapterClass [▶ 397]. This pointer identifies the network adapter to be used for sending/receiving GOOSE/SV messages.

ipApdu: Interface pointer of type I_GseSvApduClass [▶ 405]. This pointer identifies the object instance of the SV APDU (Sampled Values Application Protocol Data Unit) data.

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.2.3.3 Disable



Stops receiving SV messages.

```
METHOD Disable : BOOL
VAR_OUTPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to query the error code, error source or error text in the event of a negative feedback.

Return parameter: Positive feedback (TRUE) on success or negative feedback (FALSE) on error.

6.5.3.2.4 I_GseSvApduClass

Objects that implement this interface are function block instances that are responsible for mapping and encoding the SV-APDU data.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

I_GseSvApduClass

```
INTERFACE I_GseSvApduClass EXTENDS __SYSTEM.IQueryInterface
```

Properties

Name	Type	Access	Description
APPID	WORD	Get	APPID
bSimulation	BOOL	Get	Simulation flag in the Ethernet frame header.

Methods

Name	Description
OnSvPublisherHostToNet [▶ 406]	Host-to-network mapping routine
OnSvSubscriberNetToHost [▶ 406]	Network-to-host mapping routine.
Equals	Compares itself and another object for equality.

6.5.3.2.4.1 OnSvPublisherHostToNet

In this method, the SV publisher data is converted from the host byte order (little-endian) to the network byte order (big-endian) for transmission.

```
METHOD OnSvPublisherHostToNet
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    ipApdu    : I_GseSvApduClass;
END_VAR
```

ipAdapter: Interface pointer of type I_GseAdapterClass [▶ 397]. This pointer identifies the network adapter to be used for sending/receiving GOOSE/SV messages.

ipApdu: Interface pointer of type I_GseSvApduClass [▶ 405]. This pointer identifies the object instance of the SV APDU (Sampled Values Application Protocol Data Unit) data.

6.5.3.2.4.2 OnSvSubscriberNetToHost

In this method, the received SV subscriber data is converted from network byte order (big-endian) to host byte order (little-endian).

```
METHOD OnSvSubscriberNetToHost
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    ipApdu    : I_GseSvApduClass;
END_VAR
```

ipAdapter: Interface pointer of type I_GseAdapterClass [▶ 397]. This pointer identifies the network adapter to be used for sending/receiving GOOSE/SV messages.

ipApdu: Interface pointer of type I_GseSvApduClass [▶ 405]. This pointer identifies the object instance of the SV APDU (Sampled Values Application Protocol Data Unit) data.

6.5.3.2.5 I_GseSvAsduClass

Objects that implement this interface are function block instances that are responsible for mapping and encoding the SV-ASDU data.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

I_GseSvAsduClass

```
INTERFACE I_GseSvAsduClass EXTENDS __SYSTEM.IQueryInterface
```

Properties

Name	Type	Access	Description
eSmpMod	E_AcsiSamplingMode [▶ 503]	Get	Sampling mode.
hUser	PVOID	Get	User-defined handle (also pointer).
index	UDINT	Get	Index of the ASDU object (for several ASDUs in one SV message).
ipDataSet	I_AcsiCommonDataSetClass [▶ 366]	Get	Interface to the SV data record.
nConfRev	UDINT	Get	Version info of the SV configuration (always 1 for SV).
nSmpCnt	UINT	Get	Sampling value counter.
nSmpRate	UINT	Get	Sampling rate.
nSmpSynch	USINT	Get	Synchronization mode of the sampled values.
sDatSet	T_AcsiObjectReference [▶ 555]	Get	SV data record Reference string.
sSvID	T_AcsiVisString129 [▶ 556]	Get	SV identification string.
tRefrTm	T_UtcTime [▶ 560]	Get	SV refresh time.

6.5.3.2.6 I_GseSvPublisherEventSink

Objects that implement this interface have a user-defined event handling routine for converting the SV data to be sent from host to network format.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

I_GseSvPublisherEventSink

```
INTERFACE I_GseSvPublisherEventSink EXTENDS __SYSTEM.IQueryInterface
```

Methods for event handling

Name	Description
OnSvPublisherHostToNet [▶ 407]	User-defined event handling routine for SV data format conversion

6.5.3.2.6.1 OnSvPublisherHostToNet

In this method, the SV publisher data is converted from the host byte order (little-endian) to the network byte order (big-endian) for transmission.

```
METHOD OnSvPublisherHostToNet
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    ipApdu    : I_GseSvApduClass;
END_VAR
```

ipAdapter: Interface pointer of type I_GseAdapterClass [▶ 397]. This pointer identifies the network adapter to be used for sending/receiving GOOSE/SV messages.

ipApdu: Interface pointer of type I_GseSvApduClass [▶ 405]. This pointer identifies the object instance of the SV APDU (Sampled Values Application Protocol Data Unit) data.

6.5.3.2.7 I_GseSvSubscriberEventSink

Objects that implement this interface have a user-defined event handling routine for converting the received SV data from network to host format.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

I_GseSvSubscriberEventSink

```
INTERFACE I_GseSvSubscriberEventSink EXTENDS __SYSTEM.IQueryInterface
```

Methods for event handling

Name	Description
OnSvSubscriberNetToHost [▶ 408]	User-defined event handling routine for SV data format conversion

6.5.3.2.7.1 OnSvSubscriberNetToHost

In this method, the received SV subscriber data is converted from network byte order (big-endian) to host byte order (little-endian).

```
METHOD OnSvSubscriberNetToHost
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    ipApdu    : I_GseSvApduClass;
END_VAR
```

ipAdapter: Interface pointer of type I_GseAdapterClass [▶ 397]. This pointer identifies the network adapter to be used for sending/receiving GOOSE/SV messages.

ipApdu: Interface pointer of type I_GseSvApduClass [▶ 405]. This pointer identifies the object instance of the SV APDU (Sampled Values Application Protocol Data Unit) data.

6.5.3.2.8 I_GseSvUpdateEventSink

Objects that implement this interface have a user-defined “OnSvUpdate” service event handling routine.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Inheritance hierarchy

I_GseSvUpdateEventSink

```
INTERFACE I_GseSvUpdateEventSink EXTENDS __SYSTEM.IQueryInterface
```

Methods for event handling

Name	Description
OnSvUpdate [▶ 408]	User-defined “OnSvUpdate” service event handling routine.

6.5.3.2.8.1 OnSvUpdate

This method receives notifications about updates to the SV APDU data (applies only to the SV subscriber).

```
METHOD OnSvUpdate
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    ipApdu    : I_GseSvApduClass;
END_VAR
```

ipAdapter: Interface pointer of type I_GseAdapterClass [▶ 397]. This pointer identifies the network adapter to be used for sending/receiving GOOSE/SV messages.

ipApdu: Interface pointer of type I_GseSvApduClass [▶ 405]. This pointer identifies the object instance of the SV APDU (Sampled Values Application Protocol Data Unit) data.

6.5.3.3 I_ScsmClientClass

Objects implementing this interface control the communication setup and data exchange between the TwinCAT PLC client application and the IEC 61850 server.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmClientClass

```
INTERFACE I_ScsmClientClass
```

Methods

Name	Description
AssociateReq [▶ 413]	Enables the service: Associate
ReleaseReq [▶ 428]	Enables the service: Release
ReleaseRsp [▶ 428]	Responds to the service: Release
AbortReq [▶ 413]	Enables the service: Abort
ControlCancelReq [▶ 414]	Enables the service: Cancel
ControlOperateReq [▶ 415]	Enables the service: Operate
ControlSelectReq [▶ 416]	Enables the service: Select
ControlSelectWithValueReq [▶ 417]	Enables the service: SelectWithValue
CreateDataSetReq [▶ 418]	Enables the service: CreateDataSet
DeleteDataSetReq [▶ 419]	Enables the service: DeleteDataSet
GetAllDataValuesReq [▶ 419]	Enables the service: GetAllDataValues
GetAllLogicalDeviceValuesReq [▶ 420]	Enables the service: GetAllLogicalDeviceValues (TwinCAT-specific)
GetAllServerValuesReq [▶ 420]	Enables the service: GetAllServerValues (TwinCAT-specific)
GetBrCBValuesReq [▶ 421]	Enables the service: GetBRCBValues
GetDataSetDirectoryReq [▶ 422]	Enables the service: GetDataSetDirectory
GetDataSetValuesReq [▶ 422]	Enables the service: GetDataSetValues
GetDataValuesReq [▶ 423]	Enables the service: GetDataValues
GetGoCBValuesReq [▶ 424]	Enables the service: GetGOCBValues
GetLogicalDeviceDirectoryReq [▶ 424]	Enables the service: GetLogicalDeviceDirectory
GetLogicalNodeDirectoryReq [▶ 425]	Enables the service: GetLogicalNodeDirectory
GetMsvCBValuesReq [▶ 425]	Enables the service: GetMSVCBValues
GetServerDirectoryReq [▶ 426]	Enables the service: GetServerDirectory
GetUrCBValuesReq [▶ 427]	Enables the service: GetURCBValues
SetAllDataValuesReq [▶ 429]	Enables the service: SetAllDataValues (TwinCAT-specific)
SetBrCBValuesReq [▶ 429]	Enables the service: SetBRCBValues
SetDataSetValuesReq [▶ 430]	Enables the service: SetDataSetValues (TwinCAT-specific)
SetDataValuesReq [▶ 431]	Enables the service: SetDataValues
SetGoCBValuesReq [▶ 431]	Enables the service: SetGOCBValues
SetMsvCBValuesReq [▶ 432]	Enables the service: SetMSVCBValues
SetUrCBValuesReq [▶ 433]	Enables the service: SetURCBValues
IdentifyReq [▶ 427]	Enables the service: MMS-Identify
StatusReq [▶ 433]	Enables the service: MMS-Status

Properties

Name	Type	Access	Description
ipIED [▶ 428]	I_AcsiCommonIntelligentElectronicDeviceClass [▶ 368]	Get	Interface pointer of an object with the implementation of the top-level of the IEC 61850 data model
eState	E_AsyncEnvironmentState [▶ 519]	Get	Environmental condition of the communication connection.
ipSettings	I_ScsmClientSettingsClass [▶ 376]	Get	Protocol settings for the communication layer.
ipAssociateCnf	I_ScsmAssociateCnfEventSink [▶ 439]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnAssociateCnf [▶ 439]
ipReleaseCnf	I_ScsmReleaseCnfEventSink [▶ 450]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnReleaseCnf [▶ 450].
ipReleaseInd	I_ScsmReleaseIndEventSink [▶ 451]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnReleaseInd [▶ 451].
ipAbortInd	I_ScsmAbortIndEventSink [▶ 438]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnAbortInd [▶ 438]
ipSystemClock	I_ScsmSystemClockEventSink [▶ 454]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetSystemTime [▶ 454]
ipControlCnf	I_ScsmControlCnfEventSink [▶ 443]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnControlCnf [▶ 444]
ipCreateDataSetCnf	I_ScsmCreateDataSetCnfEventSink [▶ 444]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnCreateDataSetCnf [▶ 444]
ipDeleteDataSetCnf	I_ScsmDeleteDataSetCnfEventSink [▶ 445]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnDeleteDataSetCnf [▶ 445]
ipGetDataSetDirectory	I_ScsmGetDataSetDirectoryEventSink [▶ 445]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetDataSetDirectoryCnf [▶ 446]
ipGetDataValuesCnf	I_ScsmGetDataValuesCnfEventSink [▶ 446]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetDataValuesCnf [▶ 446]
ipGetLogicalDeviceDirectoryCnf	I_ScsmGetLogicalDeviceDirectoryCnfEventSink [▶ 447]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetLogicalDeviceDirectoryCnf [▶ 447]
ipGetLogicalNodeDirectoryCnf	I_ScsmGetLogicalNodeDirectoryCnfEventSink [▶ 447]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetLogicalNodeDirectoryCnf [▶ 448]
ipGetServerDirectoryCnf	I_ScsmGetServerDirectoryCnfEventSink [▶ 448]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnGetServerDirectoryCnf [▶ 448].
ipLastApplErrorInd	I_ScsmLastApplErrorIndEventSink [▶ 449]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnLastApplErrorInd [▶ 450]

Name	Type	Ac-ces s	Description
ipReportInd	I_ScsmReportIndEventSink [▶ 451]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnBufferedReportInd [▶ 451] and OnUnbufferedReportInd [▶ 452]
ipSetDataValuesCnf	I_ScsmSetDataValuesCnfEventSink [▶ 452]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnSetDataValuesCnf [▶ 453]
ipIdentifyCnf	I_ScsmIdentifyCnfEventSink [▶ 449]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnIdentifyCnf [▶ 449]
ipStatusCnf	I_ScsmStatusCnfEventSink [▶ 453]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnStatusCnf [▶ 453]

6.5.3.3.1 AbortReq



This method activates the Abort service. A client can use this service to disconnect from the server in a controlled manner.

```
METHOD AbortReq : BOOL
VAR_INPUT
    ipReason : I_ServiceErrorClass;
    ipSink   : I_ScsmAbortIndEventSink;
END_VAR
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipReason: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to pass an object with a user-defined cause for the disconnection.

ipSink: Interface pointer of type I_ScsmAbortIndEventSink [▶ 438] of an object with a user-defined implementation of the event handling routine: OnAbortInd (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.2 AssociateReq



This method activates the Associate service. A client can use this service to establish a connection to the server.

```
METHOD AssociateReq : BOOL
VAR_INPUT
    ipSink   : I_ScsmAssociateCnfEventSink;
END_VAR
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipSink: Interface pointer of type I_ScsmAssociateCnfEventSink [▶ 439] of an object with a user-defined implementation of the event handling routine OnAssociateCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

Example (extract):

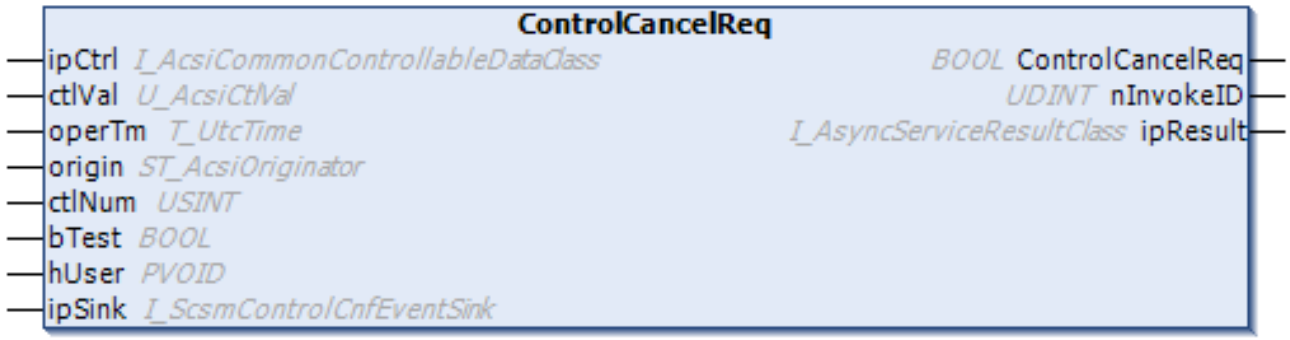
Declaration part:

```
FUNCTION_BLOCK FB_My61850Client IMPLEMENTS I_ScsmAbortEventSink
VAR
    fbClient         : FB_IEC61850CommonClass := ( ipIED := IED, settings := ( nRemotePort := 102, s
RemoteHost := '192.168.10.145' ), ipAbort := THIS^ );
    state            : BYTE;
    bSuccess         : BOOL;
    ipAsyncResult    : I_AsyncServiceResultClass;
END_VAR
```

Implementation:

```
CASE state OF
    0:(* idle state *)
        IF bAssociateReq THEN(* Establish connection *)
            bAssociateReq := FALSE;
            bSuccess := fbClient.AssociateReq( ipSink := 0, ipResult=>ipAsyncResult );
            state := SEL( bSuccess, 100, 1 );
        END_IF
    1:(* wait until connection established *)
        IF ipAsyncResult <> 0 THEN
            IF NOT ipAsyncResult.IsBusy() THEN
                state := SEL( ipAsyncResult.IsCompleted(), 100, 10 );
            END_IF
        END_IF
    10:(* connection established *)
        ;
    100:(* error state *)
        state := 0;
END_CASE
```

6.5.3.3.3 ControlCancelReq



This method activates the Cancel service.

```
METHOD ControlCancelReq : BOOL
VAR_INPUT
    ipCtrl    : I_AcsiCommonControllableDataClass;
    ctlVal    : U_AcsiCtlVal;
```

```
    operTm    : T_UtcTime;
    origin    : ST_AcsiOriginator;
    ctlNum    : USINT;
    bTest     : BOOL;
    hUser     : PVOID;
    ipSink    : I_ScsmControlCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipCtrl: Interface pointer of type I_AcsiCommonControllableDataClass [▶ 364] of a controllable object. Objects that implement this interface can be used to run switch control services such as Operate, Select, SelectWithValue or Cancel. Such objects include instances of Controllable Double Point (DPC) or Controllable Single Point (SPC).

ctlVal: Parameter for the new control value of type U_AcsiCtlVal [▶ 561].

operTm: Parameter for the switching time (operating time) of type T_UtcTime [▶ 560] for time-controlled command execution. Set this parameter to the value: operTm := AcsiConstants.NULL_TimeStamp if the controllable object does not support the service: TimeActivatedOperate.

origin: Parameter for the command origin of type ST_AcsiOriginator [▶ 538], through with a client issuing the command can be identified.

ctlNum: Identifier (e.g. sequential number) for the command, via which the client can assign the received negative or positive command confirmations to the request.

bTest: Test flag.

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type I_ScsmControlCnfEventSink [▶ 443] from an object with a user-defined implementation of the event handling routine: OnControlCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.4 ControlOperateReq



This method activates the Operate service.

```
METHOD ControlOperateReq : BOOL
VAR_INPUT
    ipCtrl    : I_AcsiCommonControllableDataClass;
```

```
    ctlVal    : U_AcsiCtlVal;
    operTm    : T_UtcTime;
    origin    : ST_AcsiOriginator;
    ctlNum    : USINT;
    bTest     : BOOL;
    eCheck    : E_AcsiCheck;
    hUser     : PVOID;
    ipSink    : I_ScsmControlCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipCtrl: Interface pointer of type I_AcsiCommonControllableDataClass [▶ 364] of a controllable object. Objects that implement this interface can be used to run switch control services such as Operate, Select, SelectWithValue or Cancel. Such objects include instances of Controllable Double Point (DPC) or Controllable Single Point (SPC).

ctlVal: Parameter for the new control value of type U_AcsiCtlVal [▶ 561].

operTm: Parameter for the switching time (operating time) of type T_UtcTime [▶ 560] for time-controlled command execution. Set this parameter to the value: operTm := AcsiConstants.NULL_TimeStamp if the controllable object does not support the service: TimeActivatedOperate.

origin: Parameter for the command origin of type ST_AcsiOriginator [▶ 538], through with a client issuing the command can be identified.

ctlNum: Identifier (e.g. sequential number) for the command, via which the client can assign the received negative or positive command confirmations to the request.

bTest: Test flag.

eCheck: Parameter of type E_AcsiCheck [▶ 479] for synchro check and/or interlocking tests.

ipSink: Interface pointer of type I_ScsmControlCnfEventSink [▶ 443] from an object with a user-defined implementation of the event handling routine: OnControlCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.5 ControlSelectReq



This method activates the Select service.

```
METHOD ControlSelectReq : BOOL
VAR_INPUT
    ipCtrl    : I_AcsiCommonControllableDataClass;
    hUser     : PVOID;
    ipSink    : I_ScsmControlCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipCtrl: Interface pointer of type I_AcsiCommonControllableDataClass [▶ 364] of a controllable object. Objects that implement this interface can be used to run switch control services such as Operate, Select, SelectWithValue or Cancel. Such objects include instances of Controllable Double Point (DPC) or Controllable Single Point (SPC).

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type I_ScsmControlCnfEventSink [▶ 443] from an object with a user-defined implementation of the event handling routine: OnControlCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.6 ControlSelectWithValueReq



This method activates the SelectWithValue service.

```
METHOD ControlSelectWithValueReq : BOOL
VAR_INPUT
    ipCtrl    : I_AcsiCommonControllableDataClass;
    ctlVal    : U_AcsiCtlVal;
    operTm    : T_UtcTime;
    origin    : ST_AcsiOriginator;
    ctlNum    : USINT;
    bTest     : BOOL;
    eCheck    : E_AcsiCheck;
    hUser     : PVOID;
    ipSink    : I_ScsmControlCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipCtrl: Interface pointer of type I_AcsiCommonControllableDataClass [▶ 364] of a controllable object. Objects that implement this interface can be used to run switch control services such as Operate, Select, SelectWithValue or Cancel. Such objects include instances of Controllable Double Point (DPC) or Controllable Single Point (SPC).

ctlVal: Parameter for the new control value of type U_AcsiCtlVal [▶ 561].

operTm: Parameter for the switching time (operating time) of type T_UtcTime [▶ 560] for time-controlled command execution. Set this parameter to the value: operTm := AcsiConstants.NULL_TimeStamp if the controllable object does not support the service: TimeActivatedOperate.

origin: Parameter for the command origin of type ST_AcsiOriginator [▶ 538], through with a client issuing the command can be identified.

ctlNum: Identifier (e.g. sequential number) for the command, via which the client can assign the received negative or positive command confirmations to the request.

bTest: Test flag.

eCheck: Parameter of type E_AcsiCheck [▶ 479] for synchro check and/or interlocking tests.

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

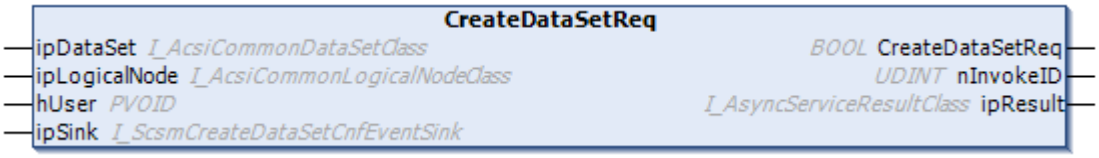
ipSink: Interface pointer of type I_ScsmControlCnfEventSink [▶ 443] from an object with a user-defined implementation of the event handling routine: OnControlCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.7 CreateDataSetReq



This method enables the service: CreateDataSet. This service allows the client to dynamically (at runtime) create a "persistent" or "non-persistent" data set.

```
METHOD CreateDataSetReq : BOOL
VAR_INPUT
    ipDataSet     : I_AcsiCommonDataSetClass;
    ipLogicalNode : I_AcsiCommonLogicalNodeClass;
    hUser         : PVOID;
    ipSink        : I_ScsmCreateDataSetCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID     : UDINT;
    ipResult      : I_AsyncServiceResultClass;
END_VAR
```

ipDataSet: Interface pointer of type I_AcsiCommonDataSetClass [▶ 366].

ipLogicalNode: Interface pointer of type I_AcsiCommonLogicalNodeClass [▶ 370].

The pointer to the logical node determines whether a "persistent" or "non-persistent" data set should be created. The "persistent" data sets are linked to a logical node. In this case, this parameter must be valid. In the case of a "non-persistent" data set, the parameter has the value null.

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type: I_ScsmCreateDataSetCnfEventSink [▶ 444] from an object with a user-defined implementation of the event handling routine: OnCreateDataSetCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.8 DeleteDataSetReq



This method enables the service: DeleteDataSet. This service allows the client to delete a dynamically (at runtime) created data set instance.

```
METHOD DeleteDataSetReq : BOOL
VAR_INPUT
    ipDataSet : I_AcsiCommonDataSetClass;
    hUser     : PVOID;
    ipSink    : I_ScsmDeleteDataSetCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipDataSet: Interface pointer of type I_AcsiCommonDataSetClass [▶ 366].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

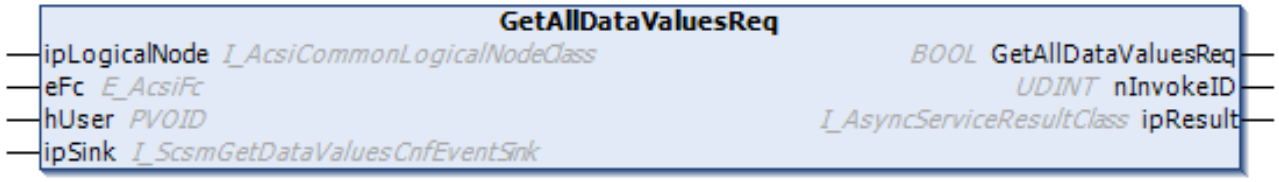
ipSink: Interface pointer of type: I_ScsmDeleteDataSetCnfEventSink [▶ 445] from an object with a user-defined implementation of the event handling routine: OnDeleteDataSetCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.9 GetAllDataValuesReq



This method activates the GetAllDataValues service.

```
METHOD GetAllDataValuesReq : BOOL
VAR_INPUT
    ipLogicalNode : I_AcsiCommonLogicalNodeClass;
    eFc           : E_AcsiFc;
    hUser         : PVOID;
    ipSink        : I_ScsmGetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID     : UDINT;
    ipResult      : I_AsyncServiceResultClass;
END_VAR
```

ipLogicalNode: Interface pointer of type I_AcsiCommonLogicalNodeClass [▶ 370].

eFc: Functional group of type E_AcsiFc [▶ 487].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

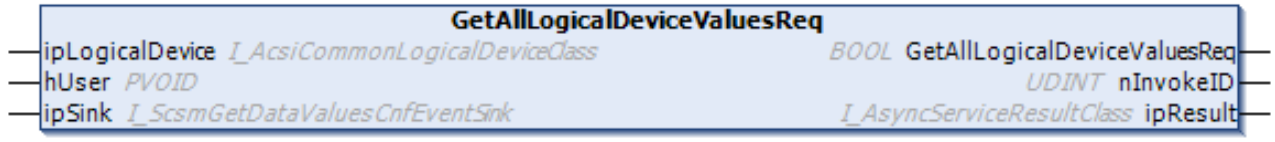
ipSink: Interface pointer of type I_ScsmGetDataValuesCnfEventSink [▶ 446] of an object with a user-defined implementation of the event handling routine: OnGetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.10 GetAllLogicalDeviceValuesReq



This method activates the GetAllLogicalDeviceValues service.

```
METHOD GetAllLogicalDeviceValuesReq : BOOL
VAR_INPUT
    ipLogicalDevice : I_AcsiCommonLogicalDeviceClass;
    hUser           : PVOID;
    ipSink          : I_ScsmGetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID       : UDINT;
    ipResult        : I_AsyncServiceResultClass;
END_VAR
```

ipLogicalDevice: Interface pointer of type I_AcsiCommonLogicalDeviceClass [▶ 369].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type I_ScsmGetDataValuesCnfEventSink [▶ 446] of an object with a user-defined implementation of the event handling routine: OnGetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.11 GetAllServerValuesReq



This method activates the GetAllServerValues service.

```
METHOD GetAllServerValuesReq : BOOL
VAR_INPUT
    ipServer  : I_AcsiCommonIntelligentElectronicDeviceClass;
    hUser     : PVOID;
    ipSink    : I_ScsmGetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipServer: Interface pointer of type I_AcsiCommonIntelligentElectronicDeviceClass [▶ 368].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

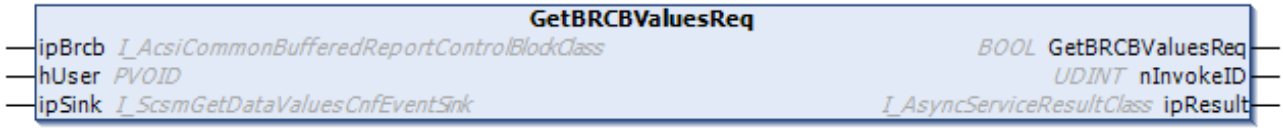
ipSink: Interface pointer of type I_ScsmGetDataValuesCnfEventSink [▶ 446] of an object with a user-defined implementation of the event handling routine: OnGetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.12 GetBrCBValuesReq



This method activates the GetBRCBValues service. The client can use this service to read the values of all data objects of a buffered report control block.

```
METHOD GetBRCBValuesReq : BOOL
VAR_INPUT
    ipBrcb    : I_AcsiCommonBufferedReportControlBlockClass;
    hUser     : PVOID;
    ipSink    : I_ScsmGetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipBrcb: Interface pointer of type I_AcsiCommonBufferedReportControlBlockClass [▶ 363].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

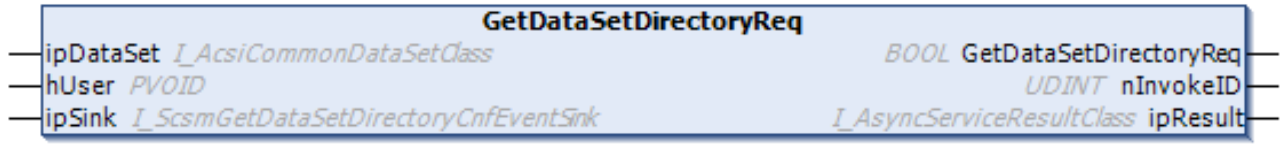
ipSink: Interface pointer of type I_ScsmGetDataValuesCnfEventSink [▶ 446] of an object with a user-defined implementation of the event handling routine: OnGetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.13 GetDataSetDirectoryReq



This method activates the GetDataSetDirectory service. A client can use this service to query the server to determine which data objects are referenced in a particular data set.

```
METHOD GetDataSetDirectoryReq : BOOL
VAR_INPUT
    ipDataSet : I_AcsiCommonDataSetClass;
    hUser     : PVOID;
    ipSink    : I_ScsmGetDataSetDirectoryCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipDataSet: Interface pointer of type I_AcsiCommonDataSetClass [▶ 366].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type I_ScsmGetDataSetDirectoryCnfEventSink [▶ 445] of an object with a user-defined implementation of the event handling routine: OnGetDataSetDirectoryCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.14 GetDataSetValuesReq



This method activates the GetDataSetValues service.

```
METHOD GetDataSetValuesReq : BOOL
VAR_INPUT
    ipDataSet : I_AcsiCommonDataSetClass;
    hUser     : PVOID;
    ipSink    : I_ScsmGetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipDataSet: Interface pointer of type I_AcsiCommonDataSetClass [▶ 366].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type I_ScsmGetDataValuesCnfEventSink [▶ 446] of an object with a user-defined implementation of the event handling routine: OnGetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.15 GetDataValuesReq



This method activates the service: GetDataValues.

```
METHOD GetDataValuesReq : BOOL
VAR_INPUT
    ipData     : I_AcsiCommonDataClass;
    eFc        : E_AcsiFc;
    hUser      : PVOID;
    ipSink     : I_ScsmGetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipData: Interface pointer of type I_AcsiCommonDataClass [▶ 364].

eFc: Functional group of type E_AcsiFc [▶ 487].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type I_ScsmGetDataValuesCnfEventSink [▶ 446] of an object with a user-defined implementation of the event handling routine: OnGetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.16 GetGoCBValuesReq



This method enables the service GetGOCBValues. With this service the client can read the values of all data objects of a GOOSE control block (GOOSE Control Block).

```
METHOD GetGoCBValuesReq : BOOL
VAR_INPUT
    ipGoCB    : I_AcsiCommonGooseControlBlockClass;
    hUser     : PVOID;
    ipSink    : I_ScsmGetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipGoCB: Interface pointer of type I_AcsiCommonGooseControlBlockClass [▶ 368].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type I_ScsmGetDataValuesCnfEventSink [▶ 446] of an object with a user-defined implementation of the event handling routine: OnGetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.17 GetLogicalDeviceDirectoryReq



This method activates the GetLogicalDeviceDirectory service. A client can use this service to query the server to determine which logical nodes are implemented in a particular logical device.

```
METHOD GetLogicalDeviceDirectoryReq : BOOL
VAR_INPUT
    ipLogicalDevice : I_AcsiCommonLogicalDeviceClass;
    hUser           : PVOID;
    ipSink          : I_ScsmGetLogicalDeviceDirectoryCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID       : UDINT;
    ipResult        : I_AsyncServiceResultClass;
END_VAR
```

ipLogicalDevice: Interface pointer of type I_AcsiCommonLogicalDeviceClass [▶ 369].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

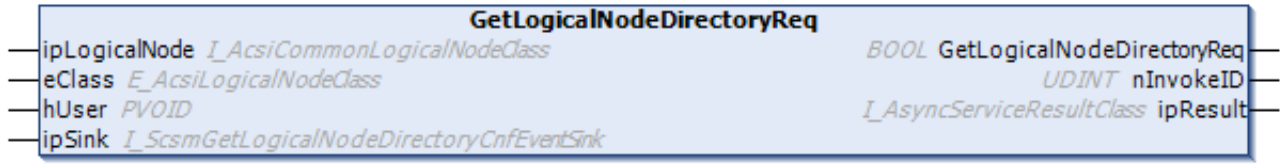
ipSink: Interface pointer of type I_ScsmGetLogicalDeviceDirectoryCnfEventSink [▶ 447] of an object with a user-defined implementation of the event handling routine OnGetLogicalDeviceDirectoryCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.18 GetLogicalNodeDirectoryReq



This method activates the GetLogicalNodeDirectory service. A client can use this service to query the server to determine which data objects, data groups and control blocks are implemented in a particular logical node.

```
METHOD GetLogicalNodeDirectoryReq : BOOL
VAR_INPUT
    ipLogicalNode : I_AcsiCommonLogicalNodeClass;
    eClass        : E_AcsiLogicalNodeClass;
    hUser         : PVOID;
    ipSink        : I_ScsmGetLogicalNodeDirectoryCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID     : UDINT;
    ipResult      : I_AsyncServiceResultClass;
END_VAR
```

ipLogicalNode: Interface pointer of type I_AcsiCommonLogicalNodeClass [▶ 370].

eClass: Object class of the logical node of type E_AcsiLogicalNodeClass [▶ 491].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

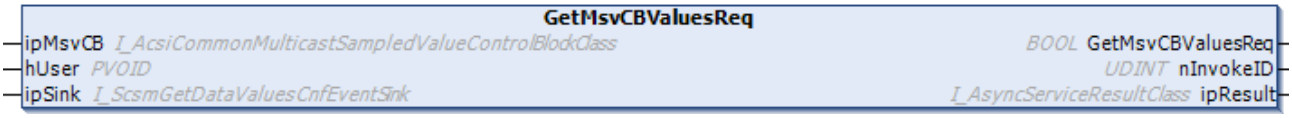
ipSink: Interface pointer of type I_ScsmGetLogicalNodeDirectoryCnfEventSink [▶ 447] of an object with a user-defined implementation of the event handling routine: OnGetLogicalNodeDirectoryCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.19 GetMsvCBValuesReq



This method enables the service: GetMSVCBValues. With this service the client can read the values of all data objects of an SV control block (MSV Control Block).

```
METHOD GetMsvCBValuesReq : BOOL
VAR_INPUT
    ipMsvCB  : I_AcsiCommonMulticastSampledValueControlBlockClass;
    hUser    : PVOID;
    ipSink   : I_ScsmGetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID: UDINT;
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipMsvCB: Interface pointer of type I_AcsiCommonMulticastSampledValueControlBlockClass [▶ 372].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type I_ScsmGetDataValuesCnfEventSink [▶ 446] of an object with a user-defined implementation of the event handling routine: OnGetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.20 GetServerDirectoryReq



This method activates the GetServerDirectoryReq service. A client can use this service to query which logical devices or files are implemented on a specific server.

```
METHOD GetServerDirectoryReq : BOOL
VAR_INPUT
    ipServer  : I_AcsiCommonIntelligentElectronicDeviceClass;
    eClass    : E_AcsiServerDirectoryClass;
    hUser     : PVOID;
    ipSink    : I_ScsmGetServerDirectoryCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipServer: Interface pointer of type I_AcsiCommonIntelligentElectronicDeviceClass [▶ 368].

eClass: Server class of type E_AcsiServerDirectoryClass [▶ 504]. This parameter specifies whether logical devices or files are to be queried.

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type I_ScsmGetServerDirectoryCnfEventSink [▶ 448] of an object with a user-defined implementation of the event handling routine OnGetServerDirectoryCnf. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.21 GetUrCBValuesReq



This method activates the GetURCBValues service.

```
METHOD GetURCBValuesReq : BOOL
VAR_INPUT
    ipUrcb    : I_AcsiCommonUnbufferedReportControlBlockClass;
    hUser     : PVOID;
    ipSink    : I_ScsmGetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipUrcb: Interface pointer of type I_AcsiCommonUnbufferedReportControlBlockClass [▶ 374].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

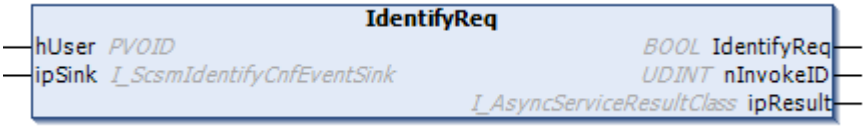
ipSink: Interface pointer of type I_ScsmGetDataValuesCnfEventSink [▶ 446] of an object with a user-defined implementation of the event handling routine: OnGetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.22 IdentifyReq



This method enables the service: MMS-Identify.

```
METHOD IdentifyReq : BOOL
VAR_INPUT
    hUser     : PVOID;
    ipSink    : I_ScsmIdentifyCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type: I_ScsmIdentifyCnfEventSink [▶ 449] from an object with a user-defined implementation of the event handling routine: OnIdentifyCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.23 ipIED

The instance of the communication function block is notified of the server data structure instance (top level) via this interface pointer. The communication function block can use this pointer for read or write access to the data structure.

```
PROPERTY ipIED : I_AcsiCommonIntelligentElectronicDeviceClass (SET)
```

ipIED: Interface pointer of type I_AcsiCommonIntelligentElectronicDeviceClass [▶ 368] of an object with the top-level implementation of the IEC 61850 data model.

6.5.3.3.24 ReleaseReq



This method activates the release service. A client can use this service to disconnect from the server in a controlled manner.

```
METHOD ReleaseReq : BOOL
VAR_INPUT
    ipSink   : I_ScsmReleaseCnfEventSink;
END_VAR
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipSink: Interface pointer of type I_ScsmReleaseCnfEventSink [▶ 450] of an object with a user-defined implementation of the event handling routine OnReleaseCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.25 ReleaseRsp



This method responds to the Release service. A client can use this service to disconnect from the server in a controlled manner.

```
METHOD ReleaseRsp : BOOL
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.26 SetAllDataValuesReq



This method activates the SetAllDataValues service.

```
METHOD SetAllDataValuesReq : BOOL
VAR_INPUT
    ipLogicalNode : I_AcsiCommonLogicalNodeClass;
    eFc           : E_AcsiFc;
    hUser         : PVOID;
    ipSink        : I_ScsmSetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID     : UDINT;
    ipResult      : I_AsyncServiceResultClass;
END_VAR
```

ipLogicalNode: Interface pointer of type I_AcsiCommonLogicalNodeClass [▶ 370].

eFc: Functional group of type E_AcsiFc [▶ 487].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type I_ScsmSetDataValuesCnfEventSink [▶ 452] of an object with a user-defined implementation of the event handling routine: OnSetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.27 SetBrCBValuesReq



This method activates the service: SetBRCBValues.

```
METHOD SetBRCBValuesReq : BOOL
VAR_INPUT
    ipBrcb    : I_AcsiCommonBufferedReportControlBlockClass;
    stSet     : ST_ScsmBrCBValuesSet;
    hUser     : PVOID;
    ipSink    : I_ScsmSetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipBrcb: Interface pointer of type I_AcsiCommonBufferedReportControlBlockClass [▶ 363].

stSet: Structured variable of type: ST_ScsmBrCBValuesSet [▶ 550]. This variable determines which RCB attribute values are to be written.

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type I_ScsmSetDataValuesCnfEventSink [▶ 452] of an object with a user-defined implementation of the event handling routine: OnSetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.28 SetDataSetValuesReq



This method activates the SetDataSetValues service.

```
METHOD SetDataSetValuesReq : BOOL
VAR_INPUT
    ipDataSet : I_AcsiCommonDataSetClass;
    hUser     : PVOID;
    ipSink    : I_ScsmSetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipDataSet: Interface pointer of type I_AcsiCommonDataSetClass [▶ 366].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

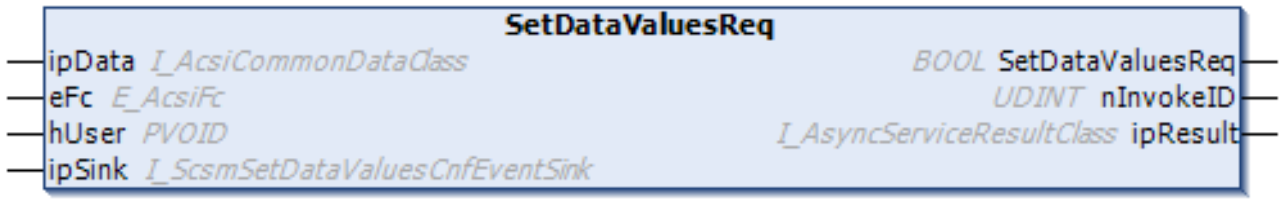
ipSink: Interface pointer of type I_ScsmSetDataValuesCnfEventSink [▶ 452] of an object with a user-defined implementation of the event handling routine: OnSetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.29 SetDataValuesReq



This method activates the SetDataValues service.

```
METHOD SetDataValuesReq : BOOL
VAR_INPUT
    ipData    : I_AcsiCommonDataClass;
    eFc       : E_AcsiFc;
    hUser     : PVOID;
    ipSink    : I_ScsmSetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipData: Interface pointer of type I_AcsiCommonDataClass [▶ 364].

eFc: Functional group of type E_AcsiFc [▶ 487].

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type I_ScsmSetDataValuesCnfEventSink [▶ 452] of an object with a user-defined implementation of the event handling routine: OnSetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.30 SetGoCBValuesReq



This method enables the service: SetGOCBValues. This method allows the client to write the values of a GOOSE Control Block.

```
METHOD SetGoCBValuesReq : BOOL
VAR_INPUT
    ipGoCB    : I_AcsiCommonGooseControlBlockClass;
    stSet     : ST_ScsmGoCBValuesSet;
    hUser     : PVOID;
    ipSink    : I_ScsmSetDataValuesCnfEventSink;
END_VAR
```

```
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipGoCB: Interface pointer of type I_AcsiCommonGooseControlBlockClass [▶ 368].

stSet: Structured variable of type: ST_ScsmGoCBValuesSet [▶ 552]. This variable determines which GOOSE control block attribute values should be written.

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type I_ScsmSetDataValuesCnfEventSink [▶ 452] of an object with a user-defined implementation of the event handling routine: OnSetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.31 SetMsvCBValuesReq



This method enables the service: SetMSVCBValues. This method allows the client to write the values of an SV control block (MSV Control Block).

```
METHOD SetMsvCBValuesReq : BOOL
VAR_INPUT
    ipMsvCB  : I_AcsiCommonMulticastSampledValueControlBlockClass;
    stSet    : ST_ScsmMsvCBValuesSet;
    hUser    : PVOID;
    ipSink   : I_ScsmSetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID: UDINT;
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipMsvCB: Interface pointer of type I_AcsiCommonMulticastSampledValueControlBlockClass [▶ 372].

stSet: Structured variable of type ST_ScsmMsvCBValuesSet [▶ 553]. This variable determines which MSVCB attribute values are to be written.

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

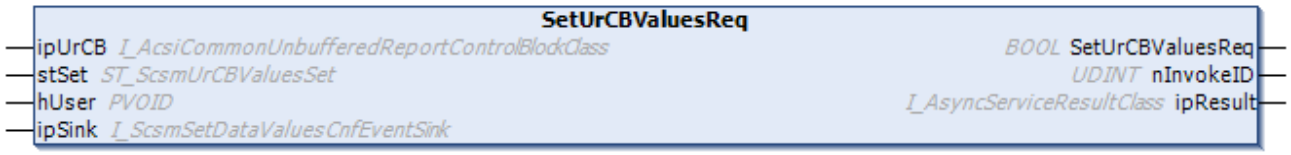
ipSink: Interface pointer of type I_ScsmSetDataValuesCnfEventSink [▶ 452] of an object with a user-defined implementation of the event handling routine: OnSetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.32 SetUrCBValuesReq



This method activates the SetURCBValues service.

```
METHOD SetURCBValuesReq : BOOL
VAR_INPUT
    ipUrcb    : I_AcsiCommonUnbufferedReportControlBlockClass;
    stSet     : ST_ScsmUrCBValuesSet;
    hUser     : PVOID;
    ipSink    : I_ScsmSetDataValuesCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipUrcb: Interface pointer of type I_AcsiCommonUnbufferedReportControlBlockClass [▶ 374].

stSet: Structured variable of type: ST_ScsmUrCBValuesSet [▶ 555]. This variable determines which RCB attribute values are to be written.

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

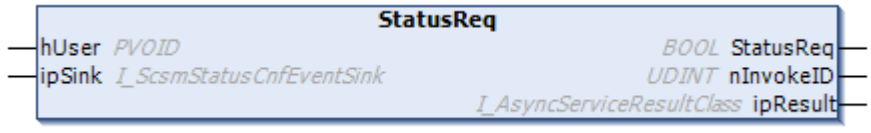
ipSink: Interface pointer of type I_ScsmSetDataValuesCnfEventSink [▶ 452] of an object with a user-defined implementation of the event handling routine: OnSetDataValuesCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.3.33 StatusReq



This method enables the service: MMS-Status.

```
METHOD StatusReq : BOOL
VAR_INPUT
    hUser     : PVOID;
    ipSink    : I_ScsmStatusCnfEventSink;
END_VAR
VAR_OUTPUT
    nInvokeID : UDINT;
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

hUser: Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.

ipSink: Interface pointer of type: I_ScsmStatusCnfEventSink [▶ 453] from an object with a user-defined implementation of the event handling routine: OnStatusCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

nInvokeID: Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.4 I_ScsmConnectionClass

Objects implementing this interface control the communication setup and data exchange between the TwinCAT PLC server application and the IEC 61850 client.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmConnectionClass

```
INTERFACE I_ScsmConnectionClass
```

Methods

Name	Description
Execute	This method triggers the execution of the communication layer.
AbortReq [▶ 435]	Enables the service: Abort.
AssociateRsp [▶ 436]	Responds to the service: Associate.
ReleaseReq [▶ 436]	Enables the service: Release.
ReleaseRsp [▶ 436]	Responds to the service: Release.

Properties

Name	Type	Access	Description
eLinkStatus	E_ScsmLinkStatus [▶ 525]	Get	Server Link status. Link status of the network adapter (accepted sockets).
eState	E_AsyncEnvironmentState [▶ 519]	Get	Environmental state of the communication connection.
ipAbortInd	I_ScsmAbortIndEventSink [▶ 438]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnAbortInd [▶ 438].
ipAssociateInd	I_ScsmAssociateIndEventSink [▶ 439]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnAssociateInd [▶ 439].
ipReleaseCnf	I_ScsmReleaseCnfEventSink [▶ 450]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnReleaseCnf [▶ 450].
ipReleaseInd	I_ScsmReleaseIndEventSink [▶ 451]	Set	Interface pointer of an object with a user-defined implementation of the event handling routine: OnReleaseInd [▶ 451].
ipServer	I_ScsmServerClass [▶ 437]	Get	Interface pointer of an object that identifies the server container.
ipSettings	I_ScsmConnectionSettingsClass [▶ 377]	Get	Protocol settings for the communication layer.
ipAssociation	I_ScsmAssociationClass [▶ 440]	Get	Interface pointer of an object that identifies the server connection (session).
sDesc	STRING	Get	The description of the server connection (session).

6.5.3.4.1 AbortReq



This method activates the Abort service. This service can be used to terminate the server connection (session) in an uncontrolled manner.

```
METHOD FINAL AbortReq : BOOL
VAR_INPUT
    ipReason : I_ServiceErrorClass;
    ipSink   : I_ScsmAbortIndEventSink;
END_VAR
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipReason: Interface pointer of type I_ServiceErrorClass [▶ 468]. This pointer can be used to pass an object with a user-defined cause for the disconnection.

ipSink: Interface pointer of type I_ScsmAbortIndEventSink [▶ 438] of an object with a user-defined implementation of the event handling routine: OnAbortInd (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.4.2 AssociateRsp



This method responds to the Associate service. With this service, a server can establish a server connection (session) to the client.

```
METHOD FINAL AssociateRsp : BOOL
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.4.3 ReleaseReq



This method activates the Release service. With this service, a server can terminate the server connection (session) to the client in a controlled manner.

```
METHOD FINAL ReleaseReq : BOOL
VAR_INPUT
    ipSink   : I_ScsmReleaseCnfEventSink;
END_VAR
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipSink: Interface pointer of type I_ScsmReleaseCnfEventSink [▶ 450] of an object with a user-defined implementation of the event handling routine OnReleaseCnf (service feedback). This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.4.4 ReleaseRsp



This method responds to the Release service. With this service, a server can terminate a server connection (session) to the client in a controlled manner.

```
METHOD FINAL ReleaseRsp : BOOL
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.5 I_ScsmServerClass

Objects implementing this interface control the communication setup and data exchange between the TwinCAT PLC server application and the IEC 61850 client.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmServerClass

```
INTERFACE I_ScsmServerClass
```

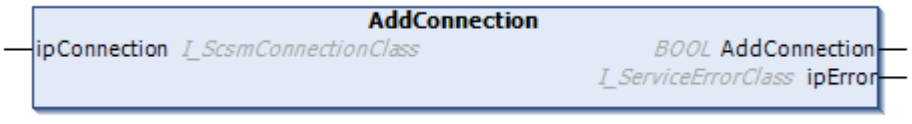
Methods

Name	Description
AddConnection [▶ 437]	Adds a connection instance to the Server.
Execute	This method triggers the execution of the communication layer.
RemoveConnection [▶ 438]	Removes a connection instance from the server.

Properties

Name	Type	Access	Description
eLinkStatus	E_ScsmLinkStatus [▶ 525]	Get	Server Link status. Link status of the network adapter (listener sockets).
ipConnections	I_ScsmConnectionClass [▶ 434]	Get	Interface pointer from an object with the implementation of the first connection instance.
ipIED	I_AcsiCommonIntelligentElectronicDeviceClass [▶ 157]	Get,Set	Interface pointer of an object with the implementation of the top-level of the IEC 61850 data model.
ipSettings	I_ScsmServerSettingsClass [▶ 437]	Get	Protocol settings for the communication layer.
nItem	UDINT	Get	The number of connection instances.
sDesc	STRING	Get,Set	The description of the server (container).

6.5.3.5.1 AddConnection



This method adds a connection instance to the server.

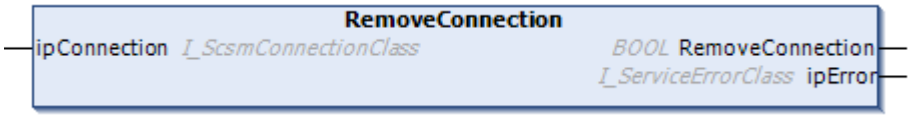
```
METHOD FINAL AddConnection : BOOL
VAR_INPUT
    ipConnection : I_ScsmConnectionClass;
END_VAR
VAR_OUTPUT
    ipError      : I_ServiceErrorClass
END_VAR
```

ipConnection: Interface pointer of type: I_ScsmConnectionClass [▶ 434] of the connection instance that is to be removed from the server.

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468].

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.5.2 RemoveConnection



This method removes a connection instance from the server.

```
METHOD FINAL RemoveConnection : BOOL
VAR_INPUT
    ipConnection : I_ScsmConnectionClass;
END_VAR
VAR_OUTPUT
    ipError      : I_ServiceErrorClass
END_VAR
```

ipConnection: Interface pointer of type: I_ScsmConnectionClass [▶ 434] of the connection instance that is to be removed from the server.

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468].

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.6 I_ScsmAbortIndEventSink

Objects that implement this interface have a user-defined Abort service event handling routine for disconnecting.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmAbortIndEventSink

```
INTERFACE I_ScsmAbortIndEventSink
```

Methods for event handling

Name	Description
OnAbortInd [▶ 438]	User-defined abort service event handling routine

6.5.3.6.1 OnAbortInd

```
METHOD OnAbortInd
VAR_INPUT
    ipAA     : I_ScsmAssociationClass;
    bPAbort  : BOOL;
    ipReason : I_ServiceErrorClass;
    ipData   : I_BaseStreamClass;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

bPAbort: Trigger for the abort indication service primitive. Triggered by the service provider if TRUE, triggered by the service user if FALSE (application has started the abort request service primitive).

ipReason: Interface pointer of type I_ServiceErrorClass [▶ 505]. This pointer can be used to obtain more detailed information about the cause of the disconnection.

ipData: Interface pointer of type I_BaseStreamClass [▶ 471]. This pointer can be used to access optional data. No data is available if the pointer value is 0.

6.5.3.7 I_ScsmAssociateCnfEventSink

Objects that implement this interface have a user-defined Associate service event handling routine for connection establishment.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmAssociateCnfEventSink

```
INTERFACE I_ScsmAssociateCnfEventSink
```

Methods for event handling

Name	Description
OnAssociateCnf [▶ 439]	User-defined associate service event handling routine

6.5.3.7.1 OnAssociateCnf

```
METHOD OnAssociateCnf
VAR_INPUT
    ipAA   : I_ScsmAssociationClass;
    eError : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.8 I_ScsmAssociateIndEventSink

Objects that implement this interface have a user-defined Associate service event handling routine for server session connection establishment.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmAssociateIndEventSink

```
INTERFACE I_ScsmAssociateIndEventSink
```

Methods for event handling

Name	Description
OnAssociateInd [▶ 439]	User-defined Associate service event handling routine

6.5.3.8.1 OnAssociateInd

```
METHOD OnAssociateInd : BOOL
VAR_INPUT
    ipAA     : I_ScsmAssociationClass;
END_VAR
```

```
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463]. This pointer can be used to query and monitor the status/progress and the result of the service execution.

Sample (extract)

```
METHOD OnAssociateInd : BOOL
VAR_INPUT
    ipAA     : I_ScsmAssociationClass;
END_VAR
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
VAR
END_VAR
```

```
(*…*)
sLastAbortReason:= '';
sLastErrorResult:= '';
(*…*)

OnAssociateInd:= fbConnection.AssociateRsp(ipResult=>ipResult);
```

6.5.3.9 I_ScsmAssociationClass.

An interface pointer of this type identifies a client-server connection.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmAssociationClass.

```
INTERFACE I_ScsmAssociationClass
```

Methods

-

Properties

Name	Type	Access	Description
ipAuthentication	I_ScsmAuthenticationClass [▶ 440]	Get	Interface pointer to the authentication object.

6.5.3.10 I_ScsmAuthenticationClass

Objects that implement this interface are used for user-defined negotiation of the ACSE (Association Control Service Element) authentication and the MMS (Manufacturing Message Specification) communication parameters.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_MmsAuthenticationClass -> I_ScsmAuthenticationClass

```
INTERFACE I_ScsmAuthenticationClass EXTENDS I_MmsAuthenticationClass
```

Methods

Name	Description
Accept [▶ 441]	Grants authentication access.
Reject [▶ 442]	Denies authentication access.
Verify [▶ 443]	Verifies the authentication access.
GetVerifyResult [▶ 442]	Returns the verification result of the authentication.

Properties

Name	Type	Access	Description
bAccepted	BOOL	Get	Verification result of the authentication. TRUE => Connection is accepted, FALSE => Connection is rejected.
sPassword	STRING	Get	Authentication password (sent password at the client and received password at the server).
ipNegotiatedAdditionalCBBSupported	I_MmsAdditionalCBBOptions [▶ 381]	Get	Interface pointer for configuring the negotiated, additionally supported CBB parameters of the MMS communication layer.
ipNegotiatedAdditionalSupported	I_MmsAdditionalSupportOptions [▶ 381]	Get	Interface pointer for configuring the negotiated, additionally supported MMS services.
ipNegotiatedParameterCBB	I_MmsParameterSupportOptions [▶ 380]	Get	Interface pointer for configuring the negotiated supported options of the MMS communication layer.
ipNegotiatedServicesSupported	I_MmsServiceSupportOptions [▶ 380]	Get	Interface pointer for configuring the negotiated supported services of the MMS communication layer.
nNegotiatedLocalDetail	T_MmsInteger32 [▶ 557]	Get	Negotiated MMS parameter: Local detail (max. size of the MMS data).
nNegotiatedMaxServOutCalled	T_MmsInteger16 [▶ 557]	Get	Negotiated MMS parameter: Max. number of incoming services.
nNegotiatedMaxServOutCalling	T_MmsInteger16 [▶ 557]	Get	Negotiated MMS parameter: M ax. Number of outgoing services.
nNegotiatedNestingLevel	T_MmsInteger8 [▶ 557]	Get	Negotiated MMS parameter: Nesting depth.

6.5.3.10.1 Accept

This method grants authentication access.

```
METHOD Accept : BOOL
VAR_INPUT
    sRespondingPass : STRING;
END_VAR
```

sRespondingPass: Password to be sent as a response in mutual authentication mode. In unilateral authentication mode, this parameter value has no meaning.

Return parameter: TRUE => Success. FALSE => Error (e.g. command not permitted or not permitted in the current state).

6.5.3.10.2 GetVerifyResult

This method provides a detailed result of the verification of the authentication data. The internal standard verification takes place on the server when the associate request data is received (AARQ telegram) and on the client when the associate response data is received (AARE telegram). With the GetVerifyResult return parameter value TRUE, the IEC 61850 server or client application may check or change the authentication status itself at a certain point in time, but only if authentication (authentication-functional-unit) has been activated in the communication requests of both communication partners.

A connection that is expected to be accepted by the server can be subsequently rejected using the Reject [▶ 442] method. A connection that is likely to be rejected by the server can be subsequently accepted using the Accept [▶ 441] method. In a server application, the methods: Accept [▶ 441] or Reject [▶ 442] may only be called within the implementation of the OnAssociateInd event handling routine. Only at this point can the application still control the authentication process itself.

In mutual authentication mode, a client application can also check the authentication status itself at a specific time. A connection that is expected to be accepted by the client can be rejected using the Reject [▶ 442] method. A connection that is likely to be rejected by the client can be subsequently accepted using the Accept [▶ 441] method. In a client application, the Accept [▶ 441] or Reject [▶ 442] methods may only be called within the implementation of the OnAssociateCnf event handling routine. A connection that has already been rejected by the server cannot be subsequently accepted by the client.

```
METHOD GetVerifyResult : BOOL
VAR_OUTPUT
    bAccepted       : BOOL;
    eType           : E_AcseAuthentication;
    sMechanismName  : STRING;
    sPassword       : STRING;
    sRespondingPass : STRING;
    eResult         : E_ACSE_AareResult;
    eDiagnostic     : E_ACSE_AareDiagnostic;
    eAbrtDiagnostic : E_ACSE_AbrtDiagnostic;
END_VAR
```

bAccepted: General result of the verification of the authentication data. TRUE => Connection is accepted and maintained, FALSE => Connection is rejected and closed. This parameter will also be TRUE if no authentication has been configured/activated for both communication partners.

eType: The used/configured authentication mode of type E_AcseAuthentication [▶ 474]. Unilateral (standard) or mutual.

sMechanismName: Name of the authentication mechanism. Client => sent name, server => received name (default: '2.2.3.1').

sPassword: Authentication password in plain text. Client => sent password, server => received password.

sRespondingPass: Password in plain text, sent in response to the mutual authentication type.

eResult: Result parameter in the Association Response command of type E_ACSE_AareResult [▶ 472]. Result parameter which the server will send as a response to the client, or which the client has already received as a response from the server (in the ACSE-AARE telegram).

eDiagnostic: Diagnostic data for the Association Response command of type E_ACSE_AareDiagnostic [▶ 471]. Additional information that the server will send as a response to the client or that the client has already received as a response from the server (in the ACSE-AARE telegram).

eAbrtDiagnostic: Diagnostic data of type E_ACSE_AbrtDiagnostic [▶ 473], which is sent in the Association Abort command in the event of a rejected connection.

Return parameter: TRUE => Success, the client/server application is allowed to check or change the authentication status. FALSE => Error (e.g. command not permitted or not permitted in the current state).

6.5.3.10.3 Reject

This method denies authentication access.

```
METHOD Reject : BOOL
```

Return parameter: TRUE => Success. FALSE => Error (e.g. command not permitted or not permitted in the current state).

6.5.3.10.4 Verify

This method performs the internal verification of the authentication parameters again.

```
METHOD Verify : BOOL
```

Return parameter: TRUE => Success. FALSE => Error (e.g. command not permitted or not permitted in the current state).

6.5.3.11 I_ScsmCommandTerminationIndEventSink

Objects that implement this interface have a user-defined Command termination service event handling routine for switch control.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmCommandTerminationIndEventSink

```
INTERFACE I_ScsmCommandTerminationIndEventSink
```

Methods for event handling

Name	Description
OnCommandTerminationInd [▶ 443]	User-defined Command termination service event handling routine

6.5.3.11.1 OnCommandTerminationInd

```
METHOD OnCommandTerminationInd
VAR_INPUT
    ipAA   : I_ScsmAssociationClass;
    info   : ST_ScsmCommandTerminationEventInfo;
    ipCtrl : I_AcsiCommonControllableDataClass;
    ipData : I_AcsiCommonDataClass;
    eFc    : E_AcsiFc;
    eError : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmCommandTerminationEventInfo [▶ 551] with additional information.

ipCtrl: Interface pointer of type: I_AcsiCommonControllableDataClass [▶ 364] of a controllable object.

ipData: Interface pointer of type I_AcsiCommonDataClass [▶ 364].

eFc: Functional group as enumeration type: E_AcsiFc [▶ 487].

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.12 I_ScsmControlCnfEventSink

Objects that implement this interface have a user-defined Control service event handling routine for switch control (Cancel, Operate, Select, SelectWithValue).

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmControlCnfEventSink

```
INTERFACE I_ScsmControlCnfEventSink
```

Methods for event handling

Name	Description
OnControlCnf [▶ 444]	User-defined control service Event handling routine for switch control.

6.5.3.12.1 OnControlCnf

```
METHOD OnControlCnf
VAR_INPUT
    ipAA    : I_ScsmAssociationClass;
    info    : ST_ScsmConfirmedEventInfo;
    ipCtrl  : I_AcsiCommonControllableDataClass;
    ipData  : I_AcsiCommonDataClass;
    eFc     : E_AcsiFc;
    eError  : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmConfirmedEventInfo [▶ 551] with additional information.

ipCtrl: Interface pointer of type: I_AcsiCommonControllableDataClass [▶ 364] of a controllable object.

ipData: Interface pointer of type I_AcsiCommonDataClass [▶ 364].

eFc: Functional group as enumeration type: E_AcsiFc [▶ 487].

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.13 I_ScsmCreateDataSetCnfEventSink

Objects that implement this interface have a user-defined CreateDataSet service event handling routine.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmCreateDataSetCnfEventSink

```
INTERFACE I_ScsmCreateDataSetCnfEventSink
```

Methods for event handling

Name	Description
OnCreateDataSetCnf [▶ 444]	User-defined CreateDataSet service event handling routine.

6.5.3.13.1 OnCreateDataSetCnf

```
METHOD OnCreateDataSetCnf
VAR_INPUT
    ipAA      : I_ScsmAssociationClass;
    info      : ST_ScsmConfirmedEventInfo;
    ipDataSet : I_AcsiCommonDataSetClass;
    eError    : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmConfirmedEventInfo [▶ 551] with additional information.

ipDataSet: Interface pointer of type: I_AcsiCommonDataSetClass [▶ 366]

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.14 I_ScsmDeleteDataSetCnfEventSink

Objects that implement this interface have a user-defined DeleteDataSet service event handling routine.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmDeleteDataSetCnfEventSink

```
INTERFACE I_ScsmDeleteDataSetCnfEventSink
```

Methods for event handling

Name	Description
OnDeleteDataSetCnf [▶ 445]	User-defined DeleteDataSet service event handling routine.

6.5.3.14.1 OnDeleteDataSetCnf

```
METHOD OnDeleteDataSetCnf
VAR_INPUT
    ipAA      : I_ScsmAssociationClass;
    info      : ST_ScsmConfirmedEventInfo;
    ipDataSet : I_AcsiCommonDataSetClass;
    eError    : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmConfirmedEventInfo [▶ 551] with additional information.

ipDataSet: Interface pointer of type: I_AcsiCommonDataSetClass [▶ 366]

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.15 I_ScsmGetDataSetDirectoryCnfEventSink

Objects that implement this interface have a user-defined GetDataSetDirectory service event handling routine.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmGetDataSetDirectoryCnfEventSink

```
INTERFACE I_ScsmGetDataSetDirectoryCnfEventSink
```

Methods for event handling

Name	Description
OnGetDataSetDirectoryCnf [▶ 446]	User-defined GetDataSetDirectory service event handling routine

6.5.3.15.1 OnGetDataSetDirectoryCnf

```
METHOD OnGetDataSetDirectoryCnf
VAR_INPUT
    ipAA             : I_ScsmAssociationClass;
    info             : ST_ScsmConfirmedEventInfo;
    ipDataSet        : I_AcsiCommonDataSetClass;
    sObjectReference : T_AcsiObjectReference;
    eFc              : E_AcsiFc;
    eError           : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmConfirmedEventInfo [▶ 551] with additional information.

ipDataSet: Interface pointer of type: I_AcsiCommonDataSetClass [▶ 366]

sObjectReference:

eFc: Functional group as enumeration type: E_AcsiFc [▶ 487].

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.16 I_ScsmGetDataValuesCnfEventSink

Objects that implement this interface have a user-defined GetDataValues service event handling routine.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmGetDataValuesCnfEventSink

```
INTERFACE I_ScsmGetDataValuesCnfEventSink
```

Methods for event handling

Name	Description
OnGetDataValuesCnf [▶ 446]	User-defined GetDataValues service event handling routine

6.5.3.16.1 OnGetDataValuesCnf

```
METHOD OnGetDataValuesCnf
VAR_INPUT
    ipAA   : I_ScsmAssociationClass;
    info   : ST_ScsmConfirmedEventInfo;
    ipData : I_AcsiCommonDataClass;
    eFc    : E_AcsiFc;
    eError : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmConfirmedEventInfo [▶ 551] with additional information.

ipData: Interface pointer of type: I_AcsiCommonDataClass [▶ 364].

eFc: Functional group as enumeration type: E_AcsiFc [▶ 487].

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.17 I_ScsmGetLogicalDeviceDirectoryCnfEventSink

Objects that implement this interface have a user-defined GetLogicalDeviceDirectory service event handling routine.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmGetLogicalDeviceDirectoryCnfEventSink

```
INTERFACE I_ScsmGetLogicalDeviceDirectoryCnfEventSink
```

Methods for event handling

Name	Description
OnGetLogicalDeviceDirectoryCnf [▶ 447]	User-defined GetLogicalDeviceDirectory service event handling routine

6.5.3.17.1 OnGetLogicalDeviceDirectoryCnf

```
METHOD OnGetLogicalDeviceDirectoryCnf
VAR_INPUT
    ipAA             : I_ScsmAssociationClass;
    info             : ST_ScsmConfirmedEventInfo;
    ipLogicalDevice  : I_AcsiCommonLogicalDeviceClass;
    sObjectReference : T_AcsiObjectReference;
    eError           : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmConfirmedEventInfo [▶ 551] with additional information.

ipLogicalDevice: Interface pointer of type I_AcsiCommonLogicalDeviceClass [▶ 369].

sObjectReference:

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.18 I_ScsmGetLogicalNodeDirectoryCnfEventSink

Objects that implement this interface have a user-defined GetLogicalNodeDirectory service event handling routine.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmGetLogicalNodeDirectoryCnfEventSink

```
INTERFACE I_ScsmGetLogicalNodeDirectoryCnfEventSink
```

Methods for event handling

Name	Description
OnGetLogicalNodeDirectoryCnf [▶ 448]	User-defined GetLogicalNodeDirectory service event handling routine

6.5.3.18.1 OnGetLogicalNodeDirectoryCnf

```
METHOD OnGetLogicalNodeDirectoryCnf
VAR_INPUT
    ipAA             : I_ScsmAssociationClass;
    info             : ST_ScsmConfirmedEventInfo;
    ipLogicalNode    : I_AcsiCommonLogicalNodeClass;
    eClass           : E_AcsiLogicalNodeClass;
    sObjectReference : T_AcsiObjectReference;
    eError           : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmConfirmedEventInfo [▶ 551] with additional information.

ipLogicalNode: Interface pointer of type I_AcsiCommonLogicalNodeClass [▶ 370].

eClass: Enumeration variable of type E_AcsiLogicalNodeClass [▶ 491].

sObjectReference:

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.19 I_ScsmGetServerDirectoryCnfEventSink

Objects that implement this interface have a user-defined GetServerDirectory service event handling routine.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmGetServerDirectoryCnfEventSink

```
INTERFACE I_ScsmGetServerDirectoryCnfEventSink
```

Methods for event handling

Name	Description
OnGetServerDirectoryCnf [▶ 448]	User-defined GetServerDirectory service event handling routine

6.5.3.19.1 OnGetServerDirectoryCnf

```
METHOD OnGetServerDirectoryCnf
VAR_INPUT
    ipAA             : I_ScsmAssociationClass;
    info             : ST_ScsmConfirmedEventInfo;
    ipServer         : I_AcsiCommonIntelligentElectronicDeviceClass;
    eClass           : E_AcsiServerDirectoryClass;
    sObjectReference : T_AcsiObjectReference;
    eError           : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmConfirmedEventInfo [▶ 551] with additional information.

ipServer: Interface pointer of type I_AcsiCommonIntelligentElectronicDeviceClass [▶ 368] of the server object that is queried.

eClass: Enumeration variable of type E_AcsiServerDirectoryClass [▶ 504]. This parameter specifies which information is queried: Logical nodes or files.

sObjectReference:

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.20 I_ScsmIdentifyCnfEventSink

Objects that implement this interface have a user-defined identify service event handling routine.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmIdentifyCnfEventSink

```
INTERFACE I_ScsmIdentifyCnfEventSink
```

Methods for event handling

Name	Description
OnIdentifyCnf [▶ 449]	User-defined identify service event handling routine.

6.5.3.20.1 OnIdentifyCnf

```
METHOD OnIdentifyCnf
VAR_INPUT
    ipAA        : I_ScsmAssociationClass;
    info        : ST_ScsmConfirmedEventInfo;
    sVendorName : T_MmsString;
    sModelName  : T_MmsString;
    sRevision   : T_MmsString;
    eError      : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmConfirmedEventInfo [▶ 551] with additional information.

sVendorName: Manufacturer designation of type: T_MmsString [▶ 558].

sModelName: Model name of type: T_MmsString [▶ 558].

sRevision: Revision string of type: T_MmsString [▶ 558].

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.21 I_ScsmLastApplErrorIndEventSink

Objects that implement this interface have a user-defined LastApplError service event handling routine for access to additional diagnostic report data.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmLastApplErrorIndEventSink

```
INTERFACE I_ScsmLastApplErrorIndEventSink
```

Methods for event handling

Name	Description
OnLastApplErrorInd [▶ 450]	User-defined LastApplError service event handling routine

6.5.3.21.1 OnLastApplErrorInd

```
METHOD OnLastApplErrorInd
VAR_INPUT
    ipAA   : I_ScsmAssociationClass;
    info   : ST_ScsmAdditionalCauseDiagnosticEventInfo;
    ipCtrl : I_AcsiCommonControllableDataClass;
    eError : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmAdditionalCauseDiagnosticEventInfo [▶ 549] with additional information about LastApplError.

ipCtrl: Interface pointer of type: I_AcsiCommonControllableDataClass [▶ 364] of a controllable object.

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.22 I_ScsmReleaseCnfEventSink

Objects that implement this interface have a user-defined release service event handling routine for disconnection.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmReleaseCnfEventSink

```
INTERFACE I_ScsmReleaseCnfEventSink
```

Methods for event handling

Name	Description
OnReleaseCnf [▶ 450]	User-defined release service event handling routine

6.5.3.22.1 OnReleaseCnf

```
METHOD OnReleaseCnf
VAR_INPUT
    ipAA   : I_ScsmAssociationClass;
    eError : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.23 I_ScsmReleaseIndEventSink

Objects that implement this interface have a user-defined release service event handling routine for disconnection.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmReleaseIndEventSink

```
INTERFACE I_ScsmReleaseIndEventSink
```

Methods for event handling

Name	Description
OnReleaseInd [▶ 451]	User-defined release service event handling routine

6.5.3.23.1 OnReleaseInd

```
METHOD OnReleaseInd : E_EventCompletion
VAR_INPUT
    ipAA    : I_ScsmAssociationClass;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

6.5.3.24 I_ScsmReportIndEventSink

Objects that implement this interface have a user-defined report service event handling routine.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmReportIndEventSink

```
INTERFACE I_ScsmReportIndEventSink
```

Methods for event handling

Name	Description
OnBufferedReportInd [▶ 451]	Custom buffered report service event handling routine
OnUnbufferedReportInd [▶ 452]	Custom unbuffered report service event handling routine

6.5.3.24.1 OnBufferedReportInd

```
METHOD OnBufferedReportInd
VAR_INPUT
    ipAA      : I_ScsmAssociationClass;
    info      : ST_ScsmReportEventInfo;
```

```
    ipBrcb    : I_AcsiCommonBufferedReportControlBlockClass;
    ipDataSet : I_AcsiCommonDataSetClass;
    ipData    : I_AcsiCommonDataClass;
    eFc       : E_AcsiFc;
    eError    : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmReportEventInfo [▶ 553] with additional information on the report.

ipBrcb: Interface pointer of type I_AcsiCommonBufferedReportControlBlockClass [▶ 363].

ipDataSet: Interface pointer of type: I_AcsiCommonDataSetClass [▶ 366]

ipData: Interface pointer of type: I_AcsiCommonDataClass [▶ 364].

eFc: Functional group as enumeration type: E_AcsiFc [▶ 487].

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.24.2 OnUnbufferedReportInd

```
METHOD OnUnbufferedReportInd
VAR_INPUT
    ipAA      : I_ScsmAssociationClass;
    info      : ST_ScsmReportEventInfo;
    ipUrcb    : I_AcsiCommonUnbufferedReportControlBlockClass;
    ipDataSet : I_AcsiCommonDataSetClass;
    ipData    : I_AcsiCommonDataClass;
    eFc       : E_AcsiFc;
    eError    : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmReportEventInfo [▶ 553] with additional information on the report.

ipUrcb: Interface pointer of type I_AcsiCommonUnbufferedReportControlBlockClass [▶ 374].

ipDataSet: Interface pointer of type: I_AcsiCommonDataSetClass [▶ 366]

ipData: Interface pointer of type: I_AcsiCommonDataClass [▶ 364].

eFc: Functional group as enumeration type: E_AcsiFc [▶ 487].

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.25 I_ScsmSetDataValuesCnfEventSink

Objects that implement this interface have a user-defined SetDataValues service event handling routine.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmSetDataValuesCnfEventSink

```
INTERFACE I_ScsmSetDataValuesCnfEventSink
```

Methods for event handling

Name	Description
OnSetDataValuesCnf [▶ 453]	User-defined SetDataValues service event handling routine

6.5.3.25.1 OnSetDataValuesCnf

```
METHOD OnSetDataValuesCnf
VAR_INPUT
    ipAA   : I_ScsmAssociationClass;
    info   : ST_ScsmConfirmedEventInfo;
    ipData : I_AcsiCommonDataClass;
    eFc    : E_AcsiFc;
    eError : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmConfirmedEventInfo [▶ 551] with additional information.

ipData: Interface pointer of type: I_AcsiCommonDataClass [▶ 364].

eFc: Functional group as enumeration type: E_AcsiFc [▶ 487].

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.26 I_ScsmStatusCnfEventSink

Objects that implement this interface have a user-defined status service event handling routine.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmStatusCnfEventSink

```
INTERFACE I_ScsmStatusCnfEventSink
```

Methods for event handling

Name	Description
OnStatusCnf [▶ 453]	User-defined status service event handling routine.

6.5.3.26.1 OnStatusCnf

```
METHOD OnStatusCnf
VAR_INPUT
    ipAA               : I_ScsmAssociationClass;
    info               : ST_ScsmConfirmedEventInfo;
    eVmdLogicalStatus  : E_MmsVmdLogicalStatus;
    eVmdPhysicalStatus : E_MmsVmdPhysicalStatus;
    ipLocalDetail      : I_MmsBitString;
    eError             : E_AcsiServiceError;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

info: Structured variable of type ST_ScsmConfirmedEventInfo [▶ 551] with additional information.

eVmdLogicalStatus: Enumeration type: E_MmsVmdLogicalStatus [▶ 524].

eVmdPhysicalStatus: Enumeration type: E_MmsVmdPhysicalStatus [▶ 524].

ipLocalDetail: Interface pointer of type: I_MmsBitString.

eError: Enumeration type: E_AcsiServiceError [▶ 505].

6.5.3.27 I_ScsmSystemClockEventSink

Objects that implement this interface have a user-defined system clock service event handling routine for access to an external time source.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_ScsmSystemClockEventSink

```
INTERFACE I_ScsmSystemClockEventSink
```

Methods for event handling

Name	Description
OnGetSystemTime [▶ 454]	User-defined system clock service event handling routine

6.5.3.27.1 OnGetSystemTime

```
METHOD OnGetSystemTime : BOOL
VAR_INPUT
    ipAA    : I_ScsmAssociationClass;
END_VAR
VAR_OUTPUT
    utcTime : T_UtcTime;
END_VAR
```

ipAA: Interface pointer of type: I_ScsmAssociationClass [▶ 440]. If several connection instances are active, this pointer can be used to determine the object instance running the service. This parameter is optional and reserved for future use. The pointer value 0 defines the parameter as optional.

utcTime: Current UTC time information of type T_UtcTime [▶ 560].

6.5.3.28 Control block access

6.5.3.28.1 I_ScsmBrCBImplClass

Objects that implement this interface are report control block instances for buffered reports (buffered report control blocks, BRCBs). The TwinCAT Telecontrol Configurator generates report control block instances that are derived/extended from FB_AcsiBufferedReportControlBlock [▶ 170]. All function blocks derived from FB_AcsiBufferedReportControlBlock [▶ 170] automatically have the interface referred to above implemented by default.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AcsiCommonControlBlockClass [▶ 363] -> I_AcsiCommonBufferedReportControlBlockClass [▶ 363] -> I_ScsmBrCBImplClass

```
INTERFACE I_ScsmBrCBImplClass EXTENDS I_AcsiCommonBufferedReportControlBlockClass
```

Properties

Name	Type	Access	Description
bGI	BOOL	Get, Set	Attribute value: Enabling the general interrogation command.
bPurgeBuf	BOOL	Get, Set	Attribute value: Enabling the purge command.
bRptEna	BOOL	Get, Set	Attribute value: Enabling/disabling the RCB control block. Further configuration needed.
cOptFlds	ST_AcsiOptionalFields [▶ 536]	Get, Set	Attribute value: Configuration of the optional fields that are transferred with a report.
cTrgOps	ST_AcsiTriggerConditions [▶ 544]	Get, Set	Attribute value: Configuration of the trigger options for sending a report.
iResvTms	INT	Get, Set	Attribute value: Configuration of the time for reserving the RCB for a specific client.
nBufTm	DWORD	Get, Set	Attribute value: Configuration of the max. report buffer time in milliseconds.
nConfRev	DWORD	Get, Set	Attribute value: Revision number of the RCB control block configuration.
nEntryID	LWORD	Get, Set	Attribute value: ID of the report entry as 64-bit numerical value.
nIntgPd	DWORD	Get, Set	Attribute value: Configuration of the max. time between integrity reports in milliseconds.
nSqNum	WORD	Get, Set	Attribute value: Report sequence number.
oEntryID	T_OCTET8 [▶ 558]	Get, Set	Attribute value: ID of the report entry as octet string.
oOwner	T_OCTET64 [▶ 559]	Get, Set	Attribute value: Owner of the RCB. IP address of the client which has reserved the RCB for itself (offline/online reservation).
sDatSet	T_AcsiObjectReference [▶ 555]	Get, Set	Attribute value: Name of the linked data set.
sRptID	T_AcsiVisString129 [▶ 556]	Get, Set	Attribute value: Report identification string.
tTimeOfEntry	T_BinaryTime [▶ 556]	Get, Set	Attribute value: Report timestamp.
ipDatSet	I_AcsiCommonDataSetClass [▶ 366]	Get	Interface pointer of the linked data set.
ipClient	I_ScsmBrCBClientJobClass [▶ 455]	Get	Interface pointer of the client implementation of the BRCB control block.
ipServer	I_ScsmBrCBServerJobClass [▶ 457]	Get	Interface pointer of the server implementation of the BRCB control block.

6.5.3.28.2 I_ScsmBrCBClientJobClass

Objects implementing this interface control the behavior of the client BRCB control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Methods

Name	Description
EnableReq [▶ 456]	Enables the control block.
DisableReq [▶ 456]	Disables the control block.
GIreq [▶ 456]	Enables the general interrogation command.
PurgeBufReq [▶ 457]	Enables the PurgeBuf command.
ResyncReq [▶ 457]	Enables the Resync command.

6.5.3.28.2.1 EnableReq

Enables the BRCB control block.

```
METHOD FINAL EnableReq : BOOL
VAR_INPUT
    ipClient  : I_ScsmClientClass;
    ipDataSet : I_AcsiCommonDataSetClass;
END_VAR
VAR_OUTPUT
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipClient: Interface pointer of type: I_ScsmClientClass [▶ 409]. This pointer determines the service executing object instance of the IEC 61850 client.

ipDataSet: Interface pointer of type I_AcsiCommonDataSetClass [▶ 366].

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.28.2.2 DisableReq

Disables the BRCB control block.

```
METHOD FINAL DisableReq : BOOL
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.28.2.3 GIreq

Enables the general interrogation command.

```
METHOD FINAL GIreq : BOOL
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.28.2.4 PurgeBufReq

Enables the Purge command.

```
METHOD FINAL PurgeBufReq : BOOL
VAR_INPUT
    ipClient : I_ScsmClientClass;
END_VAR
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipClient: Interface pointer of type: I_ScsmClientClass [▶ 409]. This pointer determines the service executing object instance of the IEC 61850 client.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.28.2.5 ResyncReq

Enables the Resync command.

```
METHOD FINAL ResyncReq : BOOL
VAR_INPUT
    ipClient : I_ScsmClientClass;
    nEntryID : LWORD;
END_VAR
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipClient: Interface pointer of type: I_ScsmClientClass [▶ 409]. This pointer determines the service executing object instance of the IEC 61850 client.

nEntryID: EntryID of the report for the buffer resynchronization command.

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.28.3 I_ScsmBrCBServerJobClass

Objects implementing this interface control the behavior of the server BRCB control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Methods

Name	Description
Execute [▶ 457]	Runs the RCB state machine of the server.

6.5.3.28.3.1 Execute

Runs the RCB state machine of the server.

```
METHOD Execute : BOOL
```

6.5.3.28.4 I_ScsmGoCBImplClass

Objects implementing this interface are control block instances for Generic Object Oriented Substation Events (GOOSE). The TwinCAT Telecontrol Configurator generates GOOSE control block instances that are derived/extended from FB_ScsmGoCBImplClass [▶ 262]. All function blocks derived from FB_ScsmGoCBImplClass [▶ 262] automatically have a default implementation of the above interface.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AcsiCommonControlBlockClass [▶ 363] -> I_AcsiCommonGooseControlBlockClass [▶ 368] -> I_ScsmGoCBImplClass

```
INTERFACE I_ScsmGoCBImplClass EXTENDS I_AcsiCommonGooseControlBlockClass
```

Properties

Name	Type	Access	Description
bFixedOffs	BOOL	Get	Attribute value: Enables/disables coding with fixed lengths.
bGoEna	BOOL	Get, Set	Attribute value: Enables/disables the GOOSE control block.
bNdsCom	BOOL	Get	Attribute value: Indicates whether the GOOSE control block requires further configuration.
ePRIORITY	E_AcsiVLanPriority [▶ 517]	Get	Priority level in the VLAN network.
ipDatSet	I_AcsiCommonDataSetClass [▶ 366]	Get	Interface pointer to the linked data set.
macAddr	ETHERNET_ADDRESS [▶ 528]	Get	Target MAC address.
nAPPID	WORD	Get	Application Identifier.
nConfRev	UDINT	Get	Revision number of the configuration of the GOOSE control block.
nMaxTime	DWORD	Get	Maximum time between repeated transmissions of a GOOSE message in milliseconds.
nMinTime	DWORD	Get	Time between direct sending after data change and first repetition in milliseconds.
nVID	WORD(0..4095)	Get	ID of the VLAN network.
oAddr	T_OCTET6 [▶ 558]	Get	Target MAC address.
sAddr	STRING(17)	Get	Target MAC address.
sDatSet	T_AcsiObjectReference [▶ 555]	Get, Set	Name of the linked data set.
sGoID	T_AcsiVisString129 [▶ 556]	Get, Set	The ID of the GOOSE messages.

6.5.3.28.5 I_ScsmGoCBClientJobClass

Objects implementing this interface control the behavior of the client GOCB control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Methods

Name	Description
EnableReq [▶ 459]	Enables the control block.
DisableReq [▶ 459]	Disables the control block.

6.5.3.28.5.1 EnableReq

Enables the GOCB control block.

```
METHOD FINAL EnableReq : BOOL
VAR_INPUT
    ipClient  : I_ScsmClientClass;
    ipDataSet : I_AcsiCommonDataSetClass;
END_VAR
VAR_OUTPUT
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipClient: Interface pointer of type: I_ScsmClientClass [▶ 409]. This pointer determines the service executing object instance of the IEC 61850 client.

ipDataSet: Interface pointer of type I_AcsiCommonDataSetClass [▶ 366].

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.28.5.2 DisableReq

Disables the GOCB control block.

```
METHOD FINAL DisableReq : BOOL
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.28.6 I_ScsmGoCBServerJobClass

Objects implementing this interface control the behavior of the server GOCB control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Methods

Name	Description
Execute [▶ 459]	Runs the GOCB state machine of the server.

6.5.3.28.6.1 Execute

Runs the GOCB state machine of the server.

```
METHOD Execute : BOOL
```

6.5.3.28.7 I_ScsmUrCBImplClass

Objects that implement this interface are report control block instances for unbuffered reports (unbuffered report control blocks, UBRCBs). The TwinCAT Telecontrol Configurator generates report control block instances that are derived/extended from FB_AcsiUnbufferedReportControlBlock [▶ 172]. All function blocks derived from FB_AcsiUnbufferedReportControlBlock [▶ 172] automatically have the interface referred to above implemented by default.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Inheritance hierarchy

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I_AcsiCommonControlBlockClass [▶ 363] -> I_AcsiCommonUnbufferedReportControlBlockClass [▶ 374] -> I_ScsmUrCBImplClass

```
INTERFACE I_ScsmUrCBImplClass EXTENDS I_AcsiCommonUnbufferedReportControlBlockClass
```

Properties

Name	Type	Access	Description
bGI	BOOL	Get, Set	Attribute value: Enabling the general interrogation command.
bResv	BOOL	Get, Set	Attribute value: Reservation of the RCB.
bRptEna	BOOL	Get, Set	Attribute value: Enabling/disabling the RCB control block. Further configuration needed.
cOptFlds	ST_AcsiOptionalFields [▶ 536]	Get, Set	Attribute value: Configuration of the optional fields that are transferred with a report.
cTrgOps	ST_AcsiTriggerConditions [▶ 544]	Get, Set	Attribute value: Configuration of the trigger options for sending a report.
nBufTm	DWORD	Get, Set	Attribute value: Configuration of the max. report buffer time in milliseconds.
nConfRev	DWORD	Get, Set	Attribute value: Revision number of the RCB control block configuration.
nIntgPd	DWORD	Get, Set	Attribute value: Configuration of the max. time between integrity reports in milliseconds.
nSqNum	BYTE	Get, Set	Attribute value: Report sequence number.
oOwner	T_OCTET64 [▶ 559]	Get, Set	Attribute value: Owner of the RCB. IP address of the client which has reserved the RCB for itself (offline/online reservation).
sDatSet	T_AcsiObjectReference [▶ 555]	Get, Set	Attribute value: Name of the linked data set.
sRptID	T_AcsiVisString129 [▶ 556]	Get, Set	Attribute value: Report identification string.
ipDatSet	I_AcsiCommonDataSetClass [▶ 366]	Get	Interface pointer of the linked data set.
ipClient	I_ScsmUrCBClientJobClass [▶ 460]	Get	Interface pointer of the client implementation of the URCB control block.
ipServer	I_ScsmUrCBServerJobClass [▶ 461]	Get	Interface pointer of the server implementation of the URCB control block.

6.5.3.28.8 I_ScsmUrCBClientJobClass

Objects implementing this interface control the behavior of the client UCB control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Methods

Name	Description
EnableReq [▶ 461]	Enables the control block.
DisableReq [▶ 461]	Disables the control block.
GIreq [▶ 461]	Enables the general interrogation command.

6.5.3.28.8.1 EnableReq

Enables the URCB control block.

```
METHOD FINAL EnableReq : BOOL
VAR_INPUT
    ipClient  : I_ScsmClientClass;
    ipDataSet : I_AcsiCommonDataSetClass;
END_VAR
VAR_OUTPUT
    ipResult  : I_AsyncServiceResultClass;
END_VAR
```

ipClient: Interface pointer of type: I_ScsmClientClass [▶ 409]. This pointer determines the service executing object instance of the IEC 61850 client.

ipDataSet: Interface pointer of type I_AcsiCommonDataSetClass [▶ 366].

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.28.8.2 DisableReq

Disables the URCB control block.

```
METHOD FINAL DisableReq : BOOL
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.28.8.3 GIreq

Enables the general interrogation command.

```
METHOD FINAL GIreq : BOOL
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

ipResult: Interface pointer of type: I_AsyncServiceResultClass [▶ 463] This pointer can be used to query and monitor the status/progress and the result of the service execution.

Return parameter: Positive feedback (TRUE) if the service primitive was sent/started successfully, negative feedback (FALSE) on error.

6.5.3.28.9 I_ScsmUrCBServerJobClass

Objects implementing this interface control the behavior of the server UBC control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Methods

Name	Description
Execute [▶ 462]	Runs the RCB state machine of the server.

6.5.3.28.9.1 Execute

Runs the RCB state machine of the server.

```
METHOD Execute : BOOL
```

6.5.3.28.10 I_ScsmMsvCBClientJobClass

Objects implementing this interface control the behavior of the client MSVCB control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

6.5.3.28.11 I_ScsmMsvCBServerJobClass

Objects implementing this interface control the behavior of the server MSVCB control block.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

6.5.3.28.12 I_ScsmMsvCBImplClass

Objects that implement this interface are Multicast Sampled Value control block instances (MSVCB). The TwinCAT Telecontrol Configurator generates control block instances that are derived/extended from FB_ScsmMsvCBImplClass [▶ 270]. All function blocks derived from FB_ScsmMsvCBImplClass [▶ 270] automatically have a default implementation of the above interface.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

I_AcsiCommonNodeClass [▶ 373] -> I_AcsiCommonDataClass [▶ 364] -> I-AcsiCommonControlBlockClass [▶ 363] -> I_AcsiCommonMulticastSampledValueControlClass [▶ 372] -> I_ScsmMsvCBImplClass

```
INTERFACE I_ScsmMsvCBImplClass EXTENDS I_AcsiCommonMulticastSampledValueControlBlockClass
```

Properties

Name	Type	Access	Description
bSvEna	BOOL	Get, Set	Attribute value: Enables/disables the MSV control block.
sMsvID	T_AcsiVisString129 [▶ 556]	Get, Set	Attribute value: The ID of the SV messages.
sDatSet	T_AcsiObjectReference [▶ 555]	Get, Set	Attribute value: Name of the linked data set.
nConfRev	UDINT	Get	Attribute value: Revision number of the MSV control block configuration.
nSmpRate	WORD	Get, Set	Attribute value: Sampling rate.
cOptFlds	ST_AcsiSvOptionalFields [▶ 543]	Get, Set	Attribute value: Configuration of the optional fields that are transmitted with an SV message.
eSmpMod	E_AcsiSamplingMode [▶ 503]	Get, Set	Attribute value: Sampling mode.
ePRIORITY	E_AcsiVLanPriority [▶ 517]	Get	DstAddress attribute value: Priority level in the VLAN network.
nVID	WORD(0..4095)	Get	DstAddress attribute value: ID of the VLAN network.
nAPPID	WORD	Get	DstAddress attribute value: Application Identifier.
macAddr	ETHERNET_ADDRESS [▶ 528]	Get	DstAddress attribute value: Destination MAC address.
oAddr	T_OCTET6 [▶ 558]	Get	DstAddress attribute value: Destination MAC address.
sAddr	STRING(17)	Get	DstAddress attribute value: Destination MAC address.
nNoASDU	WORD	Get, Set	Attribute value: Number of ASDUs in an SV message.
ipDatSet	I_AcsiCommonDataSetClass [▶ 366]	Get	Interface pointer to the linked data set.
ipClient	I_ScsmMsvCBClientJobClass [▶ 462]	Get	Interface pointer of the client implementation of the MSV control block.
ipServer	I_ScsmMsvCBServerJobClass [▶ 462]	Get	Interface pointer of the server implementation of the MSV control block.

6.5.4 I_AsyncServiceResultClass

Generic interface for status query and tracking of asynchronous processes. This includes services that require several PLC cycles to run. For objects that implement this interface, the status/progress and the result of the service execution can be monitored and queried.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

Inheritance hierarchy

I_AsyncServiceResultClass

```
INTERFACE I_AsyncServiceResultClass
```

Methods

Name	Description
CopyFrom [▶ 464]	Copies object data from another object instance
Dump [▶ 464]	Converts own object data into a formatted string
Equal [▶ 464]	Value comparison of own object data with external data values
EqualEx [▶ 465]	Value comparison of own object data with external data values
EqualTo [▶ 465]	Value comparison of own object data with the data of another object instance
IsAborted [▶ 465]	Checks whether the service was aborted
IsBusy [▶ 465]	Checks whether the service is still running
IsCompleted [▶ 466]	Checks whether the service execution was completed successfully
IsFailed [▶ 466]	Checks whether the service execution was completed with an error
SetAborted [▶ 466]	Sets the abort status
SetAbortedEx [▶ 466]	Sets the abort status
SetBusy [▶ 467]	Sets the status to busy
SetCompleted [▶ 467]	Sets the status to completed (without error)
SetFailed [▶ 467]	Sets the error state
SetFailedEx [▶ 467]	Sets the error state

Properties

Name	Type	Access	Description
eState [▶ 468]	E_AsyncServiceState [▶ 519]	Get	Status information

6.5.4.1 CopyFrom

Copies object data from another object instance.

```
METHOD CopyFrom : BOOL
VAR_INPUT
    ipSource : I_AsyncServiceResultClass;
END_VAR
```

ipSource: Interface pointer of type I_AsyncServiceResultClass [▶ 463] of the data source object instance.

Return parameter: TRUE => success, FALSE => error (e.g. interface pointer is null).

6.5.4.2 Dump

Converts own object data to a formatted string (e.g. for log outputs)

```
METHOD Dump : T_MaxString
```

Return parameter: String with the following format: ‚{State: {Severity: [Verbose][Info][Warning][Error][Critical], Id: n[nnnnn], class: „NameOrGuiid“}}‘.

6.5.4.3 Equal

Value comparison of own object data (status and error information) with external data values.

```
METHOD Equal : BOOL
VAR_INPUT
    eOther  : E_AsyncServiceState;
    ipOther : I_ServiceErrorClass;
END_VAR
```

eOther: Enumeration type E_AsyncServiceState [▶ 519] with the status to be compared.

ipOther: Interface pointer of type I_ServiceErrorClass [▶ 468] of the object instance with the error information to be compared.

Return parameter: TRUE => data identical, FALSE => data differ.

6.5.4.4 EqualEx

Value comparison of own object data (status and error information) with external data values.

```
METHOD EqualEx : BOOL
VAR_INPUT
    eOther  : E_AsyncServiceState;
    stOther : ST_ServiceErrorClass;
END_VAR
```

eOther: Enumeration type E_AsyncServiceState [▶ 519] with the status to be compared.

stOther: Structured variable of type ST_ServiceErrorClass [▶ 555] with the error information to be compared.

Return parameter: TRUE => data identical, FALSE => data differ.

6.5.4.5 EqualTo

Value comparison of own object data (status and error information) with the data of another object instance.

```
METHOD EqualTo : BOOL
VAR_INPUT
    ipOther : I_AsyncServiceResultClass;
END_VAR
```

ipOther: Interface pointer of type I_AsyncServiceResultClass [▶ 463] of the object instance whose data are to be compared.

Return parameter: TRUE => data identical, FALSE => data differ.

6.5.4.6 IsAborted

Checks whether the service was aborted

```
METHOD IsAborted : BOOL
```

Return parameter: TRUE => aborted, FALSE => not aborted (busy or completed or failed).

6.5.4.7 IsBusy

Checks whether the service is still running.

```
METHOD IsBusy : BOOL
```

Return parameter: TRUE => busy, FALSE => not busy (completed or aborted or failed).

Example (extract)

```
METHOD MyServiceReq : BOOL
VAR_INPUT
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass := _fbResult;
END_VAR
```

```
IF ipResult.IsBusy() THEN(* service allready running => abort *)
    _state := 0;
    ipResult.SetAbortedEx( stReason := ERROR_INVALID_STATE );
    RETURN;
END_IF
IF tTimeout = T#0S THEN(* invalid parameter => failure *)
    ipResult.SetFailedEx( stError := ERROR_INVALID_VALUE );
    RETURN;
```

```
END_IF
ipResult.SetBusy();
_fbTimer( IN := FALSE, PT := tTimeout );
_state := 1;
THIS^();
MyServiceReq := TRUE;
```

6.5.4.8 IsCompleted

Checks whether the service execution was completed successfully.

```
METHOD IsCompleted : BOOL
```

Return parameter: TRUE => completed without error. FALSE => not completed (busy or aborted or failed).

6.5.4.9 IsFailed

Checks whether the service execution was completed with an error.

```
METHOD IsFailed : BOOL
```

Return parameter: TRUE => completed with an error, FALSE => not completion with an error (busy or completed or aborted).

6.5.4.10 SetAborted

Sets the abort status and saves the data about the cause of the termination in its own object instance.

```
METHOD SetAborted : I_AsyncServiceResultClass
VAR_INPUT
    ipReason : I_ServiceErrorClass;
END_VAR
```

ipReason: Interface pointer of type I_ServiceErrorClass [▶ 468] of an object instance with the data on the cause of the termination.

Return parameter: Own interface pointer of type I_AsyncServiceResultClass [▶ 463].

6.5.4.11 SetAbortedEx

Sets the abort status and saves the data about the cause of the termination in its own object instance.

```
METHOD SetAbortedEx : I_AsyncServiceResultClass
VAR_INPUT
    stReason : ST_ServiceErrorClass;
END_VAR
```

stReason: Structured variable of type ST_ServiceErrorClass [▶ 555] with the data on the cause of the termination.

Return parameter: Own interface pointer of type I_AsyncServiceResultClass [▶ 463].

Example (extract)

```
METHOD MyServiceReq : BOOL
VAR_INPUT
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass := _fbResult;
END_VAR
```

```
IF ipResult.IsBusy() THEN(* service allready running => abort *)
    _state := 0;
    ipResult.SetAbortedEx( stReason := ERROR_INVALID_STATE );
    RETURN;
END_IF
IF tTimeout = T#0S THEN(* invalid parameter => failure *)
    ipResult.SetFailedEx( stError := ERROR_INVALID_VALUE );
    RETURN;
END_IF
ipResult.SetBusy();
```

```
_fbTimer( IN := FALSE, PT := tTimeout );
_state := 1;
THIS^();
MyServiceReq := TRUE;
```

6.5.4.12 SetBusy

Sets the status in the own object instance to "busy“.

```
METHOD SetBusy : I_AsyncServiceResultClass
```

Return parameter: Own interface pointer of type I_AsyncServiceResultClass [▶ 463].

Example (extract)

```
METHOD MyServiceReq : BOOL
VAR_INPUT
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass := _fbResult;
END_VAR
```

```
IF ipResult.IsBusy() THEN(* service allready running => abort *)
    _state := 0;
    ipResult.SetAbortedEx( stReason := ERROR_INVALID_STATE );
    RETURN;
END_IF
IF tTimeout = T#0S THEN(* invalid parameter => failure *)
    ipResult.SetFailedEx( stError := ERROR_INVALID_VALUE );
    RETURN;
END_IF
ipResult.SetBusy();
_fbTimer( IN := FALSE, PT := tTimeout );
_state := 1;
THIS^();
MyServiceReq := TRUE;
```

6.5.4.13 SetCompleted

Sets the status in the own object instance to "completed" (without error).

```
METHOD SetCompleted : I_AsyncServiceResultClass
```

Return parameter: Own interface pointer of type I_AsyncServiceResultClass [▶ 463].

6.5.4.14 SetFailed

Sets the error state and saves the data about the cause of the error in its own object instance.

```
METHOD SetFailed : I_AsyncServiceResultClass
VAR_INPUT
    ipError : I_ServiceErrorClass;
END_VAR
```

ipError: Interface pointer of type I_ServiceErrorClass [▶ 468] of an object instance with the data on the cause of the error.

Return parameter: Own interface pointer of type I_AsyncServiceResultClass [▶ 463].

6.5.4.15 SetFailedEx

Sets the error state and saves the data about the cause of the error in its own object instance.

```
METHOD SetFailedEx : I_AsyncServiceResultClass
VAR_INPUT
    stError : ST_ServiceErrorClass;
END_VAR
```

stError: Structured variable of type ST_ServiceErrorClass [▶ 555] with the data on the cause of the error.

Return parameter: Own interface pointer of type I_AsyncServiceResultClass [▶ 463].

Example (extract)

```
METHOD MyServiceReq : BOOL
VAR_INPUT
    tTimeout : TIME;
END_VAR
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass := _fbResult;
END_VAR
```

```
IF ipResult.IsBusy() THEN(* service allready running => abort *)
    _state := 0;
    ipResult.SetAbortedEx( stReason := ERROR_INVALID_STATE );
    RETURN;
END_IF
IF tTimeout = T#0S THEN(* invalid parameter => failure *)
    ipResult.SetFailedEx( stError := ERROR_INVALID_VALUE );
    RETURN;
END_IF
ipResult.SetBusy();
_fbTimer( IN := FALSE, PT := tTimeout );
_state := 1;
THIS^();
MyServiceReq := TRUE;
```

6.5.4.16 eState

Asynchronous service status.

```
PROPERTY eState : E_AsyncServiceState {GET}
```

eState: Status of type E_AsyncServiceState [▶ 519].

6.5.5 I_ServiceErrorClass

Generic interface for error query and storage. For objects that implement this interface, error information such as the error code, source of error or error text can be queried or compared.

Inheritance hierarchy

I_ServiceErrorClass

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
INTERFACE I_ServiceErrorClass
```

Methods

Name	Description
CopyFrom [▶ 469]	Copies object data from another object instance
Dump [▶ 469]	Converts own object data to a formatted string (e.g. for log outputs)
Equal [▶ 469]	Value comparison of own object data (error information) with external data values
EqualEx [▶ 469]	Value comparison of own object data (error information) with external data values
EqualTo [▶ 469]	Value comparison of own object data (error information) with the data of another object instance
IsFailed [▶ 470]	Checks own object data (error information) for failure
IsSucceeded [▶ 470]	Checks own object data (error information) for success
SetFailedEx [▶ 470]	Stores error information in its own object instance
SetSucceeded [▶ 470]	Stores success information in own object instance

Properties

Name	Type	Access	Description
stError [▶ 471]	ST_ServiceErrorClass [▶ 555]	Get	Object data as a structured type

6.5.5.1 CopyFrom

Copies object data from another object instance.

```
METHOD CopyFrom : BOOL
VAR_INPUT
    ipOther : I_ServiceErrorClass;
END_VAR
```

ipOther: Interface pointer of type I_ServiceErrorClass [▶ 468] of the data source object instance.

Return parameter: TRUE => success, FALSE => error (e.g. interface pointer is null).

6.5.5.2 Dump

Converts own object data to a formatted string (e.g. for log outputs)

```
METHOD Dump : T_MaxString
```

Return parameter: String with the following format: '{Severity: [Verbose][Info][Warning][Error][Critical], Id: n[nnnnn], class: „NameOrGuiid" }'.

6.5.5.3 Equal

Value comparison of own object data (error information) with external data values.

```
METHOD Equal : BOOL
VAR_INPUT
    uuidOther : GUID;
    nOther    : UDINT;
    eOther    : TcEventSeverity;
END_VAR
```

uuidOther: The event class of type GUID to be compared.

nOther: The event ID to be compared.

eOther: The event of the type TcEventSeverity [▶ 530] to be compared.

Return parameter: TRUE => data identical, FALSE => data differ.

6.5.5.4 EqualEx

Value comparison of own object data (error information) with external data values.

```
METHOD EqualEx : BOOL
VAR_INPUT
    stOther : ST_ServiceErrorClass;
END_VAR
```

stOther: Structured variable of type ST_ServiceErrorClass [▶ 555] with the data values to be compared.

Return parameter: TRUE => data identical, FALSE => data differ.

6.5.5.5 EqualTo

Value comparison of own object data (error information) with the data of another object instance.

```
METHOD EqualTo : BOOL
VAR_INPUT
    ipOther : I_ServiceErrorClass;
END_VAR
```

ipOther: Interface pointer of type I_ServiceErrorClass [▶ 468] of the object instance whose data is to be compared.

Return parameter: TRUE => data identical, FALSE => data differ.

6.5.5.6 IsFailed

Checks own object data (error information) for failure.

```
METHOD IsFailed : BOOL
```

Return parameter: TRUE => failure, FALSE => success.

6.5.5.7 IsSucceeded

Checks own object data (error information) for success.

```
METHOD IsSucceeded : BOOL
```

Return parameter: TRUE => success, FALSE => failure.

6.5.5.8 SetFailedEx

Stores error information in its own object instance.

```
METHOD SetFailedEx : I_ServiceErrorClass
VAR_INPUT
    stError : ST_ServiceErrorClass;
END_VAR
```

stError: Structured variable of type ST_ServiceErrorClass [▶ 555] with the data on the cause of the error.

Return parameter: Own interface pointer of type I_ServiceErrorClass [▶ 468].

Sample (extract)

```
METHOD DoSomething : BOOL
VAR_INPUT
    sName : STRING;
END_VAR
VAR_OUTPUT
    ipError : I_ServiceErrorClass := _fbError;
END_VAR
IF sName = '' THEN
    ipError.SetFailedEx( stError := TC_EVENTS.TcGeneralAdsEventClass.InvalidParam );
    RETURN;
END_IF

(* ToDo: *)

ipError.SetSucceeded();
DoSomething := TRUE;
```

6.5.5.9 SetSucceeded

Stores success information in its own object instance.

```
METHOD SetSucceeded : I_ServiceErrorClass
```

Return parameter: Own interface pointer of type I_ServiceErrorClass [▶ 468].

Sample (extract)

```
METHOD DoSomething : BOOL
VAR_INPUT
    sName : STRING;
END_VAR
VAR_OUTPUT
    ipError : I_ServiceErrorClass := _fbError;
END_VAR
IF sName = '' THEN
```

```
    ipError.SetFailedEx( stError := TC_EVENTS.TcGeneralAdsEventClass.InvalidParam );
    RETURN;
END_IF

(* ToDo: *)

ipError.SetSucceeded();
DoSomething := TRUE;
```

6.5.5.10 stError

```
PROPERTY stError : ST_ServiceErrorClass (GET)
```

stError: Error information as structured type ST_ServiceErrorClass [▶ 555].

6.5.6 I_BaseStreamClass

Inheritance hierarchy

I_BaseStreamClass

```
INTERFACE I_BaseStreamClass
```

6.6 Data types

6.6.1 E_ACSE_AareDiagnostic

Diagnostic information, sent in the ACSE (Association Control Service Element) Associate Response (AARE) telegram.

Namespace: Tc3_Ulosi [▶ 102]
Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ACSE_AareDiagnostic :
(
    Null                                          :=0,
    No_reason_given                               :=1,
    Application_context_name_not_supported        :=2,
    Calling_AP_title_not_recognized               :=3,
    Calling_AP_invocation_identifier_not_recognized :=4,
    Calling_AE_qualifier_not_recognized           :=5,
    Calling_AE_invocation_identifier_not_recognized :=6,
    Called_AP_title_not_recognized                :=7,
    Called_AP_invocation_identifier_not_recognized  :=8,
    Called_AE_qualifier_not_recognized            :=9,
    Called_AE_invocation_identifier_not_recognized  :=10,
    Authentication_mechanism_name_not_recognized  :=11,
    Authentication_mechanism_name_required        :=12,
    Authentication_failure                        :=13,
    Authentication_required                       :=14
)USINT;
END_TYPE
```

Values

Name	Description
Null	Null.
No_reason_given	No reason given.
Application_context_name_not_supported	The code meaning depends on the diagnostic source: acse-service-user => Application context name not supported, acse-service-provider => No common ACS version.
Calling_AP_title_not_recognized	ISO 8650-1 setting "Calling AP title" was not recognized.
Calling_AP_invocation_identifier_not_recognized	ISO 8650-1 setting "Calling AP invocation identifier" was not recognized.
Calling_AE_qualifier_not_recognized	ISO 8650-1 setting "Calling AE qualifier" was not recognized.
Calling_AE_invocation_identifier_not_recognized	ISO 8650-1 setting "Calling AE invocation identifier" was not recognized.
Called_AP_title_not_recognized	ISO 8650-1 setting "Called AP title" was not recognized.
Called_AP_invocation_identifier_not_recognized	ISO 8650-1 setting "Called AP invocation identifier" was not recognized.
Called_AE_qualifier_not_recognized	ISO 8650-1 setting "Called AE qualifier" was not recognized.
Called_AE_invocation_identifier_not_recognized	ISO 8650-1 setting "Called AE invocation identifier" was not recognized.
Authentication_mechanism_name_not_recognized	The name of the authentication mechanism was not recognized.
Authentication_mechanism_name_required	The name of the authentication mechanism is required.
Authentication_failure	Authentication failed.
Authentication_required	Authentication is required.

6.6.2 E_ACSE_AareResult

Result information, sent in the ACSE (Association Control Service Element) Associate-Response (AARE) telegram.

Namespace: Tc3_Ulosi [▶ 102]
Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ACSE_AareResult :
(
    Accepted           :=0,
    Rejected_Permanent :=1,
    Rejected_Transient :=2
)USINT;
END_TYPE
```

Values

Name	Description
Accepted	Accepted
Rejected_Permanent	Permanently rejected
Rejected_Transient	Temporarily rejected

6.6.3 E_ACSE_AareSource

Diagnostic source, sent in the ACSE (Association Control Service Element) Associate Response (AARE) telegram.

Namespace: Tc3_Ulosi [▶ 102]
Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ACSE_AareSource :
(
    ServiceUser     :=1,
    ServiceProvider :=2
)USINT;
END_TYPE
```

Values

Name	Description
ServiceUser	Service user
ServiceProvider	Service provider

6.6.4 E_ACSE_AbrtDiagnostic

Diagnostic information, sent in the ACSE (Association Control Service Element) Associate-Abort (ABRT) telegram.

Namespace: Tc3_Ulosi [▶ 102]
Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ACSE_AbrtDiagnostic :
(
    Null                                        :=0,
    No_reason_given                             :=1,
    Protocol_error                              :=2,
    Authentication_mechanism_name_not_recognized :=3,
    Authentication_mechanism_name_required      :=4,
    Authentication_failure                      :=5,
    Authentication_required                     :=6
)USINT;
END_TYPE
```

Values

Name	Description
Null	Value not available.
No_reason_given	No reason given.
Protocol_error	Protocol error
Authentication_mechanism_name_not_recognized	The name of the authentication mechanism was not recognized.
Authentication_mechanism_name_required	The name of the authentication mechanism is required.
Authentication_failure	Authentication failed
Authentication_required	Authentication required

6.6.5 E_ACSE_AbrtSource

Diagnostic source, sent in the ACSE (Association Control Service Element) Associate-Abort (ABRT) telegram.

Namespace: Tc3_Ulosi [▶ 102]
Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ACSE_AbrtSource :
(
    ServiceUser     :=0,
    ServiceProvider :=1
)USINT;
END_TYPE
```

Values

Name	Description
ServiceUser	Service user
ServiceProvider	Service provider

6.6.6 E_AcseAuthentication

Type of ACSE (Association Control Service Element) authentication.

Namespace: Tc3_Ulosi [▶ 102]
Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcseAuthentication :
(
    Unilateral :=0,
    Mutual     :=1
)SINT;
END_TYPE
```

Values

Name	Description
Unilateral	Unilateral authentication (default)
Mutual	Mutual authentication (two parties authenticate simultaneously).

6.6.7 E_AcsiAccessPermission

Configures the restriction of the access to a data value.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiAccessPermission :
(
    RO := 1,
    WO := 2,
    RW := 3
)SINT;
END_TYPE
```

Values

Name	Description
RO	Read access only.
WO	Write access only.
RW	Read and write access.

6.6.8 E_AcsiAdjSt

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiAdjSt :
(
    Completed      := 1,
    Cancelled      := 2,
    NewAdjustments := 3,
    UnderWay       := 4
)SINT;
END_TYPE
```

Values

Name	Description
Completed	
Cancelled	
NewAdjustments	
UnderWay	

6.6.9 E_AcsiAlmLev

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiAlmLev :
(
    Low    := 1,
    Normal := 2,
    Urgent := 3
) SINT;
END_TYPE
```

Values

Name	Description
Low	
Normal	
Urgent	

6.6.10 E_AcsiAnalogueValuePresCond

Configures the presence of the AnalogueValue "i" and "f" attributes in the status values (mxVal) and control values (ctlVal).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiAnalogueValuePresCond :
(
    i         := 2#0101,
    f         := 2#1010,
    both_CO_i := 2#0111,
    both_CO_f := 3#1011
)SINT;
END_TYPE
```

Values

Name	Status values (mxVal)	Control values (ctlVal)
i	i	i
f	f	f
both_CO_i	i and f	i
both_CO_f	i and f	f

6.6.11 E_AcsiAngRef

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiAngRef :
(
    Va            := 0,
    Vb            := 1,
    Vc            := 2,
    Aa            := 3,
    Ab            := 4,
    Ac            := 5,
    Vab           := 6,
    Vbc           := 7,
    Vca           := 8,
    Vother        := 9,
    Aother        := 10,
    Synchrophasor := 11
)SINT;
END_TYPE
```

Values

Name	Description
Va	
Vb	
Vc	
Aa	
Down	
Ac	
Vab	
Vbc	
Vca	
Vother	
Aother	
Synchronphasor	

6.6.12 E_AcsiApplAddCause

Additional diagnostic information about the cause of a negative feedback when a control service is executed (operate, select, cancel etc.).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiApplAddCause :
(
    Unknown                     := 0,
    NotSupported                := 1,
    BlockedBySwitchingHierarchy := 2,
```

```
    SelectFailed                := 3,
    InvalidPosition             := 4,
    PositionReached             := 5,
    ParameterChangeInExecution  := 6,
    StepLimit                   := 7,
    BlockedByMode               := 8,
    BlockedByProcess            := 9,
    BlockedByInterlocking       := 10,
    BlockedBySynchrocheck       := 11,
    CommandAlreadyInExecution   := 12,
    BlockedByHealth             := 13,
    OneOfNControl               := 14,
    AbortionByCancel            := 15,
    TimeLimitOver               := 16,
    AbortionByTrip              := 17,
    ObjectNotSelected           := 18,
    ObjectAlreadySelected       := 19,
    NoAccessAuthority           := 20,
    EndedWithOvershoot          := 21,
    AbortionDueToDeviation      := 22,
    AbortionByCommunicationLoss := 23,
    BlockedByCommand            := 24,
    None                        := 25,
    InconsistentParameters      := 26,
    LockedByOtherClient         := 27
)SINT;
END_TYPE
```

Values

Name	Description
Unknown	Unknown.
NotSupported	Not-supported.
BlockedBySwitchingHierarchy	Blocked-by-switching-hierarchy.
SelectFailed	Select-failed.
InvalidPosition	Invalid-position.
PositionReached	Position-reached.
ParameterChangeInExecution	Parameter-change-in-execution.
StepLimit	Step-limit.
BlockedByMode	Blocked-by-Mode.
BlockedByProcess	Blocked-by-process.
BlockedByInterlocking	Blocked-by-interlocking.
BlockedBySynchrocheck	Blocked-by-synchrocheck.
CommandAlreadyInExecution	Command-already-in-execution.
BlockedByHealth	Blocked-by-health.
OneOfNControl	1-of-n-control.
AbortionByCancel	Abortion-by-cancel.
TimeLimitOver	Time-limit-over.
AbortionByTrip	Abortion-by-trip.
ObjectNotSelected	Object-not-selected.
ObjectAlreadySelected	Object-already-selected.
NoAccessAuthority	No-access-authority.
EndedWithOvershoot	Ended-with-overshoot.
AbortionDueToDeviation	Abortion-due-to-deviation.
AbortionByCommunicationLoss	Abortion-by-communication-loss.
lockedByCommand	Blocked-by-command.
None	None.
InconsistentParameters	Inconsistent-parameters.
LockedByOtherClient	Locked-by-other-client.

6.6.13 E_AcsiApplError

Diagnostic information about the source of the error (for example the state machine of services, the Operate or TimeActivated service itself, or another source).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiApplError :
(
    NoError           := 0,
    Unknown           := 1,
    TimeoutTestNotOk  := 2,
    OperatorTestNotOk := 3
)SINT;
END_TYPE
```

Values

Name	Description
NoError	No error.
Unknown	Unknown.
TimeoutTestNotOk	Timeout test not ok.
OperatorTestNotOk	Operator test not ok.

6.6.14 E_AcsiAutoRecSt

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiAutoRecSt :
(
    Ready                 := 1,
    InProgress            := 2,
    Successful            := 3,
    WaitingForTrip        := 4,
    TripIssuedByProtection := 5,
    FaultDisappeared      := 6,
    WaitToComplete        := 7,
    CircuitBreakerClosed  := 8,
    CycleUnsuccessful     := 9,
    Unsuccessful          := 10,
    Aborted               := 11
)SINT;
END_TYPE
```

Values

Name	Description
Ready	
InProgress	
Successful	
WaitingForTrip	
TripIssuedByProtection	
FaultDisappeared	
WaitToComplete	
CircuitBreakerClosed	
CycleUnsuccessful	
Unsuccessful	
Aborted	

6.6.15 E_AcsiBeh

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiBeh :
(
    On          := 1,
    OnBlocked   := 2,
    Test        := 3,
    TestBlocked := 4,
    Off         := 5
)SINT;
END_TYPE
```

Values

Name	Description
On	
OnBlocked	"blocked" in Ed1 is changed in Ed2 to "on-blocked".
Test	
TestBlocked	
Off	

6.6.16 E_AcsiCBOpCap

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiCBOpCap :
(
    None                    := 1,
    Open                    := 2,
    CloseOpen               := 3,
    OpenCloseOpen           := 4,
    CloseOpenCloseOpen      := 5,
    OpenCloseOpenCloseOpen  := 6,
    More                    := 7
)SINT;
END_TYPE
```

Values

Name	Description
None	
Open	
CloseOpen	
OpenCloseOpen	
CloseOpenCloseOpen	
OpenCloseOpenCloseOpen	
More	

6.6.17 E_AcsiCheck

Checks to be performed when a control service is executed. The enumeration value is coded in 2 bits. The remaining 6 bits are not used.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiCheck :
(
    NoCheck           := 2#00,
    SynchroCheck      := 2#01,
    InterlockingCheck := 2#10,
    Both              := 2#11
)SINT;
END_TYPE
```

Values

Name	Description
NoCheck	No check.
SynchroCheck	Synchro-check.
InterlockingCheck	Interlocking-check.
Both	Both checks.

6.6.18 E_AcsiClcIntvTyp

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiClcIntvTyp :
(
    Ms       := 1,
    PerCycle := 2,
    Cycle    := 3,
    Day      := 4,
    Week     := 5,
    Month    := 6,
    Year     := 7,
    External := 8
)SINT;
END_TYPE
```

Values

Name	Description
Ms	
PerCycle	
Cycle	
Day	
Week	
Month	
Year	
External	

6.6.19 E_AcsiClcMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiClcMod :
(
    Total   := 1,
    Period  := 2,
    Sliding := 3
)SINT;
END_TYPE
```

Values

Name	Description
Total	
Period	
Sliding	

6.6.20 E_AcsiClcMth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiClcMth :
(
    Unspecified     := 1,
    TrueRms         := 2,
    PeakFundamental := 3,
    RmsFundamental  := 4,
    Min_            := 5,
    Max_            := 6,
    Avg             := 7,
    Sdv             := 8,
    Prediction      := 9,
    Rate            := 10
)SINT;
END_TYPE
```

Values

Name	Description
Unspecified	
TrueRms	
PeakFundamental	
RmsFundamental	
Min_	
Max_	
Avg	
Sdv	
Prediction	
Rate	

6.6.21 E_AcsiClcRfTyp

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiClcRfTyp :
(
    Ms       := 1,
    PerCycle := 2,
    Cycle    := 3,
    Day      := 4,
    Week     := 5,
    Month    := 6,
    Year     := 7,
    External := 8
)SINT;
END_TYPE
```

Values

Name	Description
Ms	
PerCycle	
Cycle	
Day	
Week	
Month	
Year	
External	

6.6.22 E_AcsiClcTotVA

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiClcTotVA :
(
    Vector     := 1,
    Arithmetic := 2
)SINT;
END_TYPE
```

Values

Name	Description
Vector	
Arithmetic	

6.6.23 E_AcsiCmdQual

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiCmdQual :
(
    Pulse       := 0,
    Persistent_ := 1
)SINT;
END_TYPE
```

Values

Name	Description
Pulse	
Persistent_	

6.6.24 E_AcsiCtlModels

Control model for the switch control.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiCtlModels :
```

```
(
    StatusOnly                 := 0,
    DirectWithNormalSecurity   := 1,
    SBOWithNormalSecurity      := 2,
    DirectWithEnhancedSecurity := 3,
    SBOWithEnhancedSecurity    := 4
)SINT;
END_TYPE
```

Values

Name	Description
StatusOnly	Status information only, control not allowed.
DirectWithNormalSecurity	Direct-with-normal-security (direct-operate).
SBOWithNormalSecurity	SBO-with-normal-security (operate-once | operate-many).
DirectWithEnhancedSecurity	Direct-with-enhanced-security (direct-operate).
SBOWithEnhancedSecurity	SBO-with-enhanced-security (operate-once | operate-many).

6.6.25 E_AcsiCtlReport

Control commands of a report control block instance for buffered or unbuffered reports (BRCB, URCB).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiCtlReport :
(
    None   := 0,
    Enable := 1,
    GI     := 2,
    Disable:= 3
)SINT;
END_TYPE
```

Values

Name	Description
None	No command (idle state).
Enable	Command to activate the reports (buffered or unbuffered).
GI	Command to activate the general interrogation.
Disable	Command to deactivate the reports (buffered or unbuffered) and the general interrogation.

6.6.26 E_AcsiCtlService

Control commands for the services of the data objects for the switch control.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiCtlService :
(
    None             := 0,
    Select           := 1,
    Operate          := 2,
    SelectAndOperate := 3,
    Cancel           := 4
)SINT;
END_TYPE
```

Values

Name	Description
None	No command (idle state).
Select	Activates the Select or SelectWithValue command.
Operate	Activates the Operate command.
SelectAndOperate	Activates the Select command, followed by the Operate command.
Cancel	Activates the Cancel command.

6.6.27 E_AcsiCycTrMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiCycTrMod :
(
    ThreePhaseTripping      := 1,
    OneOrThreePhaseTripping := 2,
    Specific                := 3
)SINT;
END_TYPE
```

Values

Name	Description
ThreePhaseTripping	
OneOrThreePhaseTripping	
Specific	

6.6.28 E_AcsiDbpos

Double-point status value. The enumeration value is coded in 2 bits. The remaining 6 bits are not used.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiDbpos :
(
    Intermediate := 2#00,
    Off          := 2#01,
    On           := 2#10,
    Bad          := 2#11
)SINT;
END_TYPE
```

Values

Name	Description
Intermediate	Intermediate state
Off	Off
On	On
Bad	No good

6.6.29 E_AcsiDir

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiDir :
(
    Unknown  := 0,
    Forward  := 1,
    Backward := 2,
    Both     := 3
)SINT;
END_TYPE
```

Values

Name	Description
Unknown	
Forward	
Backward	
Both	

6.6.30 E_AcsiDirMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiDirMod :
(
    NonDirectional := 1,
    Forward        := 2,
    Reverse        := 3
)SINT;
END_TYPE
```

Values

Name	Description
NonDirectional	
Forward	
Reverse	

6.6.31 E_AcsiEEHealth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiEEHealth :
(
    Ok      := 1,
    Warning := 2,
    Alarm   := 3
)SINT;
END_TYPE
```

Values

Name	Description
Ok	
Warning	
Alarm	

6.6.32 E_AcsiFailMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiFailMod :
(
    Current                    := 1,
    BreakerStatus              := 2,
    BothCurrentAndBreakerStatus := 3,
    Other                      := 4
)SINT;
END_TYPE
```

Values

Name	Description
Current	
BreakerStatus	
BothCurrentAndBreakerStatus	
Other	

6.6.33 E_AcsiFanCtl

Control value for fan control.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiFanCtl :
(
    Inactive := 1,
    Stage1   := 2,
    Stage2   := 3,
    Stage3   := 4
)SINT;
END_TYPE
```

Values

Name	Description
Inactive	
Stage1	
Stage2	
Stage3	

6.6.34 E_AcsiFanCtlGen

Generic control value for fan control.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiFanCtlGen :
(
    Inactive := 1,
    Stage1   := 2,
    Stage2   := 3,
    Stage3   := 4
)SINT;
END_TYPE
```

Values

Name	Description
Inactive	
Stage1	
Stage2	
Stage3	

6.6.35 E_AcsiFc

Functional group as enumeration type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiFc :
(
    None := 16#000000,
    MX   := 16#000001,
    ST_  := 16#000002,
    CO   := 16#000004,
    CF   := 16#000008,
    DC   := 16#000010,
    SP   := 16#000020,
    SG   := 16#000040,
    RP   := 16#000080,

    LG   := 16#000100,
    BR   := 16#000200,
    GO   := 16#000400,
    GS   := 16#000800,
    SV   := 16#001000,
    SE   := 16#002000,
    MS   := 16#004000,
    SC   := 16#008000,

    US   := 16#010000,
    EX   := 16#020000,

    XX   := 16#03FFFF
)DWORD;
END_TYPE
```

Values

Name	Description
None	
MX	
ST_	
CO	
CF	
DC	
SP	
SG	
RP	
LG	
BR	
GO	
GS	
SV	
SE	
MS	
SC	
US	
EX	
XX	

6.6.36 E_AcsiFilTyp

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiFilTyp :
(
    LowPass  := 1,
    HighPass := 2,
    Bandpass := 3,
    Bandstop := 4,
    Deadband := 5
)SINT;
END_TYPE
```

Values

Name	Description
LowPass	
HighPass	
Band-pass	
Bandstop	
Deadband	

6.6.37 E_AcsiFltLoop

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiFltLoop :
```

```
(
    PhaseAToGround := 1,
    PhaseBToGround := 2,
    PhaseCToGround := 3,
    PhaseAToB      := 4,
    PhaseBToC      := 5,
    PhaseCToA      := 6,
    Other          := 7
)SINT;
END_TYPE
```

Values

Name	Description
PhaseAToGround	
PhaseBToGround	
PhaseCToGround	
PhaseAToB	
PhaseBToC	
PhaseCToA	
Other	

6.6.38 E_AcsiGnSt

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiGnSt :
(
    Stopped  := 1,
    Stopping := 2,
    Started  := 3,
    Starting := 4,
    Disabled := 5
)SINT;
END_TYPE
```

Values

Name	Description
Stopped	
Stopping	
Started	
Starting	
Disabled	

6.6.39 E_AcsiHealth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiHealth :
(
    Ok      := 1,
    Warning := 2,
    Alarm   := 3
)SINT;
END_TYPE
```

Values

Name	Description
Ok	
Warning	
Alarm	

6.6.40 E_AcsiHvRef

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiHvRef :
(
    Fundamental := 0,
    Rms         := 1,
    Absolute    := 2
)SINT;
END_TYPE
```

Values

Name	Description
Fundamental	
Rms	
Absolute	

6.6.41 E_AcsiIntrDetMth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiIntrDetMth :
(
    Voltage                                                 := 1,
    VoltageAndCurrent                                       := 2,
    VoltageAndNormallyOpenBreakerContact                    := 3,
    VoltageAndNormallyClosedBreakerContact                  := 4,
    VoltageAndBothNormallyOpenAndNormallyClosedBreakerContacts := 5,
    NormallyOpenBreakerContact                              := 6,
    NormallyClosedBreakerContacts                           := 7,
    BothNormallyOpenAndNormallyClosedBreakerContacts        := 8
)SINT;
END_TYPE
```

Values

Name	Description
Voltage	
VoltageAndCurrent	
VoltageAndNormallyOpenBreakerContact	
VoltageAndNormallyClosedBreakerContact	
VoltageAndBothNormallyOpenAndNormallyClosedBreakerContacts	
NormallyOpenBreakerContact	
NormallyClosedBreakerContacts	
BothNormallyOpenAndNormallyClosedBreakerContacts	

6.6.42 E_AcsiLevMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiLevMod :
(
    PositiveOrRising  := 1,
    NegativeOrFalling := 2,
    Both              := 3,
    Other             := 4
)SINT;
END_TYPE
```

Values

Name	Description
PositiveOrRising	
NegativeOrFalling	
Both	
Other	

6.6.43 E_AcsiLivDeaMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiLivDeaMod :
(
    DeadLineDeadBus                                 := 1,
    LiveLineDeadBus                                 := 2,
    DeadLineLiveBus                                 := 3,
    DeadLineDeadBusOrLiveLineDeadBus                := 4,
    DeadLineDeadBusOrDeadLineLiveBus                := 5,
    LiveLineDeadBusOrDeadLineLiveBus                := 6,
    DeadLineDeadBusOrLiveLineDeadBusOrDeadLineLiveBus := 7
)SINT;
END_TYPE
```

Values

Name	Description
DeadLineDeadBus	
LiveLineDeadBus	
DeadLineLiveBus	
DeadLineDeadBusOrLiveLineDeadBus	
DeadLineDeadBusOrDeadLineLiveBus	
LiveLineDeadBusOrDeadLineLiveBus	
DeadLineDeadBusOrLiveLineDeadBusOrDeadLineLiveBus	

6.6.44 E_AcsiLogicalNodeClass

Object classes of a logical node.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiLogicalNodeClass :
(
```

```
    DataObject                          := 1,
    DataSet                             := 2,
    BufferedReportControlBlock          := 3,
    UnbufferedReporControlBlock         := 4,
    LogControlBlock                     := 5,
    SettingGroupControlBlock            := 6,
    Log_                                := 7,
    GooseControlBlock                   := 8,
    GsseControlBlock                    := 9,
    MulticastSampledValueControlBlock := 10,
    UnicastSampledValueControlBlock   := 11
)SINT;
END_TYPE
```

Values

Name	Description
DataObject	
DataSet	
BufferedReportControlBlock	
UnbufferedReportControlBlock	
LogControlBlock	
SettingGroupControlBlock	
Log_	
GooseControlBlock	
GsseControlBlock	
MulticastSampledValueControlBlock	
UnicastSampledValueControlBlock	

6.6.45 E_AcsiMechHealth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiMechHealth :
(
    Ok      := 1,
    Warning := 2,
    Alarm   := 3
)SINT;
END_TYPE
```

Values

Name	Description
Ok	
Warning	
Alarm	

6.6.46 E_AcsiMod

This parameter controls the behavior of the logical node (e.g. for processing of received data).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiMod :
(
    On          := 1,
    OnBlocked   := 2,
```

```
    Test        := 3,
    TestBlocked := 4,
    Off         := 5
)SINT;
END_TYPE
```

Values

Name	Description
On	
OnBlocked	
Test	
TestBlocked	
Off	

6.6.47 E_AcsiMonth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiMonth :
(
{ attribute 'hide'}    reserved := 0,
    January     := 1,
    February    := 2,
    March       := 3,
    April       := 4,
    May         := 5,
    June        := 6,
    July        := 7,
    August      := 8,
    September   := 9,
    October     := 10,
    November    := 11,
    December    := 12
)SINT;
END_TYPE
```

6.6.48 E_AcsiMultiplier

Multiplier.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiMultiplier :
(
    Yocto  := -24,
    Zepto  := -21,
    Atto   := -18,
    Femto  := -15,
    Pico   := -12,
    Nano   :=  -9,
    Micro  :=  -6,
    Milli  :=  -3,
    Centi  :=  -2,
    Deci   :=  -1,
    None   :=   0,
    Deca   :=   1,
    Hecto  :=   2,
    Kilo   :=   3,
    Mega   :=   6,
    Giga   :=   9,
    Tera   :=  12,
    Petra  :=  15,
    Exa    :=  18,
    Zetta  :=  21,
```

```
    Yotta  :=  24
)SINT;
END_TYPE
```

Values

Name	Description
Yocto	Value: 10^(-24), symbol: y
Zepto	Value: 10^(-21), symbol: z
Atto	Value: 10^(-18), symbol: a
Femto	Value: 10^(-15), symbol: f
Pico	Value: 10^(-12), symbol: p
Nano	Value: 10^(-9), symbol: n
Micro	Value: 10^(-6), symbol: u
Milli	Value: 10^(-3), symbol: m
Centi	Value: 10^(-2), symbol: c
Deci	Value: 10^(-1), symbol: d
None	Value: 1
Deca	Value: 10^(1), symbol: da
Hecto	Value: 10^(2), symbol: h
Kilo	Value: 10^(3), symbol: k
Mega	Value: 10^(6), symbol: M
Giga	Value: 10^(9), symbol: G
Tera	Value: 10^(12), symbol: T
Petra	Value: 10^(15), symbol: P
Exa	Value: 10^(18), symbol: E
Zetta	Value: 10^(21), symbol: Z
Yotta	Value: 10^(24), symbol: Y

6.6.49 E_AcsiOccPer

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiOccPer :
(
    Hour  := 0,
    Day   := 1,
    Week  := 2,
    Month := 3,
    Year  := 4
)SINT;
END_TYPE
```

Values

Name	Description
Hour	Hour
Day	Tag
Week	Week
Month	Month
Year	Year

6.6.50 E_AcsiOccType

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiOccType :
(
    Time_      := 0,
    WeekDay    := 1,
    WeekOfYear := 2,
    DayOfMonth := 3,
    DayOfYear  := 4
)SINT;
END_TYPE
```

Values

Name	Description
Time_	
WeekDay	
WeekOfYear	
DayOfMonth	
DayOfYear	

6.6.51 E_AcsiOpModRect

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiOpModRect :
(
    CurrentControl     := 1,
    VoltageControl     := 2,
    ActivePowerControl := 3
)SINT;
END_TYPE
```

Values

Name	Description
CurrentControl	
VoltageControl	
ActivePowerControl	

6.6.52 E_AcsiOpModSyn

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiOpModSyn :
(
    AutomaticSynchronising := 1,
    AutomaticParalleling   := 2,
    Manual                 := 3,
    Test                   := 4
)SINT;
END_TYPE
```

Values

Name	Description
AutomaticSynchronising	
AutomaticParalleling	
Manual	
Test	

6.6.53 E_AcsiOrCategory

Command originator category.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiOrCategory :
(
    NotSupported     := 0,
    BayControl       := 1,
    StationControl   := 2,
    RemoteControl    := 3,
    AutomaticBay     := 4,
    AutomaticStation := 5,
    AutomaticRemote  := 6,
    Maintenance      := 7,
    Process          := 8
)SINT;
END_TYPE
```

Values

Name	Description
NotSupported	
BayControl	
StationControl	
RemoteControl	
AutomaticBay	
AutomaticStation	
AutomaticRemote	
Maintenance	
Process	

6.6.54 E_AcsiParColMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiParColMod :
(
    MasterSlave := 1,
    MasterSlaveWithFixedSlavePosition     := 2,
    MasterSlaveWithVariableSlavePosiiton  := 3,
    ParallelOperationWithoutCommunication := 4
)SINT;
END_TYPE
```

Values

Name	Description
MasterSlave	
MasterSlaveWithFixedSlavePosition	
MasterSlaveWithVariableSlavePosiiton	
ParallelOperationWithoutCommunication	

6.6.55 E_AcsiParMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiParMod :
(
    Master      := 1,
    Slave       := 2,
    Independent := 3
)SINT;
END_TYPE
```

Values

Name	Description
Master	
Slave	
Independent	

6.6.56 E_AcsiParTraMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiParTraMod :
(
    NoModePredefined           := 1,
    Master                     := 2,
    Follower                   := 3,
    PowerFactor                := 4,
    NegativeReactance          := 5,
    CirculatingCurrent         := 6,
    CirculatingReactiveCurrent := 7,
    CirculatingReactiveCurrentByEqualizingCalculatedTransformerPowerFactor := 8
)SINT;
END_TYPE
```

Values

Name	Description
NoModePredefined	
Master	
Follower	
PowerFactor	
NegativeReactance	
CirculatingCurrent	
CirculatingReactiveCurrent	
CirculatingReactiveCurrentByEqualizingCalculatedTransformerPowerFactor	

6.6.57 E_AcsiPFSign

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiPFSign :
(
    ActivePower := 1,
    LeadLag     := 2
)SINT;
END_TYPE
```

Values

Name	Description
ActivePower	
LeadLag	

6.6.58 E_AcsiPhsRef

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiPhsRef :
(
    A := 0,
    B := 1,
    C := 2
)SINT;
END_TYPE
```

Values

Name	Description
A	
B	
C	

6.6.59 E_AcsiPhyHealth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiPhyHealth :
(
    Ok      := 1,
    Warning := 2,
    Alarm   := 3
)SINT;
END_TYPE
```

Values

Name	Description
Ok	
Warning	
Alarm	

6.6.60 E_AcsiPIDAlg

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiPIDAlg :
(
    P   := 1,
    I   := 2,
    D   := 3,
    PI  := 4,
    PD  := 5,
    ID  := 6,
    PID := 7
)SINT;
END_TYPE
```

Values

Name	Description
P	
I	
D	
PI	
PD	
ID	
PID	

6.6.61 E_AcsiPmpCtl

Control value for pump control.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiPmpCtl :
(
    Inactive := 1,
    Stage1   := 2,
    Stage2   := 3,
    Stage3   := 4
)SINT;
END_TYPE
```

Values

Name	Description
Inactive	
Stage1	
Stage2	
Stage3	

6.6.62 E_AcsiPmpCtlGen

Generic control value for pump control.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiPmpCtlGen :
(
    Inactive := 1,
    Stage1   := 2,
    Stage2   := 3,
    Stage3   := 4
)SINT;
END_TYPE
```

Values

Name	Description
Inactive	
Stage1	
Stage2	
Stage3	

6.6.63 E_AcsiPolQty

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiPolQty :
(
    None                    := 1,
    ZeroSequenceCurrent     := 2,
    ZeroSequenceVoltage     := 3,
    NegativeSequenceVoltage := 4,
    PhaseToPhaseVoltages    := 5,
    PhaseToGroundVoltages   := 6,
    PositiveSequenceVoltage := 7
)SINT;
END_TYPE
```

Values

Name	Description
None	
ZeroSequenceCurrent	
ZeroSequenceVoltage	
NegativeSequenceVoltage	
PhaseToPhaseVoltages	
PhaseToGroundVoltages	
PositiveSequenceVoltage	

6.6.64 E_AcsiPOWCap

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiPOWCap :
(
    None         := 1,
    Close        := 2,
    Open         := 3,
    CloseAndOpen := 4
)SINT;
END_TYPE
```

Values

Name	Description
None	
Close	
Open	
CloseAndOpen	

6.6.65 E_AcsiQualitySource

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiQualitySource :
(
    Process     := 0,
    Substituted := 1
)SINT;
END_TYPE
```

Values

Name	Description
Process	Process
Substituted	Replaced

6.6.66 E_AcsiQualityValidity

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiQualityValidity :
(
    Good         := 0,
    Invalid      := 1,
    Reserved     := 2,
    Questionable := 3
)SINT;
END_TYPE
```

Values

Name	Description
Good	
Invalid	
Reserved	
Questionable	

6.6.67 E_AcsiRange

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiRange :
(
    Normal   := 0,
    High     := 1,
    Low      := 2,
```

```
    HighHigh := 3,
    LowLow   := 4
)SINT;
END_TYPE
```

Values

Name	Description
Normal	
High	
Low	
HighHigh	
LowLow	

6.6.68 E_AcsiRcdMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiRcdMod :
(
    OverwriteExistingValues := 1,
    StopWhenFullOrSaturated := 2
)SINT;
END_TYPE
```

Values

Name	Description
OverwriteExistingValues	
StopWhenFullOrSaturated	

6.6.69 E_AcsiReTrMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiReTrMod :
(
    Off                              := 1,
    WithoutCheck                     := 2,
    WithCurrentCheck                 := 3,
    WithBreakerStatusCheck           := 4,
    WithCurrentAndBreakerStatusCheck := 5,
    OtherChecks                      := 6
)SINT;
END_TYPE
```

Values

Name	Description
Off	
WithoutCheck	
WithCurrentCheck	
WithBreakerStatusCheck	
WithCurrentAndBreakerStatusCheck	
OtherChecks	

6.6.70 E_AcsiRotDir

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiRotDir :
(
    Clockwise        := 1,
    CounterClockwise := 2,
    Unknown          := 3
)SINT;
END_TYPE
```

Values

Name	Description
Clockwise	
CounterClockwise	
Unknown	

6.6.71 E_AcsiRstMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiRstMod :
(
    None                                    := 1,
    Harmonic2                               := 2,
    Harmonic5                               := 3,
    Harmonic2And5                           := 4,
    WaveformAnalysis                        := 5,
    WaveformAnalysisAndHarmonic2            := 6,
    Other                                   := 7,
    WaveformAnalysisAndHarmonic5            := 8,
    WaveformAnalysisAndHarmonic2AndHarmonic5 := 9
)SINT;
END_TYPE
```

Values

Name	Description
None	
Harmonic2	
Harmonic5	
Harmonic2And5	
WaveformAnalysis	
WaveformAnalysisAndHarmonic2	
Other	
WaveformAnalysisAndHarmonic5	
WaveformAnalysisAndHarmonic2AndHarmonic5	

6.6.72 E_AcsiSamplingMode

Defines the sampling mode for measured values.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiSamplingMode :
(
    SamplesPerPeriod:=0,
    SamplesPerSecond:=1,
    SecondsPerSample:=2
)SINT;
END_TYPE
```

Values

Name	Description
SamplesPerPeriod	Samples per nominal period (default).
SamplesPerSecond	Samples per second (defines the appropriate sample number per second).
SecondsPerSample	Seconds per sample (seconds between samples).

6.6.73 E_AcsiSboClasses

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiSboClasses :
(
    OperateOnce := 0,
    OperateMany := 1
)SINT;
END_TYPE
```

Values

Name	Description
OperateOnce	Operate-once.
OperateMany	Operate-many.

6.6.74 E_AcsiSeqT

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiSeqT :
(
    PosNegZero  := 0,
    DirQuadZero := 1
)SINT;
END_TYPE
```

Values

Name	Description
PosNegZero	
DirQuadZero	

6.6.75 E_AcsiServerDirectoryClass

Object classes of a server.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiServerDirectoryClass :
(
    LogicalDevice := 1,
    File          := 2
)SINT;
END_TYPE
```

Values

Name	Description
LogicalDevice	Logical device
File	File

6.6.76 E_AcsiServiceError

Error codes that are reported in the service feedback event handling routine. When using MMS mapping, the MMS error codes are also converted (mapped) to these error codes.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiServiceError :
(
    NoError                              := 0,
    InstanceNotAvailable                 := 1,
    InstanceInUse                        := 2,
    AccessViolation                      := 3,
    AccessNotAllowedInCurrentState       := 4,
    ParameterValueInappropriate          := 5,
    ParameterValueInconsistent           := 6,
    ClassNotSupported                    := 7,
    InstanceLockedByOtherClient          := 8,
    ControlMustBeSelected                := 9,
    TypeConflict                         := 10,
    FailedDueToCommunicationsConstraint  := 11,
    FailedDueToServerConstraint          := 12
)SINT;
END_TYPE
```

Values

Name	Description
NoError	
InstanceNotAvailable	
InstanceInUse	
AccessViolation	
AccessNotAllowedInCurrentState	
ParameterValueInappropriate	
ParameterValueInconsistent	
ClassNotSupported	
InstanceLockedByOtherClient	
ControlMustBeSelected	
TypeConflict	
FailedDueToCommunicationsConstraint	
FailedDueToServerConstraint	

6.6.77 E_AcsiServiceType

Service type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiServiceType :
(
    Unknown                   := 0,
    Associate                 := 1,
    Abort                     := 2,
    Release                   := 3,
    GetServerDirectory        := 4,
    GetLogicalDeviceDirectory := 5,
    GetAllDataValues          := 6,
    GetDataValues             := 7,
    SetDataValues             := 8,
    GetDataDirectory          := 9,
    GetDataDefinition         := 10,
    GetDataSetValues          := 11,
    SetDataSetValues          := 12,
    CreateDataSet             := 13,
    DeleteDataSet             := 14,
    GetDataSetDirectory       := 15,
    SelectActiveSG            := 16,
    SelectEditSG              := 17,
    SetEditSGValue            := 18,
    ConfirmEditSGValues       := 19,
    GetEditSGValue            := 20,
    GetSGCBValues             := 21,
    Report                    := 22,
    GetBRCBValues             := 23,
    SetBRCBValues             := 24,
    GetURCBValues             := 25,
    SetURCBValues             := 26,
    GetLCBValues              := 27,
    SetLCBValues              := 28,
    QueryLogByTime            := 29,
    QueryLogAfter             := 30,
    GetLogStatusValues        := 31,
    SendGOOSEMessage          := 32,
    GetGoCBValues             := 33,
    SetGoCBValues             := 34,
    GetGoReference            := 35,
    GetGOOSEElementNumber     := 36,
    SendMSVMessage            := 37,
    GetMSVCBValues            := 38,
    SetMSVCBValues            := 39,
    SendUSVMessage            := 40,
    GetUSVCBValues            := 41,
    SetUSVCBValues            := 42,
    Select                    := 43,
    SelectWithValue           := 44,
    Cancel                    := 45,
    Operate                   := 46,
    CommandTermination        := 47,
    TimeActivatedOperate      := 48,
    GetFile                   := 49,
    SetFile                   := 50,
    DeleteFile                := 51,
    GetFileAttributeValues    := 52,
    TimeSynchronization       := 53,
    InternalChangeUnknown     := 54,
    (* TwinCAT specific *)
    GetLogicalNodeDirectory   := 100,
    GetAllServerValues        := 101,
    GetAllLogicalDeviceValues := 102,
    SetAllDataValues          := 103
)SINT;
END_TYPE
```

6.6.78 E_AcsiSetCharact

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiSetCharact :
(
    AnsiExtremlyInverse := 1,
    AnsiVeryInverse := 2,
    AnsiNormalInverse := 3,
    AnsiModerateInverse := 4,
    AnsiDefiniteTime := 5,
    LongTimeExtermelyInverse := 6,
    LongTimeVeryInverse := 7,
    LongTimeInverse := 8,
    IecNormalInverse := 9,
    IecVeryInverse := 10,
    IecInverse := 11,
    IecExtremelyInverse := 12,
    IecShortTimeInverse := 13,
    IecLongTimeInverse := 14,
    IecDefiniteTime := 15,
{ attribute 'hide'}    reserved := 16,
    Polynom1 := 17,
    Polynom2 := 18,
    Polynom3 := 19,
    Polynom4 := 20,
    Polynom5 := 21,
    Polynom6 := 22,
    Polynom7 := 23,
    Polynom8 := 24,
    Polynom9 := 25,
    Polynom10 := 26,
    Polynom11 := 27,
    Polynom12 := 28,
    Polynom13 := 29,
    Polynom14 := 30,
    Polynom15 := 31,
    Polynom16 := 32,
    Multiline1 := 33,
    Multiline2 := 34,
    Multiline3 := 35,
    Multiline4 := 36,
    Multiline5 := 37,
    Multiline6 := 38,
    Multiline7 := 39,
    Multiline8 := 40,
    Multiline9 := 41,
    Multiline10 := 42,
    Multiline11 := 43,
    Multiline12 := 44,
    Multiline13 := 45,
    Multiline14 := 46,
    Multiline15 := 47,
    Multiline16 := 48
)SINT;
END_TYPE
```

6.6.79 E_AcsiSev

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiSev :
(
    Unknown  := 0,
    Critical := 1,
    Major    := 2,
    Minor    := 3,
    Warning  := 4
)SINT;
END_TYPE
```

Values

Name	Description
Unknown	
Critical	
Major	
Minor	
Warning	

6.6.80 E_AcsiShOpCap

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiShOpCap :
(
    None         := 1,
    Open         := 2,
    Close        := 3,
    OpenAndClose := 4
)SINT;
END_TYPE
```

Values

Name	Description
None	
Open	
Close	
OpenAndClose	

6.6.81 E_AcsiSIUnit

SI unit.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiSIUnit :
(
(* Base units: *)
    None                    :=1, (* Dimensionless *)
    Meter                   :=2, (* Length m *)
    Kilogram                :=3, (* Mass kg *)
    Second                  :=4, (* Time s *)
    Ampere                  :=5, (* Current A *)
    Kelvin                  :=6, (* Temperature K *)
    Mole                    :=7, (* Amount of substance mol *)
    Candela                 :=8, (* Luminous intensity cd *)

(*Derived units: *)
    Degrees                 :=9,  (* Plane angle deg *)
    Radian                  :=10, (* Plane angle rad *)
    Steradian               :=11, (* Solid angle sr *)
    Gray                    :=21, (* Absorbed dose (J/kg) Gy *)
    Becquerel               :=22, (* Activity (1/s) q *)
    DegreesCelsius          :=23, (* Relative temperature °C *)
    Sievert                 :=24, (* Dose equivalent (J/kg) Sv *)
    Farad                   :=25, (* Electric capacitance (C/V) F *)
    Coulomb                 :=26, (* Electric charge (As) C *)
    Siemens                 :=27, (* Electric conductance (A/V) S *)
    Henry                   :=28, (* Electric inductance (Wb/A) H *)
    Volt                    :=29, (* Electric potential (W/A) V *)
    Ohm                     :=30, (* Electric resistance (VA) *)
```

```
    Joule                   :=31, (* Energy (Nm) J *)
    Newtown                 :=32, (* Force (kg m/s²) N *)
    Hertz                   :=33, (* Frequency (1/s) Hz *)
    Lux                     :=34, (* Illuminance (lm/m²) lx *)
    Lumen                   :=35, (* Luminous flux (cd sr) Lm *)
    Weber                   :=36, (* Magnetic flux (Vs) Wb *)
    Tesla                   :=37, (* Magnetic flux density (Wb/m²) T *)
    Watt                    :=38, (* Power (J/s) W *)
    Pascal                  :=39, (* Pressure (N/m²) Pa *)

(* Extended units: *)
    SquareMeter             :=41, (* Area (m²) m² *)
    CubicMeter              :=42, (* Volume (m³) m³ *)
    MetersPerSecond         :=43, (* Velocity (m/s) m/s *)
    MetersPerSecondSquare   :=44, (* Acceleration (m/s²) m/s² *)
    CubicMetersPerSecond    :=45, (* Volumetric flow rate (m³/s) m³/s *)
    MetersPerCubicMeter     :=46, (* Fuel efficiency (m/m³) m/m³ *)
    KilogramMeter           :=47, (* Moment of mass (kg m) M *)
    KilogramPerCubicMeter   :=48, (* Density (kg/m³) kg/m³ *)
    MeterSquarePerSecond    :=49, (* Viscosity (m²/s) m²/s *)
    WattPerMeterKelvin      :=50, (* Thermal conductivity (W/m K) W/m K *)
    JoulePerKelvin          :=51, (* Heat capacity (J/K) J/K *)
    PartsPerMillion         :=52, (* Concentration ppm *)
    RotationsPerSecond      :=53, (* Rotational speed (1/s) 1/s *)
    RadianPerSecond         :=54, (* Angular velocity (rad/s) rad/s *)

(* Industry specific units: *)
    VoltAmpere              :=61, (* Apparent power (VA) VA *)
    Watts                   :=62, (* Real power (I²R) W *)
    VoltAmpereReactive      :=63, (* Reactive power (VISin) VAr *)
    DegreesPh               :=64, (* Phase angle *)
    DimensionlessPF         :=65, (* Power factor Cos *)
    VoltSeconds             :=66, (* Volt seconds (Ws/A) Vs *)
    VoltSquare              :=67, (* Volts squared (W²/A²) V² *)
    AmpSecond               :=68, (* Amp seconds (As) As *)
    AmpSquare               :=69, (* Amps squared (A²) A² *)
    AmpSquareSecond         :=70, (* Amps squared time (A²s) A²t *)
    VoltAmpereHours         :=71, (* Apparent energy VAh *)
    WattHours               :=72, (* Real energy Wh *)
    VoltAmpereReactiveHours :=73, (* Reactive energy VArh *)
    VoltPerHertz            :=74  (* Magnetic flux V/Hz *)
)SINT;
END_TYPE
```

Values

Name	Description
None	Dimensionless
Meter	Length m
Kilogram	Mass kg
Second	Time s
Ampere	Current A
Kelvin	Temperature K
Mole	Amount of substance mol

6.6.82 E_AcsiSptEndSt

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiSptEndSt :
(
```

```
    EndedNormally                              := 1,
    EndedWithOvershoot                         := 2,
    MeasurementWasDeviating                    := 3,
    LossOfCommunicationWithDispatchCentre      := 4,
    LossOfCommunicationWithLocalAreaNetwork    := 5,
    LossOfCommunicationWithTheLocalInterface   := 6,
    Timeout                                    := 7,
    Voluntarily                                := 8,
    NoisyEnvironments                          := 9,
    MaterialFailure                            := 10,
    NewSetpointRequest                         := 11,
    ImproperEnvironment                        := 12,
    StabilityTimeWasReached                    := 13,
    ImmobilisationTimeWasReached               := 14,
    EquipmentInWrongMode                       := 15,
    Unknown                                    := 16
)SINT;
END_TYPE
```

Values

Name	Description
EndedNormally	
EndedWithOvershoot	
MeasurementWasDeviating	
LossOfCommunicationWithDispatchCentre	
LossOfCommunicationWithLocalAreaNetwork	
LossOfCommunicationWithTheLocalInterface	
Timeout	
Voluntarily	
NoisyEnvironments	
MaterialFailure	
NewSetpointRequest	
ImproperEnvironment	
StabilityTimeWasReached	
ImmobilisationTimeWasReached	
EquipmentInWrongMode	
Unknown	

6.6.83 E_AcsiStClcTun

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiStClcTun :
(
    NotTuned                                        := 1,
    Tuned                                           := 2,
    TunedButNotCompensated                          := 3,
    Umax                                            := 4,
    UmaxButNotCompensated                           := 5,
    UmaxButNotCompensatedDueToUContinousLimitation  := 6
)SINT;
END_TYPE
```

Values

Name	Description
NotTuned	
Tuned	
TunedButNotCompensated	
Umax	
UmaxButNotCompensated	
UmaxButNotCompensatedDueToUContinousLimitation	

6.6.84 E_AcsiStrWeekDay

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiStrWeekDay :
(
    Monday    := 1,
    Tuesday   := 2,
    Wednesday := 3,
    Thursday  := 4,
    Friday    := 5,
    Saturday  := 6,
    Sunday    := 7
)SINT;
END_TYPE
```

Values

Name	Description
Monday	Monday
Tuesday	Tuesday
Wednesday	Wednesday
Thursday	Thursday
Friday	Friday
Saturday	Saturday
Sunday	Sunday

6.6.85 E_AcsiSwOpCap

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiSwOpCap :
(
    None         := 1,
    Open         := 2,
    Close        := 3,
    OpenAndClose := 4
)SINT;
END_TYPE
```

Values

Name	Description
None	
Open	
Close	
OpenAndClose	

6.6.86 E_AcsiSwTyp

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiSwTyp :
(
    LoadBreakSwitch         := 1,
    Disconnector            := 2,
    EarthingSwitch          := 3,
    HighSpeedEarthingSwitch := 4
)SINT;
END_TYPE
```

Values

Name	Description
LoadBreakSwitch	
Disconnector	
EarthingSwitch	
HighSpeedEarthingSwitch	

6.6.87 E_AcsiTcmd

Control value for a step switch. The enumeration value is coded in 2 bits. The remaining 6 bits are not used.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiTcmd :
(
    Stop     := 2#00,
    Lower    := 2#01,
    Higher   := 2#10,
    {attribute 'hide'}reserved := 2#11
)SINT;
END_TYPE
```

Values

Name	Description
Stop	Stoppen
Lower	Decrement
Higher	Increment
reserved	Reserved

6.6.88 E_AcsiTmSyn

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiTmSyn :
(
    NotSynchronized                     := 0,
    SynchronizedByLocalAreaClockSignal  := 1,
    SynchronizedByGlobalAreaClockSignal := 2
)SINT;
END_TYPE
```

Values

Name	Description
NotSynchronized	
SynchronizedByLocalAreaClockSignal	
SynchronizedByGlobalAreaClockSignal	

6.6.89 E_AcsiTnkTyp

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiTnkTyp :
(
    PressureOnly         := 1,
    LevelOnly            := 2,
    BothPressureAndLevel := 3
)SINT;
END_TYPE
```

Values

Name	Description
PressureOnly	
LevelOnly	
BothPressureAndLevel	

6.6.90 E_AcsiTpcRxMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiTpcRxMod :
(
    Unused       := 1,
    Blocking     := 2,
    Permissive   := 3,
    Direct       := 4,
    Unblocking   := 5,
    Status       := 6
)SINT;
END_TYPE
```

Values

Name	Description
Unused	
Blocking	
Permissive	
Direct	
Unblocking	
Status	

6.6.91 E_AcsiTpcTxMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiTpcTxMod :
(
    Unused     := 1,
    Blocking   := 2,
    Permissive := 3,
    Direct     := 4,
    Unblocking := 5,
    Status     := 6
)SINT;
END_TYPE
```

Values

Name	Description
Unused	
Blocking	
Permissive	
Direct	
Unblocking	
Status	

6.6.92 E_AcsiTrBeh

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiTrBeh :
(
    SinglePoleTripping := 1,
    Undefined          := 2,
    ThreePoleTripping  := 3
)SINT;
END_TYPE
```

Values

Name	Description
SinglePoleTripping	
Undefined	
ThreePoleTripping	

6.6.93 E_AcsiTrgMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiTrgMod :
(
    Internal_ := 1,
    External  := 2,
    Both      := 3
)SINT;
END_TYPE
```

Values

Name	Description
Internal_	
External	
Both	

6.6.94 E_AcsiTrMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiTrMod :
(
    ThreePhaseTripping      := 1,
    OneOrThreePhaseTripping := 2,
    Specific                := 3,
    OnePhaseTripping        := 4
)SINT;
END_TYPE
```

Values

Name	Description
ThreePhaseTripping	
OneOrThreePhaseTripping	
Specific	
OnePhaseTripping	

6.6.95 E_AcsiTypRsCrv

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiTypRsCrv :
(
    None                     := 1,
    DefiniteTimeDelayedReset := 2,
    InverseReset             := 3
)SINT;
END_TYPE
```

Values

Name	Description
None	
DefiniteTimeDelayedReset	
InverseReset	

6.6.96 E_AcsiUnbDetMth

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiUnbDetMth :
(
    NegativeSequence                  := 1,
    ZeroSequence                      := 2,
    NegativeSequencePositiveSequence  := 3,
    ZeroSequencePositiveSequenceDirect := 4,
    PhaseVectorsComparison            := 5,
    Others                            := 6
)SINT;
END_TYPE
```

Values

Name	Description
NegativeSequence	
ZeroSequence	
NegativeSequencePositiveSequence	
ZeroSequencePositiveSequenceDirect	
PhaseVectorsComparison	
Others	

6.6.97 E_AcsiUnBlkMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiUnBlkMod :
(
    Off        := 1,
    Permanent  := 2,
    TimeWindow := 3
)SINT;
END_TYPE
```

Values

Name	Description
Off	
Permanent	
TimeWindow	

6.6.98 E_AcsiVLanPriority

Virtual LAN tag control information: priority code point.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiVLanPriority :
(
    BK := 0,
    BE := 1,
    EE := 2,
    CA := 3,
    VI := 4,
    VO := 5,
    IC := 6,
    NC := 7
)BYTE;
END_TYPE
```

Values

Name	Description
BK	Background (0).
BE	Best Effort (1).
EE	Excellent Effort (2).
CA	Critical Applications (3)
VI	Video (4), < 100 ms delay.
VO	Voice (5), < 10 ms delay.
IC	Internetwork Control (6).
NC	Network Control (7), highest.

6.6.99 E_AcsiWeekDay

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiWeekDay :
(
{ attribute 'hide'}    reserved := 0,
    Monday    := 1,
    Tuesday   := 2,
    Wednesday := 3,
    Thursday  := 4,
    Friday    := 5,
    Saturday  := 6,
    Sunday    := 7
)SINT;
END_TYPE
```

Values

Name	Description
reserved	Reserved
Monday	Monday
Tuesday	Tuesday
Wednesday	Wednesday
Thursday	Thursday
Friday	Friday
Saturday	Saturday
Sunday	Sunday

6.6.100 E_AcsiWeiMod

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AcsiWeiMod :
(
    Off            := 1,
    Operate        := 2,
    Echo           := 3,
    EchoAndOperate := 4
)SINT;
END_TYPE
```

Values

Name	Description
Off	
Operate	
Echo	
EchoAndOperate	

6.6.101 E_AsyncEnvironmentState

Environmental condition of the communication link.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AsyncEnvironmentState :
(
    Idle               := 0,
    OutgoingConnect    := 1,
    IncomingConnect    := 2,
    Established        := 3,
    OutgoingDisconnect := 4,
    IncomingDisconnect := 5,
    Aborting           := 6
)BYTE;
END_TYPE
```

Values

Name	Description
Idle	Idle state. Connection is closed.
OutgoingConnect	An outgoing (client) connection is being established.
IncomingConnect	An incoming (server) connection is being established.
Established	Connection has been established.
OutgoingDisconnect	A (client) connection is being released.
IncomingDisconnect	A (server) connection is being released.
Aborting	The connection is being released.

6.6.102 E_AsyncServiceState

Asynchronous service status.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_AsyncServiceState :
(
    Completed := 0,
    Busy      := 1,
    Failed    := 2,
    Aborted   := 3
)BYTE;
END_TYPE
```

Values

Name	Description
Completed	Service was executed without error.
Busy	Service is currently being executed.
Failed	Service was terminated with an error.
Aborted	Service was aborted by the Abort-Request command from the user application or Abort-Indication from the communication layer.

6.6.103 E_EventCompletion

Execution status of the event handling routine.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_EventCompletion :
(
    Completed := 0,
    Pending   := 1,
    Idle      := 2
)BYTE;
END_TYPE
```

Values

Name	Description
Completed	Executed (completed). Not waiting for the response.
Pending	Execution still pending and waiting for response.
Idle	The event handling routine is not implemented.

6.6.104 E_GseDispatchMode

GSE network adapter receiving mode. This determines whether the frames received should also be sent to the operating system for further processing.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_GseDispatchMode :
(
    NonPromiscuous := 0,
    Monitor        := 1
)BYTE;
END_TYPE
```

Values

Name	Description
NonPromiscuous	Frames that are addressed to the GSE network adapter and processed (including Broadcast and Multicast frames) are no longer sent to the operating system for further processing.
Monitor	All frames received are sent to the operating system for further processing.

6.6.105 E_GseDispatchMode

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_GseDispatchMode :
(
    NonPromiscuous:= 0,
    Monitor       := 1
)BYTE;
END_TYPE
```

Values

Name	Description
NonPromiscuous	Received Ethernet telegrams are not passed on to the operating system.
Monitor	Received Ethernet telegrams are passed on to the operating system and can be processed further.

6.6.106 E_GseGoCBPublisherState

GOOSE control block publisher status.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_GseGoCBPublisherState :
(
    NonExistent       := 0,
    SendValues        := 1,
    RetransmitPending := 2,
    Retransmit        := 3
)BYTE;
END_TYPE
```

Values

Name	Description
NonExistent	Disabled.
SendValues	Active and sending GOOSE message (including update).
RetransmitPending	Active and waiting for GOOSE message repetition.
Retransmit	Active and sending GOOSE message repetition.

6.6.107 E_GseGoCBSubscriberState

GOOSE control block subscriber status.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_GseGoCBSubscriberState :
(
    Off          := 0,
    NonExistent  := 1,
    Valid        := 2,
    Questionable := 3
)BYTE;
END_TYPE
```

Values

Name	Description
Off	Disabled.
NonExistent	Active and has not yet received any GOOSE messages.
Valid	Active and is continually receiving GOOSE messages.
Questionable	Active and the GOOSE messages are either not being received at all or too late (allowed reading time/validity duration has expired).

6.6.108 E_GseLinkStatus

GSE network adapter link status.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_GseLinkStatus :
(
    Ok        := 0,
    LinkError := 1
);
END_TYPE
```

Values

Name	Description
Ok	The link has been established.
LinkError	The link has been interrupted.

6.6.109 E_GseRetransmissionStrategy

Time algorithm for the repeated GOOSE messages.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_GseRetransmissionStrategy :
(
    Arithmetic:=0
)BYTE;
END_TYPE
```

Values

Name	Description
Arithmetic	The first message is sent after the minimum interval time. Each subsequent message (repetition) is sent after the time interval, which is a product of the previous interval time and the multiplier, until the maximum time is reached. If the value changes, a GOOSE message is sent immediately and the interval time is reset to the minimum value again.

6.6.110 E_GseSvCBState

SV (Sampled Values) control block state.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_GseSvCBState :
(
    Disabled:=0,
    Enabled:=1
)USINT;
END_TYPE
```

Values

Name	Description
Disabled	SV Publisher/Subscriber is deactivated.
Enabled	SV Publisher/Subscriber is activated.

6.6.111 E_GseSvGeneratorMode

Configures the sampling mode of the sine wave signal generator.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_GseSvGeneratorMode :
(
    AtSmpRatePerPeriod:=0,
    AtSmpRatePerSecond:=1,
    AtPlcTaskCycleTime:=2
)SINT:=AtPlcTaskCycleTime;
END_TYPE
```

Values

Name	Description
AtSmpRatePerPeriod	Generates samples at a specified sampling rate per period.
AtSmpRatePerSecond	Generates samples at a specified sampling rate per second.
AtPlcTaskCycleTime	The sample values are generated during the PLC task cycle. For the oversampling factor: 1, one sample value is generated per PLC cycle, and with an oversampling factor of n, n sample values are generated per PLC cycle (the signal generator's call method is called n times in each cycle).

6.6.112 E_GseSvSmpSynch

SV (Sampled Values) synchronization type.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_GseSvSmpSynch :
(
    None:=0,
    Local:=1,
    Global:=2
)USINT;
END_TYPE
```

Values

Name	Description
None	No synchronization.
Local	Synchronized locally.
Global	Synchronized globally.

6.6.113 E_MmsVmdLogicalStatus

MMS Type: vmdLogicalStatus

Namespace: Tc3_Mms [▶ 102]
Library: Tc3_Mms (Tc3_Mms.compiled-library)

```
{attribute 'qualified_only'}
TYPE E_MmsVmdLogicalStatus :
(
    StateChangesAllowed      :=0,
    NoStateChangesAllowed    :=1,
    LimitedServicesPermitted :=2,
    SupportServicesAllowed   :=3
)BYTE;
END_TYPE
```

6.6.114 E_MmsVmdPhysicalStatus

MMS Type: vmdPhysicalStatus.

Namespace: Tc3_Mms [▶ 102]
Library: Tc3_Mms (Tc3_Mms.compiled-library)

```
{attribute 'qualified_only'}
TYPE E_MmsVmdPhysicalStatus :
(
    Operational          :=0,
    PartiallyOperational :=1,
    Inoperable           :=2,
    NeedsCommissioning   :=3
)BYTE;
END_TYPE
```

6.6.115 E_Rfc1006TpduSize

Maximum TPDU byte length.

Namespace: Tc3_Rfc1006 [▶ 102]
Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

```
TYPE E_Rfc1006TpduSize :
(
    Size_0    := 0,
    Size_128  := 7,
    Size_256  := 8,
    Size_512  := 9,
    Size_1024 := 10,
    Size_2048 := 11,
```

```
    Size_4096 := 12,
    Size_8192 := 13,
    Size_16384:= 14,
    Size_32768:= 15
)BYTE;
END_TYPE
```

Values

Name	Description
Size_0	Not specified.
Size_128	128 Byte (default).
Size_256	256 bytes.
Size_512	512 bytes.
Size_1024	1024 bytes.
Size_2048	2048 bytes.
Size_4096	4096 bytes.
Size_8192	8192 bytes.
Size_16384	16384 bytes.
Size_32768	32768 bytes.

6.6.116 E_ScsmEdition

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_ScsmEdition :
(
    Ed1 := 1,
    Ed2 := 2
)BYTE := Ed2;
END_TYPE
```

Values

Name	Description
Ed1	EN 61850-8-1:2005
Ed2	EN 61850-8-1:2011

6.6.117 E_ScsmLinkStatus

Link status of the network adapter (sockets). See: E_TcpLinkStatus [▶ 526].

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
ALIAS E_ScsmLinkStatus : E_TcpLinkStatus
```

6.6.118 E_ScsmRCBState

RCB (Report Control Block) status information.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
TYPE E_ScsmRCBState :
(
    Disabled:=0,
    Reserved:=1,
    Enabled:=2,
```

```
    Resync:=3
)SINT;
END_TYPE
```

6.6.119 E_TcpLinkStatus

Link status of the network adapter.

Namespace: Tc3_Sockets [▶ 102]
Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_TcpLinkStatus :
(
    Unknown   :=-1,
    Ok        :=0,
    LinkError :=1
)SINT:=Unknown;
END_TYPE
```

Values

Name	Description
Unknown	Link status unknown or not yet determined
Ok	Connection established
LinkError	Link error or no connection

6.6.120 E_TcpSocketType

Socket type.

Namespace: Tc3_Sockets [▶ 102]
Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_TcpSocketType :
(
    None     :=0,
    Listener :=1,
    Accepted :=2,
    Client   :=3
)BYTE;
END_TYPE
```

Values

Name	Description
None	Invalid socket (unopened, closed socket)
Listener	The Listener socket of the server
Accepted	Accepted server connection socket
Client	Client connection socket

6.6.121 E_TraceLevel

Priority level for logging the error messages and log messages.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_TraceLevel :
(
    None    := 0,// don't trace any message
```

```
    Verbose := 16#00000001,// bit 0 => log verbose messages
    TxData  := 16#00000002,// bit 1 => log tx data
    RxData  := 16#00000004,// bit 2 => log rx data
    Info    := 16#00000100,// bit 8..15 => trace pdu info messages
    PduReq  := 16#00000200,
    PduInd  := 16#00000400,
    PduRsp  := 16#00000800,
    PduCnf  := 16#00001000,
    Warning := 16#00010000,// bit 16..23 => trace warning messages
    Error   := 16#01000000,// bit 24..30 => trace error messages
    Critical:= 16#80000000// bit 31 => trace critical messages
)DWORD;
END_TYPE
```

Values

Name	Description
None	None
Verbose	Text
TxData	Transmitted data
RxData	Received data
Info	Information
PduReq	PDU-Request
PduInd	PDU-Indication
PduRsp	PDU-Response
PduCnf	PDU-Confirmation
Warning	Warning
Error	Error
Critical	Critical

6.6.122 E_UtcTimeAccuracy

Number of significant bits in the member variable: T_UtcTime [▶ 560].fractionOfSecond. Values 25 to 30 are not used.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_UtcTimeAccuracy :
(
    Null        :=  0,
    _01         :=  1,
    _02         :=  2,
    _03         :=  3,
    _04         :=  4,
    _05         :=  5,
    _06         :=  6,
    _07         :=  7,
    _08         :=  8,
    _09         :=  9,
    _10         := 10,
    _11         := 11,
    _12         := 12,
    _13         := 13,
    _14         := 14,
    _15         := 15,
    _16         := 16,
    _17         := 17,
    _18         := 18,
    _19         := 19,
    _20         := 20,
    _21         := 21,
    _22         := 22,
    _23         := 23,
    _24         := 24,
    _25_invalid := 2#11001,
```

```
    _26_invalid := 2#11010,
    _27_invalid := 2#11011,
    _28_invalid := 2#11100,
    _29_invalid := 2#11101,
    _30_invalid := 2#11110,
    Unspecified := 31
)BYTE;
END_TYPE
```

Values

Name	Description
Zero	0 bits
_01	1-bit
_02	2-bit
_03	3-bit
_04	4-bit
_05	5-bit
_06	6-bit
_07	7-bit
_08	8-bit
_09	9-bit
_10	10-bit
_11	11-bit
_12	12-bit
_13	13-bit
_14	14-bit
_15	15-bit
_16	16-bit
_17	17-bit
_18	18-bit
_19	19-bit
_20	20-bit
_21	21-bit
_22	22-bit
_23	23-bit
_24	24-bit
_25_invalid.._30_invalid	Invalid
Unspecified	Not specified

6.6.123 External data types

6.6.123.1 ETHERNET_ADDRESS

Network adapter MAC address.

Namespace: TwinCAT TypeSystem [▶ 101]
Library: Not required. TwinCAT system resource from v3.1.4024.0

```
TYPE ETHERNET_ADDRESS :
    STRUCT
        b : ARRAY [0..5] OF BYTE;
    END_STRUCT
END_TYPE
```

6.6.123.2 GUID

System ID.

Namespace: TwinCAT TypeSystem [▶ 101]
Library: Not required. TwinCAT system resource from v3.1.4024.0

```
TYPE GUID :
STRUCT
    Data1 : DWORD;
    Data2 : WORD;
    Data3 : WORD;
    Data4 : ARRAY[0..7] OF BYTE;
END_STRUCT
END_TYPE
```

6.6.123.3 OTCID

Object ID of the TwinCAT real-time network adapter.

Namespace: TwinCAT TypeSystem [▶ 101]
Library: Not required. TwinCAT system resource from v3.1.4024.0

```
TYPE OTCID : UDINT;
END_TYPE
```

6.6.123.4 T_AmsNetID

A PLC variable of this type is a string containing the AMS network ID of the target device to which the ADS command is directed. The string consists of six numerical fields, separated by dots. Valid AMS network addresses are, for example, '1.1.1.2.7.1' or '200.5.7.170.1.7'. If an empty string is passed, the AMS network ID of the local device is automatically assumed.

Namespace: Tc2_System
Library: Tc2_System (Tc2_System.compiled-library)

```
TYPE T_AmsNetID : STRING(23);
END_TYPE
```

6.6.123.5 T_HSOCKET

TCP/IP socket connection handle.

Namespace: Tc2_TcpIp
Library: Tc2_TcpIp (Tc2_TcpIp.compiled-library)

```
TYPE T_HSOCKET
```

6.6.123.6 T_IPv4Addr

A variable of this type is a string with the (Ipv4) Internet protocol network address. E.g. '172.16.7.199'.

```
TYPE T_IPv4Addr : STRING(15);
END_TYPE
```

Prerequisites

Development Environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, Arm®)	Tc2_System (system)

6.6.123.7 T_MaxString

The variable of this type is PLC string with the maximal length. Longer strings are allowed, but the string functions are limited to 255 characters.

```
TYPE T_MaxString : STRING(MAX_STRING_LENGTH);
END_TYPE
```

```
VAR_GLOBAL CONSTANT
    MAX_STRING_LENGTH : UDINT := 255;
ENd_VAR
```

Prerequisites

Development Environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, Arm®)	Tc2_System (system)

6.6.123.8 T_ThrottleTimes

Cycle times for the receive data throttle polling mode.

Namespace: Tc2_TcpIp
Library: Tc2_TcpIp (Tc2_TcpIp.compiled-library)

```
TYPE T_ThrottleTimes
```

6.6.123.9 TcEthernetAdapterPriority

Priority ID of the TwinCAT real-time network adapter.

Namespace: TwinCAT TypeSystem [▶ 101]
Library: Not required. TwinCAT system resource from v3.1.4024.0

```
TYPE TcEthernetAdapterPriority :
(
    TcEthernetAdapterPriorityLow  :=1,
    TcEthernetAdapterPriorityHigh :=16
);
END_TYPE
```

6.6.123.10 TcEventEntry

Structured type with event or error information.

Namespace: TwinCAT TypeSystem [▶ 101]
Library: Not required. TwinCAT system resource from v3.1.4024.0

```
TYPE TcEventEntry :
STRUCT
        uuidEventClass : GUID;
        nEventId       : UDINT;
        eSeverity      : TcEventSeverity;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
uuidEventClass	GUID [▶ 529]	A unique ID for the event class. The GUID [▶ 529] specifies the event source.
nEventID	UDINT	Event ID or error number.
eSeverity	TcEventSeverity [▶ 530]	Degree of severity.

6.6.123.11 TcEventSeverity

Severity of a message.

Namespace: TwinCAT TypeSystem [▶ 101]
Library: Not required. TwinCAT system resource from v3.1.4024.0

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE TcEventSeverity :
(
    Verbose  := 0,
    Info     := 1,
    Warning  := 2,
    Error    := 3,
    Critical := 4
);
END_TYPE
```

Values

Name	Description
Verbose	Text
Info	Information
Warning	Warning
Error	Error
Critical	Critical

6.6.123.12 TIMESTRUCT

Time in system time format.

```
TYPE TIMESTRUCT
STRUCT
   wYear         : WORD;
   wMonth        : WORD;
   wDayOfWeek    : WORD;
   wDay          : WORD;
   wHour         : WORD;
   wMinute       : WORD;
   wSecond       : WORD;
   wMilliseconds : WORD;
END_STRUCT
END_TYPE
```

Value	Meaning
wYear	The year: 1970 ~ 2106
wMonth	The month: 1 ~ 12 (January = 1, February = 2 etc.)
wDayOfWeek	The day of the week: 0 ~ 6 (Sunday = 0, Monday = 1 etc.)
wDay	Day of the month: 1 ~ 31
wHour	Hour: 0 ~ 23
wMinute	Minute: 0 ~ 59
wSecond	Second: 0 ~ 59
wMilliseconds	Millisecond: 0 ~ 999

Requirements

Development environment	Target platform	PLC libraries to be integrated (category group)
TwinCAT v3.1.0	PC or CX (x86, x64, Arm®)	Tc2_Utilities (System)

6.6.124 ST_AcsiAnalogueValue

Analog value as structured type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiAnalogueValue :
STRUCT
    i : DINT;
    f : REAL;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
i	DINT	Integer value.
f	REAL	Floating point value.

6.6.125 ST_AcsiCalendarTime

Calendar time as a structured type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiCalendarTime :
STRUCT
    occ     : WORD;
    occType : E_AcsiOccType;
    occPer  : E_AcsiOccPer;
    weekDay : E_AcsiWeekDay;
    month   : E_AcsiMonth;
    day     : BYTE;
    hr      : BYTE;
    mn      : BYTE;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
occ	WORD	Occurrence of a calendar item.
occType	E_AcsiOccType [▶ 495]	Type of calendar item.
occPer	E_AcsiOccPer [▶ 494]	Repeat Period.
weekDay	E_AcsiWeekDay [▶ 518]	Day of the week.
month	E_AcsiMonth [▶ 493]	Month.
day	BYTE	Day.
hr	BYTE	Hour.
mn	BYTE	Minute.

6.6.126 ST_AcsiCell

Cell as a structured type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiCell :
STRUCT
    xStart  : REAL;
    xEnd    : REAL;
    yStart  : REAL;
    yEnd    : REAL;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
xStart	REAL	Cell coordinate x-start.
xEnd	REAL	Cell coordinate x-end.
yStart	REAL	Cell coordinate y-start.
yEnd	REAL	Cell coordinate y-end.

6.6.127 ST_AcsiDataSetClassInfo

Data set member container information.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiDataSetClassInfo :
STRUCT
    nMembers    : UDINT;
    bPersistent : BOOL;
    bDeletable  : BOOL;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
nMembers	UDINT	Number of data set members.
bPersistent	BOOL	TRUE => Persistent data set (bound to an LN). FALSE => Non-persistent data set (bound to active connection, will be deleted when connection is terminated).
bDeletable	BOOL	TRUE => Data set was configured dynamically (at runtime and can be deleted). FALSE => Data set was configured statically (cannot be deleted).

6.6.128 ST_AcsiFunctionalConstraints

Functional group as a structured type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiFunctionalConstraints :
STRUCT
    MX : BIT;
    ST_: BIT;
    CO : BIT;
    CF : BIT;
    DC : BIT;
    SP : BIT;
    SG : BIT;
    RP : BIT;
    LG : BIT;
    BR : BIT;
    GO : BIT;
    GS : BIT;
    SV : BIT;
    SE : BIT;
    MS : BIT;
    SC : BIT;
    US : BIT;
    EX : BIT;
//  XX : BIT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
MX	BIT	Measurands (analogue values) (use of FC in the definition of DATA).
ST_	BIT	Status information (use of FC in the definition of DATA).
CO	BIT	Control (use of FC in the definition of DATA).
CF	BIT	Configuration (use of FC in the definition of DATA).
DC	BIT	Description (use of FC in the definition of DATA).
SP	BIT	Setpoint (use of FC in the definition of DATA and control blocks).
SG	BIT	Setting group (use of FC in the definition of DATA).
RP	BIT	Unbuffered report (- > Reserved for control classes and use of the FC in the definition of control blocks).
LG	BIT	Logging .
BR	BIT	Buffered report (- > Reserved for control classes and use of the FC in the definition of control blocks).
GO	BIT	Goose control (- > Reserved for control classes and use of the FC in the definition of control blocks).
GS	BIT	Gsse control (- > Reserved FOR control classes AND use OF the FC in the definition OF control blo cks).
SV	BIT	Substitution (use of FC in the definition of DATA).
SE	BIT	Setting group editable (use of FC in the definition of DATA).
MS	BIT	Multicast sampled value control (- > Reserved for control classes and use of the FC in the definition of control blocks).
SC	BIT	SCL.
US	BIT	Unicast sampled value control (- > Reserved for control classes and use of the FC in the definition of control blocks).
EX	BIT	Extended definition (use of FC in the definition of DATA).
XX	BIT	Shall represent all DataAttributes of a Data of any FC. This value shall only used in t he FCD; "XX" shall not used as value in DataAttributes.

6.6.129 ST_AcsiIEDGroupClassInfo

Container information of the IED group.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiIEDGroupClassInfo :
STRUCT
    nIEDs : UDINT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
nIEDs	UDINT	Number of IEDs in the container.

6.6.130 ST_AcsiIntelligentElectronicDeviceClassInfo

Container information of the IED.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiIntelligentElectronicDeviceClassInfo :
STRUCT
    nLogicalDevices : UDINT;
```

```
    nAaNodes        : UDINT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
nLogicalDevices	UDINT	Number of logical devices in the server container.
nAaNodes	UDINT	Number of associated connections in the server's container.

6.6.131 ST_AcsiLogicalDeviceClassInfo

Container information of the logical device.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiLogicalDeviceClassInfo :
STRUCT
    nLogicalNodes : UDINT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
nLogicalNodes	UDINT	Number of logical node in the container of the logical device.

6.6.132 ST_AcsiLogicalNodeClassInfo

Container information of the logical node.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiLogicalNodeClassInfo :
STRUCT
    nDatas                              : UDINT;
    nDataSets                           : UDINT;
    nBufferedReportControlBlocks        : UDINT;
    nUnbufferedReportControlBlocks      : UDINT;
    nLogControlBlocks                   : UDINT;
    nSettingGroupControlBlocks          : UDINT;
    nGooseControlBlocks                 : UDINT;
    nGsseControlBlocks                  : UDINT;
    nMulticastSampledValueControlBlocks : UDINT;
    nUnicastSampledValueControlBlocks   : UDINT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
nDatas	UDINT	Number of data and data attributes in the container of the logical node.
nDataSets	UDINT	Number of data sets in the container of the logical node.
nBufferedReportControlBlocks	UDINT	Number of BRCBs in the container of the logical node.
nUnbufferedReportControlBlocks	UDINT	Number of URCBs in the container of the logical node.
nLogControlBlocks	UDINT	Number of LCBs in the container of the logical node.
nSettingGroupControlBlocks	UDINT	Number of SGCBs in the container of the logical node.
nGooseControlBlocks	UDINT	Number of GOCBs in the container of the logical node.
nGsseControlBlocks	UDINT	Number of GSCBs in the container of the logical node.
nMulticastSampledValueControlBlocks	UDINT	Number of MSVCBs in the container of the logical node.
nUnicastSampledValueControlBlocks	UDINT	Number of USVCBs in the container of the logical node.

6.6.133 ST_AcsiLogOptionalFields

This structure provides information about the optional fields transferred in a log.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiLogOptionalFields :
STRUCT
    ReasonForInclusion : BIT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
ReasonForInclusion	BIT	If TRUE, the log with the cause of the reporting is transferred.

6.6.134 ST_AcsiOptionalFields

This structure provides information about the optional fields transferred in a report.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiOptionalFields :
STRUCT
{ attribute 'hide'}    reserved : BIT;
    SequenceNumber     : BIT;
    ReportTimeStamp    : BIT;
    ReasonForInclusion : BIT;
    DataSetName        : BIT;
    DataReference      : BIT;
    BufferOverflow     : BIT;
    EntryID            : BIT;
    ConfRevision       : BIT;
    Segmentation       : BIT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
reserved	BIT	Reserved.
SequenceNumber	BIT	If TRUE, the report is transferred with a sequence number.
ReportTimeStamp	BIT	If TRUE, the report is transferred with a time stamp.
ReasonForInclusion	BIT	If TRUE, the report with the cause of the reporting is transferred.
DataSetName	BIT	If TRUE, the report also includes the name of each referenced data set.
DataReference	BIT	If TRUE, the report also includes the object references of the data set member.
BufferOverflow	BIT	If TRUE, the report includes the attribute for the buffer overflow.
EntryID	BIT	If TRUE, the entries in the report are assigned consecutive numbers.
ConfRevision	BIT	If TRUE, the report includes the attribute for the revision of the configuration.
Segmentation	BIT	This bit is reserved.

6.6.135 ST_AcsiOptionalSvFields

This structure provides information about the optional fields transferred in an SV message (Sampled Values).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiSvOptionalFields :
STRUCT
    RefreshTime           : BIT;
    SampleSynchronized    : BIT;
    SampleRate            : BIT;
    DataSetName           : BIT;
    Security              : BIT;
    SampleMode            : BIT;
    SynchSourceIdentity   : BIT;
{attribute 'hide'}unused7 : BIT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
RefreshTime	BIT	If TRUE, the SV message is transmitted with the "RefrTm" attribute.
SampleSynchronized	BIT	"SmpSynch" attribute is always transferred in the SV message. This value is now always TRUE and may no longer be FALSE (IEC 61850-9-2:2004).
SampleRate	BIT	If TRUE, the SV message is transmitted with the "SmpRate" attribute.
DataSetName	BIT	If TRUE, the SV message is transmitted with the "DatSet" attribute.
Security	BIT	If TRUE, the SV message is transmitted with the "Security" attribute.
SampleMode	BIT	If TRUE, the SV message is transmitted with the "SmpMod" attribute.
SynchSourceIdentity	BIT	If TRUE, the SV message is transmitted with the "GmIdentity" attribute.

6.6.136 ST_AcsiOriginator

Information about the command originator as a structured type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiOriginator :
STRUCT
    orCat   : E_AcsiOrCategory;
    orIdent : T_OCTET64;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
orCat	E_AcsiOrCategory [▶ 496]	Category of the command originator.
orIdent	T_OCTET64 [▶ 559]	Identification of the command originator.

6.6.137 ST_AcsiPhyComAddr

This structure describes a physical communication address.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiPhyComAddr :
STRUCT
    Addr     : T_OCTET6;
    PRIORITY : BYTE;
    VID      : WORD;
    APPID    : WORD;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
Addr	T_OCTET6 [▶ 558]	Media Access Control (MAC) address.
PRIORITY	BYTE	User priority.
VID	WORD	VLAN identification.
APPID	WORD	Application identification.

6.6.138 ST_AcsiPoint

Point as a structured type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiPoint :
STRUCT
    xVal : REAL;
    yVal : REAL;
    zVal : REAL;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
xVal	REAL	Point coordinate x.
yVal	REAL	Point coordinate y.
zVal	REAL	Point coordinate z.

6.6.139 ST_AcsiPresCond

This structure configures the availability of the optional data attributes system-wide.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiPresCond :
STRUCT
    AnalogueValue               : E_AcsiAnalogueValuePresCond;
    Cell_xEnd                   : BIT := 0;
    Cell_yStart                 : BIT := 0;
    Cell_yEnd                   : BIT := 0;
    Point_zVal                  : BIT := 0;
    RangeConfig_limDb           : BIT := 0;
    Unit_multiplier             : BIT := 0;
    ValWithTrans_transInd       : BIT := 0;
    Vector_ang                  : BIT := 0;
    Oper_AnalogueValue_operTm   : BIT := 0;
    Oper_BOOLEAN_operTm         : BIT := 0;
    Oper_CODEDENUM_operTm       : BIT := 0;
    Oper_ENUMERATED_operTm      : BIT := 0;
    Oper_INT32_operTm           : BIT := 0;
    Oper_INT8_operTm            : BIT := 0;
    SBOw_AnalogueValue_operTm   : BIT := 0;
    SBOw_BOOLEAN_operTm         : BIT := 0;
    SBOw_CODEDENUM_operTm       : BIT := 0;
    SBOw_ENUMERATED_operTm      : BIT := 0;
    SBOw_INT32_operTm           : BIT := 0;
    SBOw_INT8_operTm            : BIT := 0;
    Cancel_AnalogueValue_operTm : BIT := 0;
    Cancel_BOOLEAN_operTm       : BIT := 0;
    Cancel_CODEDENUM_operTm     : BIT := 0;
    Cancel_ENUMERATED_operTm    : BIT := 0;
    Cancel_INT32_operTm         : BIT := 0;
    Cancel_INT8_operTm          : BIT := 0;
    BRCB_ResvTms                : BIT := 0;
    BRCB_Owner                  : BIT := 0;
    URCB_Owner                  : BIT := 0;
    LCB_OptFlds                 : BIT := 0;
    LCB_BufTm                   : BIT := 0;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
AnalogueValue	E_AcsiAnalogueValuePresCond [▶ 475]	Configures the presence of the AnalogueValue "i" and "f" attributes in the control values (ctlVal) and status values (mxVal).
Cell_xEnd	BIT	False | 0 := optional (default); True | 1 := mandatory;
Cell_yStart	BIT	-//-
Cell_yEnd	BIT	-//-
Point_zVal	BIT	-//-
RangeConfig_limDb	BIT	-//-
Unit_multiplier	BIT	-//-
ValWithTrans_transInd	BIT	-//-
Vector_ang	BIT	-//-
Oper_AnalogueValue_operTm	BIT	-//-
Oper_BOOLEAN_operTm	BIT	-//-
Oper_CODEDENUM_operTm	BIT	-//-
Oper_ENUMERATED_operTm	BIT	-//-
Oper_INT32_operTm	BIT	-//-
Oper_Int8_operTm	BIT	-//-
SBOw_AnalogueValue_operTm	BIT	-//-
SBOw_BOOLEAN_operTm	BIT	-//-
SBOw_CODEDENUM_operTm	BIT	-//-
SBOw_ENUMERATED_operTm	BIT	-//-
SBOw_INT32_operTm	BIT	-//-
SBOw_INT8_operTm	BIT	-//-
Cancel_AnalogueValue_operTm	BIT	-//-
Cancel_BOOLEAN_operTm	BIT	-//-
Cancel_CODEDENUM_operTm	BIT	-//-
Cancel_ENUMERATE_operTm	BIT	-//-
Cancel_INT32_operTm	BIT	-//-
Cancel_INT8_operTm	BIT	-//-
BRCB_ResvTms	BIT	-//-
BRCB_Owner	BIT	-//-
URCB_Owner	BIT	-//-
LCB_OptFlds	BIT	-//-
LCB_BufTm	BIT	-//-

6.6.140 ST_AcsiPulseConfig

Pulse configuration as structured type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiPulseConfig :
STRUCT
    cmdQual : E_AcsiCmdQual;
    onDur   : DWORD;
    offDur  : DWORD;
    numPls  : DWORD;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
cmdQual	E_AcsiCmdQual [▶ 482]	Command Criterion.
onDur	DWORD	Pulse duration: ON.
offDur	DWORD	Pulse duration: OFF
numPls	DWORD	Number of pulses.

6.6.141 ST_AcsiQuality

Quality as a structured type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiQuality :
STRUCT
    Validity0       : BIT;
    Validity1       : BIT;
    Overflow        : BIT;
    OutOfRange      : BIT;
    BadReference    : BIT;
    Oscillatory     : BIT;
    Failure         : BIT;
    OldData         : BIT;
    Inconsistent    : BIT;
    Inaccurate      : BIT;
    Source          : BIT;
    Test            : BIT;
    OperatorBlocked : BIT;
    Derived         : BIT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
Validity0	BIT	Validity LSB. Value coding: see the table below.
Validity1	BIT	Validity MSB. Value coding: see the table below.
Overflow	BIT	Overflow.
OutOfRange	BIT	Not in permissible range.
BadReference	BIT	Incorrect reference.
Oscillatory	BIT	Oscillating.
Failure	BIT	Failure.
OldData	BIT	Old data.
Inconsistent	BIT	Inconsistent.
Inaccurate	BIT	Inaccurate.
Source	BIT	Source (0:=process, 1:=replaced).
Test	BIT	Test.
OperatorBlocked	BIT	Operator blocked.
Derived	BIT	Derived (0 = actual sensor value, 1 = calculated value).

Validity value coding

Value	Validity0	Validity1	Description
Good	0	0	Good (default value).
Invalid	0	1	Invalid.
Reserved	1	0	Reserved.
Questionable	1	1	Questionable.

6.6.142 ST_AcsiRangeConfig

Range configuration as a structured type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiRangeConfig :
STRUCT
    hhLim : ST_AcsiAnalogueValue;
    hLim  : ST_AcsiAnalogueValue;
    lLim  : ST_AcsiAnalogueValue;
    llLim : ST_AcsiAnalogueValue;
    min_  : ST_AcsiAnalogueValue;
    max_  : ST_AcsiAnalogueValue;
    limDb : DWORD;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
hhLim	ST_AcsiAnalogueValue [▶ 531]	High-High-Limit.
hLim	ST_AcsiAnalogueValue [▶ 531]	High-Limit.
lLim	ST_AcsiAnalogueValue [▶ 531]	Low-Limit.
llLim	ST_AcsiAnalogueValue [▶ 531]	Low-Low-Limit.
min_	ST_AcsiAnalogueValue [▶ 531]	Minimum value.
max_	ST_AcsiAnalogueValue [▶ 531]	Maximum value.
limDb	DWORD	

6.6.143 ST_AcsiReasonCode

Cause of transfer of a list item in a report. A report can include one, several or all list items of a data set. Each list item can have an own cause of transfer.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiReasonCode :
STRUCT
{ attribute 'hide'}reserved : BIT;
    DataChange              : BIT;
    QualityChange           : BIT;
    DataUpdate              : BIT;
    Integrity               : BIT;
    GeneralInterrogation    : BIT;
    ApplicationTrigger      : BIT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
reserved	BIT	Reserved.
DataChange	BIT	If TRUE, the report was sent due to a change in the data.
QualityChange	BIT	If TRUE, the report was sent due to a change of quality.
DataUpdate	BIT	If TRUE, the report was sent due to a data update. In contrast to the option: "DataChange", the report is sent even if the data is unchanged but was reset.
Integrity	BIT	If TRUE, the report was sent because the integrity had expired. In this case, the data sets are sent regularly after the expiry of the integrity.
GeneralInterrogation	BIT	If TRUE, the report was sent because of a one-time general interrogation of the data sets.
ApplicationTrigger	BIT	If TRUE, the report was triggered/sent by a function in the application.

6.6.144 ST_AcsiScaledValueConfig

Scaled value configuration as structured type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiScaledValueConfig :
STRUCT
    scaleFactor : REAL;
    offset      : REAL;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
scaleFactor	REAL	Scaling factor.
offset	REAL	Offset.

6.6.145 ST_AcsiSvOptionalFields

This structure provides information about the optional fields transferred with a sampled value.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiSvOptionalFields :
STRUCT
    RefreshTime           : BIT;
    SampleSynchronized    : BIT;
    SampleRate            : BIT;
    DataSetName           : BIT;
    Security              : BIT;
    SampleMode            : BIT;
    SynchSourceIdentity   : BIT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
RefreshTime	BIT	When TRUE, the sampled value is transmitted with the attribute value "RefrTm".
SampleSynchronized	BIT	This value is ignored and exists for compatibility with IEC 61850-9-2 Edition 1.0.
SampleRate	BIT	When TRUE, the sampled value is transmitted with the attribute value "SmpRate".
DataSetName	BIT	When TRUE, the sampled value is transmitted with the attribute value "DatSet".
Security	BIT	When TRUE, the sampled value is transmitted with the attribute value "Security".
SampleMode	BIT	When TRUE, the sampled value is transmitted with the attribute value "SmpMod".
SynchSourceIdentity	BIT	When TRUE, the sampled value is transmitted with the attribute value "GmIdentity".

6.6.146 ST_AcsiTriggerConditions

Trigger options for sending a report. Several options may be used at the same time. The options "Integrity" and "GeneralInterrogation" must not be set at the same time.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiTriggerConditions :
STRUCT
{ attribute 'hide'}reserved  : BIT;
    DataChange               : BIT;
    QualityChange            : BIT;
    DataUpdate               : BIT;
    Integrity                : BIT;
    GeneralInterrogation     : BIT;
{ attribute 'hide'}reserved2 : BIT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
reserved	BIT	Reserved.
DataChange	BIT	If TRUE, sending of a report was activated due to a change in the data.
QualityChange	BIT	If TRUE, sending of a report was activated due to a change of quality.
DataUpdate	BIT	If TRUE, sending of a report was activated due to a data update. In contrast to the option "DataChange", the report is sent even if the data is unchanged but was reset.
Integrity	BIT	If TRUE, sending of a report was activated because the integrity had expired. In this case, the data sets are sent regularly after the expiry of the integrity.
GeneralInterrogation	BIT	If TRUE, sending of a report was activated because of a one-time general interrogation of the data sets.
reserved2	BIT	Reserved to match the alignment with "ReasonCode".

6.6.147 ST_AcsiUnit

Unit of measurement as a structured type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiUnit :
STRUCT
    SIUnit     : E_AcsiSIUnit;
    multiplier : E_AcsiMultiplier;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
SIUnit	E_AcsiSIUnit [▶ 508]	SI unit.
multiplier	E_AcsiMultiplier [▶ 493]	Multiplier.

6.6.148 ST_AcsiValWithTrans

Value with transition state as structured type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiValWithTrans :
STRUCT
    posVal   : SINT;
    transInd : BOOL;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
posVal	SINT	Position.
transInd	BOOL	Transition state.

6.6.149 ST_AcsiVector

Vector as a structured type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_AcsiVector :
STRUCT
    mag : ST_AcsiAnalogueValue;
    ang : ST_AcsiAnalogueValue;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
mag	ST_AcsiAnalogueValue [▶ 531]	Size.
ang	ST_AcsiAnalogueValue [▶ 531]	Angle.

6.6.150 ST_GseAdapterInfo

Status and statistical data of the GSE network adapter.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
TYPE ST_GseAdapterInfo :
STRUCT
    eStatus    : E_GseLinkStatus;
    nRx        : UDINT;
    nTx        : UDINT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
eStatus	E_GseLinkStatus [▶ 522]	GSE network adapter link status.
nRx	UDINT	Number of frames received.
nTx	UDINT	Number of frames sent.

6.6.151 ST_GseGoCBPublisherTx

Information in the last-sent GOOSE message.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
TYPE ST_GseGoCBPublisherTx :
STRUCT
    eState               : E_GseGoCBPublisherState;
    (*------------------------------------------*)
    srcAddr              : ETHERNET_ADDRESS;
    dstAddr              : ETHERNET_ADDRESS;
    nAPPID               : WORD;
    nVID                 : WORD(0..4095);
    ePRIORITY            : E_AcsiVLanPriority;
    (*------------------------------------------*)
    sGoCBRef             : T_AcsiObjectReference;
    nTimeAllowedToLive   : UDINT;
    sDatSet              : T_AcsiObjectReference;
    sGoID                : T_AcsiVisString129;
    tT                   : T_UtcTime;
    nStNum               : UDINT;
    nSqNum               : UDINT;
    bTest                : BOOL;
    nConfRev             : UDINT;
    bNdsCom              : BOOL;
    nNumDatSetEntries    : UDINT;
    (*------------------------------------------*)
    nMinTime             : UDINT(1..16#7FFFFFFF);
    nMaxTime             : UDINT(1..16#7FFFFFFF);
    nMultiplier          : BYTE(1..16#FF);
    nTimeRemainingToLive : UDINT;

    ipAdapter            : I_GseAdapterClass;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
eState	E_GseGoCBPublisherState [▶ 521]	Publisher state.
srcAddr	ETHERNET_ADDRESS [▶ 528]	MAC address of the message source.
dstAddr	ETHERNET_ADDRESS [▶ 528]	MAC address of the message destination.
nAPPID	WORD	APPID.
nVID	WORD(0..4095)	Virtual LAN identification.
ePRIORITY	E_AcsiVLanPriority [▶ 517]	Virtual LAN priority.
sGoCBRef	T_AcsiObjectReference [▶ 555]	GOOSE control block reference string.
nTimeAllowedToLive	UDINT	Permitted lifetime/validity duration of the GOOSE message [ms].
sDatSet	T_AcsiObjectReference [▶ 555]	Data set reference string.
sGoID	T_AcsiVisString129 [▶ 556]	GOOSE ID.
tT	T_UtcTime [▶ 560]	Timestamp (UTC).
nStNum	UDINT	Status number.
nSqNum	UDINT	Sequence number.
bTest	BOOL	Simulation/test flag.
nConfRev	UDINT	Configuration revision number.
bNdsCom	BOOL	Configuration/parameterization required.
nNumDatSetEntries	UDINT	Number of data set entries.
nMinTime	UDINT(1..16#7FFFFFFF)	Start time for message repetitions [ms].
nMaxTime	UDINT(1..16#7FFFFFFF)	Final time for message repetitions [ms].
nMultiplier	BYTE(1..16#FF)	Multiplier for message repetitions [ms].
nTimeRemainingToLive	UDINT	Remaining lifetime/validity duration of the GOOSE message [ms].
ipAdapter	I_GseAdapterClass [▶ 397]	GSE network adapter interface pointer.

6.6.152 ST_GseGoCBSubscriberRx

Information in the last-received GOOSE message.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
TYPE ST_GseGoCBSubscriberRx :
STRUCT
    eState               : E_GseGoCBSubscriberState;
    (*-------------------------------------------*)
    srcAddr              : ETHERNET_ADDRESS;
    dstAddr              : ETHERNET_ADDRESS;
    nAPPID               : WORD;
    nVID                 : WORD(0..4095);
    ePRIORITY            : E_AcsiVLanPriority;
    (*-------------------------------------------*)
    sGoCBRef             : T_AcsiObjectReference;
    nTimeAllowedToLive   : UDINT;
    sDatSet              : T_AcsiObjectReference;
    sGoID                : T_AcsiVisString129;
    tT                   : T_UtcTime;
    nStNum               : UDINT;
    nSqNum               : UDINT;
    bTest                : BOOL;
    nConfRev             : UDINT;
    bNdsCom              : BOOL;
    nNumDatSetEntries    : UDINT;
    (*-------------------------------------------*)
    nTimeRemainingToLive : UDINT;
```

```
    ipAdapter           : I_GseAdapterClass;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
eState	E_GseGoCBSubscriberState [▶ 521]	Subscriber state.
srcAddr	ETHERNET_ADDRESS [▶ 528]	MAC address of the message source.
dstAddr	ETHERNET_ADDRESS [▶ 528]	MAC address of the message destination.
nAPPID	WORD	APPID.
nVID	WORD(0..4095)	Virtual LAN identification.
ePRIORITY	E_AcsiVLanPriority [▶ 517]	Virtual LAN priority.
sGoCBRef	T_AcsiObjectReference [▶ 555]	GOOSE control block reference string.
nTimeAllowedToLive	UDINT	Permitted lifetime/validity duration of the GOOSE message [ms].
sDatSet	T_AcsiObjectReference [▶ 555]	Data set reference string.
sGoID	T_AcsiVisString129 [▶ 556]	GOOSE ID.
tT	T_UtcTime [▶ 560]	Timestamp (UTC).
nStNum	UDINT	Status number.
nSqNum	UDINT	Sequence number.
bTest	BOOL	Simulation/test flag.
nConfRev	UDINT	Configuration revision number.
bNdsCom	BOOL	Configuration/parameterization required.
nNumDatSetEntries	UDINT	Number of data set entries.
nTimeRemainingToLive	UDINT	Remaining lifetime/validity duration of the GOOSE message [ms].
ipAdapter	I_GseAdapterClass [▶ 397]	GSE network adapter interface pointer.

6.6.153 ST_GseMsvCBPublisherTx

Information in the last-sent SV message.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
TYPE ST_GseMsvCBPublisherTx :
STRUCT
    eState              : E_GseSvCBState;
    (*-------------------------------------------------------------*)
    srcAddr             : ETHERNET_ADDRESS;
    dstAddr             : ETHERNET_ADDRESS;
    nAPPID              : WORD;
    nVID                : WORD(0..4095);
    ePRIORITY           : E_AcsiVLanPriority;
    (*-------------------------------------------------------------*)
    ipAdapter           : I_GseAdapterClass;
    ipApdu              : I_GseSvApduClass;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
eState	E_GseSvCBState [▶ 523]	Publisher state.
srcAddr	ETHERNET_ADDRESS [▶ 528]	MAC address of the message source.
dstAddr	ETHERNET_ADDRESS [▶ 528]	MAC address of the message destination.
nAPPID	WORD	APPID.
nVID	WORD(0..4095)	Virtual LAN identification.
ePRIORITY	E_AcsiVLanPriority [▶ 517]	Virtual LAN priority.
ipAdapter	I_GseAdapterClass [▶ 397]	GSE network adapter interface pointer.
ipApdu	I_GseSvApduClass [▶ 405]	SV-APDU object interface pointer.

6.6.154 ST_GseMsvCBSubscriberRx

Information in the last-received SV message.

Namespace: Tc3_Gse [▶ 102]
Library: Tc3_Gse (Tc3_Gse.compiled-library)

```
TYPE ST_GseMsvCBSubscriberRx :
STRUCT
    eState              : E_GseSvCBState;
    (*-------------------------------------------------------------*)
    srcAddr             : ETHERNET_ADDRESS;
    dstAddr             : ETHERNET_ADDRESS;
    nAPPID              : WORD;
    nVID                : WORD(0..4095);
    ePRIORITY           : E_AcsiVLanPriority;
    (*-------------------------------------------------------------*)
    ipAdapter           : I_GseAdapterClass;
    ipApdu              : I_GseSvApduClass;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
eState	E_GseSvCBState [▶ 523]	Subscriber state.
srcAddr	ETHERNET_ADDRESS [▶ 528]	MAC address of the message source.
dstAddr	ETHERNET_ADDRESS [▶ 528]	MAC address of the message destination.
nAPPID	WORD	APPID.
nVID	WORD(0..4095)	Virtual LAN identification.
ePRIORITY	E_AcsiVLanPriority [▶ 517]	Virtual LAN priority.
ipAdapter	I_GseAdapterClass [▶ 397]	GSE network adapter interface pointer.
ipApdu	I_GseSvApduClass [▶ 405]	SV-APDU object interface pointer.

6.6.155 ST_ScsmAdditionalCauseDiagnosticEventInfo

This structure provides additional diagnostic information about a negative feedback during the execution of a control service (operate, select, cancel etc.). The MMS InformationReport service is available for transferring this information when using MMS mapping.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
TYPE ST_ScsmAdditionalCauseDiagnosticEventInfo :
STRUCT
    eService : E_AcsiServiceType;
    CntrlObj : T_AcsiObjectReference;
    Error    : E_AcsiApplError;
```

```
    Origin   : ST_AcsiOriginator;
    CtlNum   : USINT;
    AddCause : E_AcsiApplAddCause;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
eService	E_AcsiServiceType [▶ 505]	Designation of service, the execution of which has generated the additional diagnostic information.
CntrlObj	T_AcsiObjectReference [▶ 555]	String with the object reference of the object (control service) that generated the additional diagnostic information during execution. If the string is empty, this parameter is optional and is not available.
Error	E_AcsiApplError [▶ 478]	Diagnostic information about the error source (state machine of the service, of the Operate or TimeActivated service itself or another).
Origin	ST_AcsiOriginator [▶ 538]	Identifies the client that initiated the execution of the service.
CtlNum	USINT	This number corresponds to the service sequence number of the client that initiated the execution of the service. If the value is 0, this parameter is optional and is not available.
AddCause	E_AcsiApplAddCause [▶ 476]	Additional information about the cause.

6.6.156 ST_ScsmBrCBJobInfo

BRCB statistical and status information.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
TYPE ST_ScsmBrCBJobInfo :
STRUCT
    eState       : E_ScsmRCBState;(* Client status. *)
    sRptID       : T_AcsiVisString129;
    sDatSet      : T_AcsiObjectReference;
    nSqNum       : WORD;
    tTimeOfEntry : T_BinaryTime;
    stReasonCode : ST_AcsiReasonCode;
    nEntryID     : LWORD;
    ipAA         : I_ScsmAssociationClass;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
eState	E_ScsmRCBState [▶ 525]	RCB status.
sRptID	T_AcsiVisString129 [▶ 556]	Report identification string.
sDatSet	T_AcsiObjectReference [▶ 555]	Data set reference string.
nSqNum	WORD	Report sequence number.
tTimeOfEntry	T_BinaryTime [▶ 556]	Report timestamp.
stReasonCode	ST_AcsiReasonCode [▶ 542]	Report cause of transfer.
nEntryID	LWORD	Identification of the report entry.
ipAA	I_ScsmAssociationClass [▶ 440]	Interface pointer to the application association.

6.6.157 ST_ScsmBrCBValuesSet

Selection of BRCB control block attributes to be written.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
TYPE ST_ScsmBrCBValuesSet :
STRUCT
    RptID       : BIT;(* R/W *)
    RptEna      : BIT;(* R/W *)
    DatSet      : BIT;(* R/W *)
    ConfRev     : BIT;(* Read only attribute! *)
    OptFlds     : BIT;(* R/W *)
    BufTm       : BIT;(* R/W *)
    SqNum       : BIT;(* Read only attribute! *)
    TrgOps      : BIT;(* R/W *)
    IntgPd      : BIT;(* R/W *)
    GI          : BIT;(* R/W *)
    PurgeBuf    : BIT;(* R/W *)
    EntryID     : BIT;(* R/W *)
    TimeOfEntry : BIT;(* Read only attribute! *)
    ResvTms     : BIT;(* R/W *)
    Owner       : BIT;(* Read only attribute! *)
END_STRUCT
END_TYPE
```

6.6.158 ST_ScsmCommandTerminationEventInfo

This structure provides additional information about the service feedback: CommandTermination.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
TYPE ST_ScsmCommandTerminationEventInfo :
STRUCT
    eService : E_AcsiServiceType;
    CntrlObj : T_AcsiObjectReference;
    OperTm   : T_UtcTime;
    Origin   : ST_AcsiOriginator;
    CtlNum   : USINT;
    T        : T_UtcTime;
    Test     : BOOL;
    Check    : E_AcsiCheck;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
eService	E_AcsiServiceType [▶ 505]	Designation of service, the execution of which has generated the service feedback.
CntrlObj	T_AcsiObjectReference [▶ 555]	String with the object reference of the object (control service) that generated the additional information during execution. If the string is empty, this parameter is optional and is not available.
OperTm	T_UtcTime [▶ 560]	
Origin	ST_AcsiOriginator [▶ 538]	Identifies the client that initiated the execution of the service.
CtlNum	USINT	This number corresponds to the service sequence number of the client that initiated the execution of the service.
T	T_UtcTime [▶ 560]	
Test	BOOL	
Check	E_AcsiCheck [▶ 479]	

6.6.159 ST_ScsmConfirmedEventInfo

This structure provides additional information about the service feedback.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
TYPE ST_ScsmConfirmedEventInfo :
STRUCT
    eService       : E_AcsiServiceType;
    hUser          : PVOID;
    nInvokeID      : UDINT;
    nEventSequence : UDINT;
    bMoreFollows   : BOOL;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
eService	E_AcsiServiceType [▶ 505]	Designation of service, the execution of which has generated the service feedback.
hUser	PVOID	Optional user-defined parameter or pointer. Any value for this parameter can be transferred in the methods for service activation (XyzMethodReq). The value is then mirrored in the event handling routine for the service feedback.
nInvokeID	UDINT	Identifies the service activation. Each new service activation (XyzMethodReq) is numbered consecutively. A copy of this number is mirrored in the service feedback (OnXyzMethodCnf).
nEventSequence	UDINT	Sequence number of the event handling routine. If the event handling routine is called repeatedly, the calls are numbered consecutively.
bMoreFollows	BOOL	FALSE => last call of the event handling routine. TRUE => further calls of the event handling routine will follow.

6.6.160 ST_ScsmGoCBJobInfo

GOCB statistical and status information.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
TYPE ST_ScsmGoCBJobInfo :
STRUCT
    bEnabled : BOOL;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
bEnabled	BOOL	GOOSE activation status.

6.6.161 ST_ScsmGoCBValuesSet

Selection of GOOSE control block attributes to be written.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
TYPE ST_ScsmGoCBValuesSet :
STRUCT
    GoEna      : BIT;(* R/W *)
    GoID       : BIT;(* R/W *)
    DatSet     : BIT;(* R/W *)
    ConfRev    : BIT;(* Read only attribute! *)
    NdsCom     : BIT;
    DstAddress : BIT;
    MinTime    : BIT;
    MaxTime    : BIT;
```

```
    FixedOffs  : BIT;
END_STRUCT
END_TYPE
```

6.6.162 ST_ScsmMsvCBValuesSet

Selection of SV control block attributes to be written.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
TYPE ST_ScsmMsvCBValuesSet :
STRUCT
    SvEna      : BIT; (* R/W *)
    MsvID      : BIT; (* Read only attribute! *)
    DatSet     : BIT; (* Read only attribute! *)
    ConfRev    : BIT; (* Read only attribute! *)
    SmpRate    : BIT; (* Read only attribute! *)
    OptFlds    : BIT; (* Read only attribute! *)
    SmpMod     : BIT; (* Read only attribute! *)
    DstAddress : BIT; (* Read only attribute! *)
    noASDU     : BIT; (* Read only attribute! *)
END_STRUCT
END_TYPE
```

6.6.163 ST_ScsmReportEventInfo

This structure provides additional information about the received report.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
TYPE ST_ScsmReportEventInfo :
STRUCT
    eService            : E_AcsiServiceType;
    RptID               : T_AcsiObjectReference;
    OptFlds             : ST_AcsiOptionalFields;
    SeqNum              : UINT;
    TimeOfEntry         : T_BinaryTime;
    DatSet              : T_AcsiObjectReference;
    BufOvfl             : BOOL;
    EntryID             : T_OCTET8;
    ConfRev             : UDINT;
    SubSeqNum           : UINT;
    MoreSegmentsFollow  : BOOL;
    DataReference       : T_AcsiObjectReference;
    ReasonCode          : ST_AcsiReasonCode;
    nEventSequence      : UDINT;
    bMoreFollows        : BOOL;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
eService	E_AcsiServiceType [▶ 505]	Designation of service, the execution of which has generated the report.
RptID	T_AcsiObjectReference [▶ 555]	Unique report ID.
OptFlds	ST_AcsiOptionalFields [▶ 536]	Information about the optional fields.
SeqNum	UINT	Report sequence number.
TimeOfEntry	T_BinaryTime [▶ 556]	Report timestamp.
DatSet	T_AcsiObjectReference [▶ 555]	DataSet reference
BufOvfl	BOOL	Buffer overflow.
EntryID	T_OCTET8 [▶ 558]	Identification of the report entry.
ConfRev	UDINT	Configuration revision.
SubSeqNum	UINT	Subsequence number.
MoreSegmentsFollow	BOOL	If TRUE, further segments follow.
DataReference	T_AcsiObjectReference [▶ 555]	Datareferenz.
ReasonCode	ST_AcsiReasonCode [▶ 542]	Cause of transfer.
nEventSequence	UDINT	Sequence number of the event handling routine. If the event handling routine is called repeatedly, the calls are numbered consecutively.
bMoreFollows	BOOL	FALSE => last call of the event handling routine. TRUE => further calls of the event handling routine will follow.

6.6.164 ST_ScsmUrCBJobInfo

URCB statistical and status information.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
TYPE ST_ScsmUrCBJobInfo :
STRUCT
    eState       : E_ScsmRCBState;(* Client status. *)
    sRptID       : T_AcsiVisString129;
    sDatSet      : T_AcsiObjectReference;
    nSqNum       : BYTE;
    tTimeOfEntry : T_BinaryTime;
    stReasonCode : ST_AcsiReasonCode;
    ipAA         : I_ScsmAssociationClass;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
eState	E_ScsmRCBState [▶ 525]	RCB status.
sRptID	T_AcsiVisString129 [▶ 556]	Report identification string.
sDatSet	T_AcsiObjectReference [▶ 555]	Data set reference string.
nSqNum	BYTE	Report sequence number.
tTimeOfEntry	T_BinaryTime [▶ 556]	Report timestamp.
stReasonCode	ST_AcsiReasonCode [▶ 542]	Report cause of transfer.
ipAA	I_ScsmAssociationClass [▶ 440]	Interface pointer to the application association.

6.6.165 ST_ScsmUrCBValuesSet

Selection of URCB control block attributes to be written.

Namespace: Tc3_iec61850_8_1 [▶ 102]
Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

```
TYPE ST_ScsmUrCBValuesSet :
STRUCT
    RptID   : BIT;(* R/W *)
    RptEna  : BIT;(* R/W *)
    Resv    : BIT;(* R/W *)
    DatSet  : BIT;(* R/W *)
    ConfRev : BIT;(* Read only attribute! *)
    OptFlds : BIT;(* R/W *)
    BufTm   : BIT;(* R/W *)
    SqNum   : BIT;(* Read only attribute! *)
    TrgOps  : BIT;(* R/W *)
    IntgPd  : BIT;(* R/W *)
    GI      : BIT;(* R/W *)
    Owner   : BIT;(* Read only attribute! *)
END_STRUCT
END_TYPE
```

6.6.166 ST_ServiceErrorClass

Structured type with event or error information. See: TcEventEntry [▶ 530].

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
ALIAS ST_ServiceErrorClass : TcEventEntry
```

Components

Name	Type	Description
uuidEventClass	GUID	Event class. The GUID specifies the event source.
nEventID	UDINT	Event ID or error number.
eSeverity	TcEventSeverity [▶ 530]	Severity level

6.6.167 T_AcsiCurrency

Null-terminated ISO 4217 currency code as text (IEC 61850-7-2 type "Currency").

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
ALIAS T_AcsiCurrency : STRING(3)
```

6.6.168 T_AcsiObjectName

Object name as a string with a maximum length of 64 characters (default value).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
ALIAS T_AcsiObjectName : STRING(Param_Acsi.cMaxLengthOfObjectName)
```

6.6.169 T_AcsiObjectReference

Object reference as a string with a maximum length of 255 characters (default value).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
ALIAS T_AcsiObjectReference : STRING(Param_Acsi.cMaxLengthOfObjectReference)
```

6.6.170 T_AcsiTag

In a variable of this type an additional information can be stored (by default an unsigned number but also pointers).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
ALIAS T_AcsiTag : UXINT
```

6.6.171 T_AcsiUnicode255

Unicode string with a maximum length of 255 characters (UTF-8 encoded/decoded during data transmission).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
ALIAS T_AcsiUnicode255 : T_MaxWString
```

6.6.172 T_AcsiVisString64

Visible string with a maximum length of 64 characters (default value).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
ALIAS T_AcsiVisString64 : STRING(Param_Acsi.cMaxLengthOfVisString64)
```

6.6.173 T_AcsiVisString129

Visible string with a maximum length of 129 characters (default value).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
ALIAS T_AcsiVisString129 : STRING(Param_Acsi.cMaxLengthOfVisString129)
```

6.6.174 T_AcsiVisString255

Visible string with a maximum length of 255 characters (default value).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
ALIAS T_AcsiVisString255 : STRING(Param_Acsi.cMaxLengthOfVisString255)
```

6.6.175 T_BinaryTime

Binary time information.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
TYPE T_BinaryTime :
STRUCT
    timeOfDay : TOD;
    day       : WORD;
{ attribute 'hide'}reserved : WORD := 0;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
timeOfDay	TOD	Number of milliseconds since midnight of the current day.
day	WORD	Number of days since January 1, 1984.
reserved	WORD	Reserved (0).

6.6.176 T_INT24

Signed 24-bit number.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
ALIAS T_INT24 : ARRAY[0..2] OF BYTE
```

6.6.177 T_INT128

Signed 128-bit number.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
TYPE T_INT128 :
STRUCT
    Lo : LWORD;
    Hi : LWORD;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
Lo	LWORD	Lower bytes.
Hi	LWORD	Upper bytes.

6.6.178 T_MmsInteger8

Signed 8-bit number.

Namespace: Tc3_Mms [▶ 102]
Library: Tc3_Mms (Tc3_Mms.compiled-library)

```
ALIAS T_MmsInteger8 : SINT(-128..127)
```

6.6.179 T_MmsInteger16

Signed 16-bit number.

Namespace: Tc3_Mms [▶ 102]
Library: Tc3_Mms (Tc3_Mms.compiled-library)

```
ALIAS T_MmsInteger16 : INT(-32768..32767)
```

6.6.180 T_MmsInteger32

Signed 32-bit number.

Namespace: Tc3_Mms [▶ 102]
Library: Tc3_Mms (Tc3_Mms.compiled-library)

```
ALIAS T_MmsInteger32 : DINT(-2147483648..2147483647)
```

6.6.181 T_MmsString

MMS type: MMSString (encoded/decoded as UTF-8 during data transmission).

Namespace: Tc3_Mms [▶ 102]
Library: Tc3_Mms (Tc3_Mms.compiled-library)

```
ALIAS T_MmsString : T_MaxWString
```

6.6.182 T_MmsVisibleString

MMS type: VisibleString.

Namespace: Tc3_Mms [▶ 102]
Library: Tc3_Mms (Tc3_Mms.compiled-library)

```
ALIAS T_MmsVisibleString : T_MaxString
```

6.6.183 T_OCTET3

Octet string with a maximum length of 3 octets.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
ALIAS T_OCTET3 : ARRAY[0..2] OF BYTE
```

6.6.184 T_OCTET4

Octet string with a maximum length of 4 octets.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
ALIAS T_OCTET4 : ARRAY[0..3] OF BYTE
```

6.6.185 T_OCTET6

Octet string with a maximum length of 6 octets.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
ALIAS T_OCTET6 : ARRAY[0..5] OF BYTE
```

6.6.186 T_OCTET8

Octet string with a maximum length of 8 octets.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
ALIAS T_OCTET8 : ARRAY[0..7] OF BYTE
```

6.6.187 T_OCTET16

Octet string with a maximum length of 16 octets.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
ALIAS T_OCTET16 : ARRAY[0..15] OF BYTE
```

6.6.188 T_OCTET20

Octet string with a maximum length of 20 octets.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
ALIAS T_OCTET20 : ARRAY[0..19] OF BYTE
```

6.6.189 T_OCTET32

Octet string with a maximum length of 32 octets.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
ALIAS T_OCTET32 : ARRAY[0..31] OF BYTE
```

6.6.190 T_OCTET35

Octet string with a maximum length of 35 octets.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
ALIAS T_OCTET35 : ARRAY[0..34] OF BYTE
```

6.6.191 T_OCTET64

Octet string with a maximum length of 64 octets.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
ALIAS T_OCTET64 : ARRAY[0..63] OF BYTE
```

6.6.192 T_UINT24

Unsigned 24-bit number.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
ALIAS T_UINT24 : ARRAY[0..2] OF BYTE
```

6.6.193 T_UINT128

Unsigned 128-bit number.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
TYPE T_UINT128 :
STRUCT
    Lo : LWORD;
    Hi : LWORD;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
Lo	LWORD	Lower bytes.
Hi	LWORD	Upper bytes.

6.6.194 T_UtcTime

UTC time information.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
TYPE T_UtcTime :
STRUCT
    secondSinceEpoch : DT;
    fractionOfSecond : T_UINT24;
    quality          : T_UtcTimeQuality := ( Accuracy0 := 0, Accuracy1 := 0, Accuracy2 := 0, Accurac
y3 := 0, Accuracy4 := 0 );
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
secondSinceEpoch	DT	Number of whole seconds since January 1, 1970.
fractionOfSecond	T_UINT24 [▶ 559]	Seconds fraction encoded as 24-bit number. Coding: Bit 0 := 0.5s Bit 1 := 0.25s Bit 2 := 0.125s etc. The max. resolution is ~60 microseconds.
quality	T_UtcTimeQuality [▶ 560]	Additional information about the quality and accuracy of the time information.

6.6.195 T_UtcTimeQuality

Information about the quality and accuracy of the UTC time information.

Namespace: Tc3_Collections [▶ 101]
Library: Tc3_Collections (Tc3_Collections.compiled-library)

```
TYPE T_UtcTimeQuality :
STRUCT
    LeapSecondsKnown     : BIT;
    ClockFailure         : BIT;
    ClockNotSynchronized : BIT;
    Accuracy0            : BIT;
    Accuracy1            : BIT;
    Accuracy2            : BIT;
    Accuracy3            : BIT;
    Accuracy4            : BIT;
END_STRUCT
END_TYPE
```

Components

Name	Type	Description
LeapSecondsKnown	BIT	Quality of the leap seconds: FALSE := unknown, TRUE := known.
ClockFailure	BIT	Quality of the timer: FALSE := good, TRUE := faulty.
ClockNotSynchronized	BIT	Quality of the time synchronization: FALSE := synchronous, TRUE := not synchronous.
Accuracy0	BIT	Accuracy of the seconds fraction. This value determines the maximum number of significant bits of the T_UtcTime [▶ 560].fractionOfSecond member.
Accuracy1	BIT	
Accuracy2	BIT	
Accuracy3	BIT	
Accuracy4	BIT	

6.6.196 U_AcsiAnalogueCtlVal

Analog value as UNION type (integer or floating point number).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE U_AcsiAnalogueCtlVal :
UNION
    i : DINT;
    f : REAL;
END_UNION
END_TYPE
```

Components

Name	Type	Description
i	DINT	Analog value as integer
f	REAL	Analog value as floating point number

6.6.197 U_AcsiCtlVal

General control value for different data classes (CDC) used for the switch control.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE U_AcsiCtlVal :
UNION
    SPC : BOOL;
    DPC : BOOL;
    INC : DINT;
    ENC : U_AcsiEnumeratedCtlVal;
    BSC : E_AcsiTcmd;
    ISC : SINT(-64..63);
    APC : U_AcsiAnalogueCtlVal;
    BAC : E_AcsiTcmd;
END_UNION
END_TYPE
```

Components

Name	Type	Description
SPC	BOOL	Control value for single command (controllable single point): FALSE := Off TRUE := On
DPC	BOOL	Control value for double command (controllable double point): FALSE := Off TRUE := On
INC	DINT	Control value for a setpoint (controllable integer status).
ENC	U_AcsiEnumeratedCtlVal [▶ 562]	Control value for a setpoint (Controllable Enumerated Status).
BSC	E_AcsiTcmd [▶ 513]	Control value for a regulating step command (Binary Controlled Step Position Information).
ISC	SINT(-64..63)	Control value for regulating step command (integer controlled step position information).
APC	U_AcsiAnalogueCtlVal [▶ 561]	Control value for an analog setpoint (Controllable Analog Process Value).
BAC	E_AcsiTcmd [▶ 513]	Control value for an analog setpoint (Binary Controlled Analog Process Value).

6.6.198 U_AcsiEnumeratedCtlVal

Control value for controllable enumerated status (ENC).

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE U_AcsiEnumeratedCtlVal :
UNION
    iAny      : SINT;
    Mode      : E_AcsiMod;
    PmpCtlGen : E_AcsiPmpCtlGen;
    PmpCtl    : E_AcsiPmpCtl;
    FanCtlGen : E_AcsiFanCtlGen;
    FanCtl    : E_AcsiFanCtl;
    OpModSyn  : E_AcsiOpModSyn;
END_UNION
END_TYPE
```

Components

Name	Type	Description
iAny	SINT	Control value as a generic 8-bit integer.
Mode	E_AcsiMod [▶ 492]	Control value for mode.
PmpCtlGen	E_AcsiPmpCtlGen [▶ 499]	Generic control value for pump control.
PmpCtl	E_AcsiPmpCtl [▶ 499]	Control value for pump control.
FanCtlGen	E_AcsiFanCtlGen [▶ 486]	Generic control value for fan control.
FanCtl	E_AcsiFanCtl [▶ 486]	Control value for fan control.
OpModSyn	E_AcsiOpModSyn [▶ 495]	Control value for the operation mode.

6.6.199 U_AcsiQuality

Quality as a union data type.

Namespace: Tc3_Acsi [▶ 101]
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE U_AcsiQuality :
UNION
    q: ST_AcsiQuality;
    n16: WORD;
END_UNION
END_TYPE
```

Components

Name	Type	Description
q	ST_AcsiQuality [▶ 541]	Quality as a structured type.
n16	WORD	Quality as an unsigned 16-bit number.

6.7 Parameter lists

6.7.1 Param_Acsi

Namespace: Tc3_Acsi [▶ 101]
Type: Parameter list
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Parameter

Name	Type	Value	Description
cMaxLengthOfObjectName	UDINT	64	Maximum length of the object name string.
cMaxLengthOfObjectReference	UDINT	255	Maximum length of the reference string.
cMaxLengthOfVisString64	UDINT	64	Maximum length of the “VisString64” data type.
cMaxLengthOfVisString129	UDINT	129	Maximum length of the “VisString129” data type.
cMaxLengthOfVisString255	UDINT	255	Maximum length of the “VisString255” data type.
cMaxLengthOfClassName	UDINT	20	Maximum name length of the logical node or CDC.
cMaxLengthOfQualityString	INT(18..19)	18	Maximum length of the “Quality” data type formatted as a string. When the value is 19, the optional parameter “Derived” is also included in the string.
cDefault_TimeStamp	T_UtcTime	secondSinceEpoch:=DT#1970-01-01-00:00:00, fractionOfSecond:=[0,0,0], quality:=(LeapSecondsKnown:=0, ClockFailure:=0, ClockNotSynchronized:=0, Accuracy0:=0, Accuracy1:=0, Accuracy2:=0, Accuracy3:=0, Accuracy4:=0)	Default initial value for all “TimeStamp” DA instances.
cDefault_EntryTime_D	DATE	D#1984-01-01	Default initial value for all "EntryTime.day" data element instances.
cDefault_EntryTime_TimeOfDay	TOD	TOD#0:0:0	Default initial value for all “EntryTime.TimeOfDay” data element instances.
cDefault_EntryTime	T_BinaryTime	timeOfDay:=cDefault_EntryTime_TimeOfDay, day:=0	Default initial value for all “EntryTime” DA instances.
cDefault_Currency	STRING(3)	''	Default initial value for all “Currency” DA instances.
cDefault_EntryID	T_OCTET8	[8(0)]	Default initial value for all “EntryID” DA instances.
cDefault_Originator	ST_AcsiOriginator	orCat:=E_AcsiOrCategory.NotSupported, orIdent:=[64(0)]	Default initial value for all “Originator” DA instances.
cDefault_Quality	ST_AcsiQuality	Validity0:=0, Validity1:=0, Overflow:=0, OutOfRange:=0, BadReference:=0, Oscillatory:=0, Failure:=0, OldData:=0, Inconsistent:=0, Inaccurate:=0, Source:=0, Test:=0, OperatorBlocked:=0	Default initial value for all “Quality” DA instances.
cDefault_Dbpos	E_AcsiDbpos	E_AcsiDbpos.Intermediate	Default initial value for all “Dbpos” DA instances.
cDefault_Tcmd	E_AcsiTcmd	E_AcsiTcmd.Stop	Default initial value for all “Tcmd” DA instances.

Name	Type	Value	Description
cDefault_BrCB_OptFlds	ST_AcsiOptionalFields	SequenceNumber:=TRUE, ReportTimeStamp:=TRUE, ReasonForInclusion:=TRUE, DataSetName:=TRUE, DataReference:=FALSE, BufferOverflow:=TRUE, EntryID:=TRUE, ConfRevision:=TRUE, Segmentation:=FALSE	Default initial value for all BRCB “OptionalFields” DA instances.
cDefault_BrCB_TrgOps	ST_AcsiTriggerConditions	DataChange:=FALSE, QualityChange:=FALSE, DataUpdate:=FALSE, Integrity:=FALSE, GeneralInterrogation:=TRUE	Default initial value for all BRCB “TriggerConditions” DA instances.
cDefault_BrCB_TimeOfEntry	T_BinaryTime	timeOfDay:=TOD#0:0:0, day:=0	Default initial value for all BRCB “TimeOfEntry” DA instances.
cDefault_UrCB_OptFlds	ST_AcsiOptionalFields	SequenceNumber:=TRUE, ReportTimeStamp:=TRUE, ReasonForInclusion:=TRUE, DataSetName:=TRUE, DataReference:=FALSE, BufferOverflow:=FALSE, EntryID:=FALSE, ConfRevision:=TRUE, Segmentation:=FALSE	Default initial value for all URCB “OptionFields” DA instances.
cDefault_UrCB_TrgOps	ST_AcsiTriggerConditions	DataChange:=FALSE, QualityChange:=FALSE, DataUpdate:=FALSE, Integrity:=FALSE, GeneralInterrogation:=TRUE	Default initial value for all URCB “TriggerConditions” DA instances.
cDefault_GoCB_MinTime	UDINT(1..16#7FFFFFFF)	50	Default GOCB delay time for sending the repetitions in milliseconds.
cDefault_GoCB_MaxTime	UDINT(1..16#7FFFFFFF)	5,000	Default GOCB monitoring time in milliseconds.
cDefault_GoCB_Multiplier	BYTE(1..BYTE#16#FF)	2	Default initial value for GOOSE publisher transmission repeat multiplier.
cDefault_GoCB_Security	DWORD(0..16#0FFFFFFF)	0	Default GOOSE header Reserved1 and Reserved2 28 security bits.
cDefault_LCB_OldEntrTm	T_BinaryTime	timeOfDay:=TOD#0:0:0, day:=0	Default initial value for all LCB “OldEntrTm”.
cDefault_LCB_NewEntrTm	T_BinaryTime	timeOfDay:=TOD#0:0:0, day:=0	Default initial value for all LCB “NewEntrTm”.
cDefault_LCB_OptFlds	ST_AcsiLogOptionalFields	ReasonForInclusion:=TRUE	Default initial value for all LCB “OptionFields” DA instances.
cDefault_LCB_TrgOps	ST_AcsiTriggerConditions	DataChange:=FALSE, QualityChange:=FALSE, DataUpdate:=FALSE, Integrity:=FALSE, GeneralInterrogation:=TRUE	Default initial value for all LCB “TriggerConditions” DA instances.

Name	Type	Value	Description
cDefault_MsvCB_OptFlds	ST_AcsiSvOptionalFields	SampleSynchronized:=TRUE	Default initial value for all MSVCB “OptionFields” DA instances.
cDefault_UsvCB_OptFlds	ST_AcsiSvOptionalFields	SampleSynchronized:=TRUE	Default initial value for all USVCB “OptionFields” DA instances.
cDefault_Per	E_AcsiPer	E_AcsiPer.Hourly	Default initial value for all “perRs” or “chrPerRs” DA instances.
cDefault_Acs	E_AcsiAcs	E_AcsiAcs.Low	Default initial value for all “sptAcs” DA instances.
cPresence_AnalogueValue	E_AcsiAnalogueValuePresCond	E_AcsiAnalogueValuePresCond.both_CO_f	Default presence of the “AnalogueValue” data elements: “i” and/or “f”. In the status value “i” and “f” (both) are visible, in the control value only “f” is visible.
cPresence_Cell_xEnd	BOOL	FALSE	Default presence of “Cell.xEnd”.
cPresence_Cell_yStart	BOOL	FALSE	Default presence of “Cell.yStart”.
cPresence_Cell_yEnd	BOOL	FALSE	Default presence of “Cell.xEnd”.
cPresence_Point_zVal	BOOL	FALSE	Default presence of “Point.zVal”.
cPresence_RangeConfig_limDb	BOOL	FALSE	Default presence of “RangeConfig.limDb”.
cPresence_Unit_multiplier	BOOL	FALSE	Default presence of “Unit.multiplier”.
cPresence_ValWithTrans_transInd	BOOL	FALSE	Default presence of “ValWithTrans.transInd”.
cPresence_Vector_ang	BOOL	FALSE	Default presence of “Vector.ang”.
cPresence_Oper_AnalogueValue_operTm	BOOL	FALSE	Default presence of “Oper.ctlVal(AnalogueValue).operTm”
cPresence_Oper_BOOLEAN_operTm	BOOL	FALSE	Default presence of “Oper.ctlVal(BOOLEAN).operTm”
cPresence_Oper_CODEDENUM_operTm	BOOL	FALSE	Default presence of “Oper.ctlVal(CODEDENUM).operTm”
cPresence_Oper_ENUMERATED_operTm	BOOL	FALSE	Default presence of “Oper.ctlVal(ENUMERATED).operTm”
cPresence_Oper_INT32_operTm	BOOL	FALSE	Default presence of “Oper.ctlVal(INT32).operTm”

Name	Type	Value	Description
cPresence_Oper_INT8_operTm	BOOL	FALSE	Default presence of “Oper.ctlVal(INT8).operTm”
cPresence_SBOw_AnalogueValue_operTm	BOOL	FALSE	Default presence of “SBOw.ctlVal(AnalogueValue).operTm”
cPresence_SBOw_BOOLEAN_operTm	BOOL	FALSE	Default presence of “SBOw.ctlVal(BOOLEAN).operTm”
cPresence_SBOw_CODEDENUM_operTm	BOOL	FALSE	Default presence of “SBOw.ctlVal(CODEDENUM).operTm”
cPresence_SBOw_ENUMERATED_operTm	BOOL	FALSE	Default presence of “SBOw.ctlVal(ENUMERATED).operTm”
cPresence_SBOw_INT32_operTm	BOOL	FALSE	Default presence of “SBOw.ctlVal(INT32).operTm”
cPresence_SBOw_INT8_operTm	BOOL	FALSE	Default presence of “SBOw.ctlVal(INT8).operTm”
cPresence_Cancel_AnalogueValue_operTm	BOOL	FALSE	Default presence of “Cancel.ctlVal(AnalogueValue).operTm”
cPresence_Cancel_BOOLEAN_operTm	BOOL	FALSE	Default presence of “Cancel.ctlVal(BOOLEAN).operTm”
cPresence_Cancel_CODEDENUM_operTm	BOOL	FALSE	Default presence of “Cancel.ctlVal(CODEDENUM).operTm”
cPresence_Cancel_ENUMERATED_operTm	BOOL	FALSE	Default presence of “Cancel.ctlVal(ENUMERATED).operTm”
cPresence_Cancel_INT32_operTm	BOOL	FALSE	Default presence of “Cancel.ctlVal(INT32).operTm”
cPresence_Cancel_INT8_operTm	BOOL	FALSE	Default presence of “Cancel.ctlVal(INT8).operTm”
cPresence_BrCB_ResvTms	BOOL	FALSE	Default presence of “BRCB.ResvTms”
cPresence_BrCB_Owner	BOOL	FALSE	Default presence of “BRCB.Owner”
cPresence_UrCB_Owner	BOOL	FALSE	Default presence of “URCB.Owner”

Name	Type	Value	Description
cPresence_LCB_OptFlds	BOOL	FALSE	Default presence of “LCB.OptFlds”
cPresence_LCB_BufTm	BOOL	FALSE	Default presence of “LCB.BufTm”
cPresence_GoCB_MinTime	BOOL	FALSE	Default presence of “GOCB.MinTime”
cPresence_GoCB_MaxTime	BOOL	FALSE	Default presence of “GOCB.MaxTime”
cPresence_GoCB_FixedOffs	BOOL	FALSE	Default presence of “GOCB.FixedOffs”

6.7.2 Param_Gse

Namespace: Tc3_Gse [▶ 102]
Type: Parameter list
Library: Tc3_Gse (Tc3_Gse.compiled-library)

Parameter

Name	Type	Value	Description
cMaxGseCtxBufferFrames	UDINT(1..256)	32	Max. Number of buffered GSE telegrams (GOOSE, SV, GSSE) in the context buffer.
cMaxGseCtxBufferSize	UDINT(1600..320000)	32000	Max. byte size of the context buffer. This buffer is used to save the GSE telegrams (GOOSE, SV, GSSE) that were received in a task context other than the context of the IEC 61850 PLC task. GSE telegrams can be received in a different task context if several protocols use the same network adapter (e.g. GOOSE and EtherCAT Automation Protocol). The buffered GSE telegrams are always processed by the IEC 61850 PLC task.

6.7.3 Param_Sockets

Namespace: Tc3_Sockets [▶ 102]
Type: Parameter list
Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Parameter

Name	Type	Value	Description
cMaxSocketReceiveBufferSize	UDINT	16#20000	Max. Byte size of the receive buffer for the client/server telegrams.
cDefaultReceiverThrottleTimes	T_ThrottleTimes [▶ 530]	[T#0S, T#10MS, T#20MS, T#40MS, T#60MS, T#80MS, T#100MS, T#120MS, T#140MS, T#160MS, T#180MS, T#200MS, T#250MS]	Default times for throttling the polling cycles/ retrieval frequency of the received TCP/IP telegrams (client/server communication).

6.8 Global variables

6.8.1 TC_EVENTS constants

TC_EVENTS is a global TwinCAT system variable list (GVL). Each constant in this list represents an event class that is available on a TwinCAT system. The instances of the event classes are structured data types. Their member variables are instances of individual events. The events can be used, for example, in a PLC project or in a PLC library for logging the most diverse messages. The TwinCAT system has several predefined event classes. Further event classes are brought along by the integrated PLC libraries (functions) or defined by the user in the PLC application. The Global Variable List of the event classes is automatically created and updated by the TwinCAT system. The number of available event classes can therefore vary from system to system.

Namespace: TwinCAT TypeSystem [▶ 101].TC_EVENTS
Type: Global Variable List (GVL)
Library: Not required. TwinCAT System Resource from v3.1.4022.14

TwinCAT system event classes

Name	Type
TcSystemEventClass	ST_TcSystemEventClass [▶ 591]
TcGeneralAdsEventClass	ST_TcGeneralAdsEventClass [▶ 574]
TcRouterEventClass	ST_TcRouterEventClass [▶ 585]
TcRTimeEventClass	ST_TcRTimeEventClass [▶ 586]
Win32EventClass	ST_Win32EventClass [▶ 600]

TF6510 event classes

Name	Type
TcIEC61850EventClass	ST_TcIEC61850EventClass [▶ 578]
TcScsmEventClass	ST_TcScsmEventClass [▶ 588]
TcMmsAbortEventClass	ST_TcMmsAbortEventClass [▶ 578]
TcMmsAssociationEventClass	ST_TcMmsAssociationEventClass [▶ 578]
TcMmsClientEventClass	ST_TcMmsClientEventClass [▶ 579]
TcMmsIncomingConnectEventClass	ST_TcMmsIncomingConnectEventClass [▶ 579]
TcMmsIncomingDisconnectEventClass	ST_TcMmsIncomingDisconnectEventClass [▶ 579]
TcMmsOutgoingConnectEventClass	ST_TcMmsOutgoingConnectEventClass [▶ 580]
TcMmsOutgoingDisconnectEventClass	ST_TcMmsOutgoingDisconnectEventClass [▶ 580]
TcMmsReceiverEventClass	ST_TcMmsReceiverEventClass [▶ 581]
TcMmsSenderEventClass	ST_TcMmsSenderEventClass [▶ 581]
TcMmsTransactionEventClass	ST_TcMmsTransactionEventClass [▶ 581]
TcUlosiAbortEventClass	ST_TcUlosiAbortEventClass [▶ 594]
TcUlosiAssociationEventClass	ST_TcUlosiAssociationEventClass [▶ 595]
TcUlosiClientEventClass	ST_TcUlosiClientEventClass [▶ 595]
TcUlosiIncomingConnectEventClass	ST_TcUlosiIncomingConnectEventClass [▶ 595]
TcUlosiIncomingDisconnectEventClass	ST_TcUlosiIncomingDisconnectEventClass [▶ 596]
TcUlosiOutgoingConnectEventClass	ST_TcUlosiOutgoingConnectEventClass [▶ 655]
TcUlosiOutgoingDisconnectEventClass	ST_TcUlosiOutgoingDisconnectEventClass [▶ 596]
TcUlosiReceiverEventClass	ST_TcUlosiReceiverEventClass [▶ 596]
TcUlosiSenderEventClass	ST_TcUlosiSenderEventClass [▶ 597]
TcUlosiTransactionEventClass	ST_TcUlosiTransactionEventClass [▶ 597]
TcUlosiServerEventClass	ST_TcUlosiServerEventClass [▶ 597]
TcUlosiConnectionEventClass	ST_TcUlosiConnectionEventClass [▶ 598]
TcUlosiAcseAbrtDiagnosticEventClass	ST_TcUlosiAcseAbrtDiagnosticEventClass [▶ 598]
TcUlosiAcseAareDiagnosticEventClass	ST_TcUlosiAcseAareDiagnosticEventClass [▶ 598]
TcUlosiSpduReasonCodeEventClass	ST_TcUlosiSpduReasonCodeEventClass [▶ 599]
TcRfc1006AbortEventClass	ST_TcRfc1006AbortEventClass [▶ 581]
TcRfc1006AssociationEventClass	ST_TcRfc1006AssociationEventClass [▶ 582]
TcRfc1006ClientEventClass	ST_TcRfc1006ClientEventClass [▶ 582]
TcRfc1006IncomingConnectEventClass	ST_TcRfc1006IncomingConnectEventClass [▶ 582]
TcRfc1006IncomingDisconnectEventClass	ST_TcRfc1006IncomingDisconnectEventClass [▶ 583]
TcRfc1006OutgoingConnectEventClass	ST_TcRfc1006OutgoingConnectEventClass [▶ 583]
TcRfc1006OutgoingDisconnectEventClass	ST_TcRfc1006OutgoingDisconnectEventClass [▶ 583]
TcRfc1006ReceiverEventClass	ST_TcRfc1006ReceiverEventClass [▶ 584]
TcRfc1006SenderEventClass	ST_TcRfc1006SenderEventClass [▶ 584]
TcRfc1006TransactionEventClass	ST_TcRfc1006TransactionEventClass [▶ 584]
TcTpktAbortEventClass	ST_TcTpktAbortEventClass [▶ 592]
TcTpktAssociationEventClass	ST_TcTpktAssociationEventClass [▶ 593]
TcTpktPduEventClass	ST_TcTpktPduEventClass [▶ 593]

Name	Type
TcTpktReceiverEventClass	ST_TcTpktReceiverEventClass [▶ 593]
TcTpktSenderEventClass	ST_TcTpktSenderEventClass [▶ 594]
TcTpktTransactionEventClass	ST_TcTpktTransactionEventClass [▶ 594]
TcSocketsAbortEventClass	ST_TcSocketsAbortEventClass [▶ 588]
TcSocketsAssociationEventClass	ST_TcSocketsAssociationEventClass [▶ 588]
TcSocketsClientEventClass	ST_TcSocketsClientEventClass [▶ 589]
TcSocketsIncomingConnectEventClass	ST_TcSocketsIncomingConnectEventClass [▶ 589]
TcSocketsIncomingDisconnectEventClass	ST_TcSocketsIncomingDisconnectEventClass [▶ 589]
TcSocketsOutgoingConnectEventClass	ST_TcSocketsOutgoingConnectEventClass [▶ 589]
TcSocketsOutgoingDisconnectEventClass	ST_TcSocketsOutgoingDisconnectEventClass [▶ 590]
TcSocketsReceiverEventClass	ST_TcSocketsReceiverEventClass [▶ 590]
TcSocketsSenderEventClass	ST_TcSocketsSenderEventClass [▶ 590]
TcSocketsTransactionEventClass	ST_TcSocketsTransactionEventClass [▶ 591]
TcSocketsWIn32SrvEventClass	ST_TcSocketsWin32SrvEventClass [▶ 591]
TcAcsiEventClass	ST_TcAcsiEventClass [▶ 573]

6.8.1.1 ST_TcAcsiEventClass

The table below contains a list of the events of the ACSI event class.

Namespace: TC_EVENTS [▶ 570].TcAcsiEventClass

Type: Global Variable

Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

```
TYPE ST_TcAcsiEventClass:
STRUCT
        NoError                              : TcEventEntry;
        InstanceNotAvailable                 : TcEventEntry;
        InstanceInUse                        : TcEventEntry;
        AccessViolation                      : TcEventEntry;
        AccessNotAllowedInCurrentState       : TcEventEntry;
        ParameterValueInappropriate          : TcEventEntry;
        ParameterValueInconsistent           : TcEventEntry;
        ClassNotSupported                    : TcEventEntry;
        InstanceLockedByOtherClient          : TcEventEntry;
        ControlMustBeSelected                : TcEventEntry;
        TypeConflict                         : TcEventEntry;
        FailedDueToCommunicationsConstraint : TcEventEntry;
        FailedDueToServerConstraint          : TcEventEntry;
END_STRUCT
END_TYPE
```

Events

Name	ID	Severity	Description
NoError	0	Verbose	
InstanceNotAvailable	1	Error	
InstanceInUse	2	Error	
AccessViolation	3	Error	
AccessNotAllowedInCurrentState	4	Error	
ParameterValueInappropriate	5	Error	
ParameterValueInconsistent	6	Error	
ClassNotSupported	7	Error	
InstanceLockedByOtherClient	8	Error	
ControlMustBeSelected	9	Error	
TypeConflict	10	Error	
FailedDueToCommunicationsConstraint	11	Error	
FailedDueToServerConstraint	12	Error	

6.8.1.2 ST_TcGeneralAdsEventClass

The table below contains a list of the events of the TwinCAT ADS event class.

Namespace: TC_EVENTS [▶ 570].TcGeneralAdsEventClass

Type: Global Variable

Library: Not required, TwinCAT System Resource from v3.1.4022.14

Events

Name	ID		Severity	Description
	Hex	Dec		
ServiceNotSupported	0x701	1793	Error	Service is not supported by the server
InvalidIndexGroup	0x702	1794	Error	Invalid index group
InvalidIndexOffset	0x703	1795	Error	Invalid index offset
InvalidAccess	0x704	1796	Error	Reading and writing not permitted.
InvalidSize	0x705	1797	Error	Parameter size not correct
InvalidData	0x706	1798	Error	Invalid parameter values
NotReady	0x707	1799	Error	Device is not ready to operate
Busy	0x708	1800	Error	Device is busy
InvalidContext	0x709	1801	Error	Invalid context (must be in Windows)
NoMemory	0x70A	1802	Error	Insufficient memory
InvalidParam	0x70B	1803	Error	Invalid parameter values
NotFound	0x70C	1804	Error	Not found (files, ...)
Syntax	0x70D	1805	Error	Syntax error in file or command
Incompatible	0x70E	1806	Error	Objects do not match
AlreadyExists	0x70F	1807	Error	Object already exists
SymbolNotFound	0x710	1808	Error	Symbol not found
SymbolVersionInvalid	0x711	1809	Error	Symbol version invalid
InvalidState	0x712	1810	Error	Device in invalid state
TransModeNotSupported	0x713	1811	Error	AdsTransMode not supported
NotificationHandleInvalid	0x714	1812	Error	Notification handle is invalid
ClientUnknown	0x715	1813	Error	Notification client not registered
NoMoreHandles	0x716	1814	Error	No further notification handle
InvalidWatchSize	0x717	1815	Error	Notification size too large
NotInit	0x718	1816	Error	Device not initialized
DeviceTimeout	0x719	1817	Error	Device has a timeout
NoInterface	0x71A	1818	Error	Interface query failed
InvalidInterface	0x71B	1819	Error	Wrong interface requested
InvalidClassID	0x71C	1820	Error	Class ID is invalid
InvalidObjectID	0x71D	1821	Error	Object ID is invalid
Pending	0x71E	1822	Error	Requirement still outstanding
Aborted	0x71F	1823	Error	Request is aborted

Name	ID		Severity	Description
	Hex	Dec		
SignalWarning	0x720	1824	Error	Signal warning
InvalidArrayIndex	0x721	1825	Error	Invalid array index
SymbolNotActive	0x722	1826	Error	Symbol not active
AccessDenied	0x723	1827	Error	Access refused
LicenseMissing	0x724	1828	Error	Missing license
LicenseExpired	0x725	1829	Error	License expired
LicenseExceeded	0x726	1830	Error	License exceeded
LicenseInvalid	0x727	1831	Error	License invalid
LicenseSystemID	0x728	1832	Error	System ID license invalid
LicenseNoTimeLimit	0x729	1833	Error	License not limited in time
LicenseFutureIssue	0x72A	1834	Error	License problem: Time in the future
LicenseTimeTooLong	0x72B	1835	Error	License period too long
Exception	0x72c	1836	Error	Exception at system startup
LicenseDuplicated	0x72D	1837	Error	License file read twice
	0x72E	1838	Error	Invalid signature
	0x72F	1839	Error	Public certificate
	0x740	1856	Error	Client error
InvalidServiceParam	0x741	1857	Error	Service contains an invalid parameter
ListEmpty	0x742	1858	Error	Polling list is empty
VariableConAlreadyInUse	0x743	1859	Error	Var connection already in use
InvokeIDAlreadyInUse	0x744	1860	Error	The called ID is already in use
TimeoutElapsed	0x745	1861	Error	Timeout has occurred
Win32Error	0x746	1862	Error	Error in Win32 subsystem
TimeoutInvalid	0x747	1863	Error	Invalid client timeout value
PortNotOpen	0x748	1864	Error	ADS port not opened
NoAMSAddr	0x749	1865	Error	
SyncInternalError	0x750	1872	Error	Internal error in Ads syn
AddHash	0x751	1873	Error	Hash table overflow
RemoveHash	0x752	1874	Error	Key not found in the hash
NoMoreSymbols	0x753	1875	Error	No further symbols in the cache
SyncResInvalid	0x754	1876	Error	Invalid response received
SyncPortLocked	0x755	1877	Error	Sync Port is disabled

6.8.1.3 ST_TcIEC61850EventClass

The table below contains a list of the events of the IEC 61850 event class.

Namespace: TC_EVENTS [▶ 570].TcIEC61850EventClass

Type: Global Variable

Library: Tc3_iec61850 (Tc3_iec61850.compiled-library)

Events

Name	ID	Severity	Description
InvalidParameterValue	1	Error	
InvalidObjectState	2	Error	
InvalidMmsObject	3	Error	
InvalidAssociationObject	4	Error	
ObjectNotFound	5	Error	
InvalidIEDObject	6	Error	

6.8.1.4 ST_TcMmsAbortEventClass

The table below contains a list of the events of the MMS event class for the connection termination.

Namespace: TC_EVENTS [▶ 570].TcMmsAbortEventClass

Type: Global Variable

Library: Tc3_Mms (Tc3_Mms.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	
InvalidUlosiObject	2	Error	

6.8.1.5 ST_TcMmsAssociationEventClass

The table below contains a list of the events of the MMS event class for the management of a connection (Association).

Namespace: TC_EVENTS [▶ 570].TcMmsAssociationEventClass

Type: Global Variable

Library: Tc3_Mms (Tc3_Mms.compiled-library)

Events

Name	ID	Severity	Description
StateTransitionError	1	Error	
StateValidationError	2	Error	
DecoderFailure	3	Error	
EncoderFailure	4	Error	
InvalidAssociationObject	5	Error	
InvalidPduObject	6	Error	

6.8.1.6 ST_TcMmsClientEventClass

The table below contains a list of the events of the MMS event class for the management of a client instance.

Namespace: TC_EVENTS [▶ 570].TcMmsClientEventClass

Type: Global Variable

Library: Tc3_Mms (Tc3_Mms.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	

6.8.1.7 ST_TcMmsIncomingConnectEventClass

The table below contains a list of the events of the MMS event class for the incoming connection.

Namespace: TC_EVENTS [▶ 570].TcMmsIncomingConnectEventClass

Type: Global Variable

Library: Tc3_Mms (Tc3_Mms.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidConnectionObject	3	Error	
InvalidSettingsObject	4	Error	
InvalidUlosiObject	5	Error	
InvalidServerObject	6	Error	
InitiateIndTimeoutError	7	Error	
InitiateRspTimeoutError	8	Error	
PduCreationError	9	Error	

6.8.1.8 ST_TcMmsIncomingDisconnectEventClass

The table below contains a list of the events of the MMS event class for the incoming disconnection.

Namespace: TC_EVENTS [▶ 570].TcMmsIncomingDisconnectEventClass

Type: Global Variable

Library: Tc3_Mms (Tc3_Mms.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidConnectionObject	3	Error	
InvalidUlosiObject	4	Error	
InvalidClientObject	5	Error	
ConcludeRspTimeoutError	6	Error	
PduCreationError	7	Error	

6.8.1.9 ST_TcMmsOutgoingConnectEventClass

The table below contains a list of the events of the MMS event class for the outgoing connection.

Namespace: TC_EVENTS [▶ 570].TcMmsOutgoingConnectEventClass

Type: Global Variable

Library: Tc3_Mms (Tc3_Mms.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidClientObject	3	Error	
InvalidSettingsObject	4	Error	
InvalidUlosiObject	5	Error	
InitiateCnfTimeoutError	6	Error	
PduCreationError	7	Error	

6.8.1.10 ST_TcMmsOutgoingDisconnectEventClass

The table below contains a list of the events of the MMS event class for the outgoing disconnection.

Namespace: TC_EVENTS [▶ 570].TcMmsOutgoingDisconnectEventClass

Type: Global Variable

Library: Tc3_Mms (Tc3_Mms.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidConnectionObject	3	Error	
InvalidSettingsObject	4	Error	
InvalidUlosiObject	5	Error	
InvalidClientObject	6	Error	
ConcludeCnfTimeoutError	7	Error	
PduCreationError	8	Error	

6.8.1.11 ST_TcMmsReceiverEventClass

The table below contains a list of the events of the MMS event class for the reception of the data.

Namespace: TC_EVENTS [▶ 570].TcMmsReceiverEventClass

Type: Global Variable

Library: Tc3_Mms (Tc3_Mms.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	
InvalidSettingsObject	2	Error	
InvalidUlosiObject	3	Error	
InvalidPrimitiveState	4	Error	
ResponseTimeoutError	5	Error	
DecoderFailure	6	Error	

6.8.1.12 ST_TcMmsSenderEventClass

The table below contains a list of the events of the MMS event class for the transmission of the data.

Namespace: TC_EVENTS [▶ 570].TcMmsSenderEventClass

Type: Global Variable

Library: Tc3_Mms (Tc3_Mms.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	

6.8.1.13 ST_TcMmsTransactionEventClass

The table below contains a list of the events of the MMS event class for the management of the data transactions.

Namespace: TC_EVENTS [▶ 570].TcMmsTransactionEventClass

Type: Global Variable

Library: Tc3_Mms (Tc3_Mms.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	
InvalidSettingsObject	2	Error	
InvalidUlosiObject	3	Error	
EncoderFailure	4	Error	
TransactionTimeoutError	5	Error	

6.8.1.14 ST_TcRfc1006AbortEventClass

The table below contains a list of the events of the RFC 1006 event class for the connection termination.

Namespace: TC_EVENTS [▶ 570].TcRfc1006AbortEventClass

Type: Global Variable

Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	
InvalidTpktObject	2	Error	

6.8.1.15 ST_TcRfc1006AssociationEventClass

The table below contains a list of the events of the RFC 1006 event class for the management of a connection (Association).

Namespace: TC_EVENTS [▶ 570].TcRfc1006AssociationEventClass

Type: Global Variable

Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Events

Name	ID	Severity	Description
StateTransitionError	1	Error	
StateValidationError	2	Error	

6.8.1.16 ST_TcRfc1006ClientEventClass

The table below contains a list of the events of the RFC 1006 event class for the management of a client instance.

Namespace: TC_EVENTS [▶ 570].TcRfc1006ClientEventClass

Type: Global Variable

Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	

6.8.1.17 ST_TcRfc1006IncomingConnectEventClass

The table below contains a list of the events of the RFC 1006 event class for the incoming connection.

Namespace: TC_EVENTS [▶ 570].TcRfc1006IncomingConnectEventClass

Type: Global Variable

Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidSettingsObject	3	Error	
InvalidTpktObject	4	Error	
InvalidConnectionObject	5	Error	
InvalidServerObject	6	Error	
CRTimeoutError	7	Error	
CCTimeoutError	8	Error	

6.8.1.18 ST_TcRfc1006IncomingDisconnectEventClass

The table below contains a list of the events of the RFC 1006 event class for the incoming disconnection.

Namespace: TC_EVENTS [▶ 570].TcRfc1006IncomingDisconnectEventClass

Type: Global Variable

Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidSettingsObject	3	Error	
InvalidTpktObject	4	Error	
DRTimeoutError	5	Error	

6.8.1.19 ST_TcRfc1006OutgoingConnectEventClass

The table below contains a list of the events of the RFC 1006 event class for the outgoing connection.

Namespace: TC_EVENTS [▶ 570].TcRfc1006OutgoingConnectEventClass

Type: Global Variable

Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidSettingsObject	3	Error	
InvalidTpktObject	4	Error	
InvalidClientObject	5	Error	
CCTimeoutError	6	Error	

6.8.1.20 ST_TcRfc1006OutgoingDisconnectEventClass

The table below contains a list of the events of the RFC 1006 event class for the outgoing disconnection.

Namespace: TC_EVENTS [▶ 570].TcRfc1006OutgoingDisconnectEventClass

Type: Global Variable

Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidTpktObject	3	Error	

6.8.1.21 ST_TcRfc1006ReceiverEventClass

The table below contains a list of the events of the RFC 1006 event class for the reception of the data.

Namespace: TC_EVENTS [▶ 570].TcRfc1006ReceiverEventClass

Type: Global Variable

Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	
InvalidSettingsObject	2	Error	
InvalidTpktObject	3	Error	
InvalidPrimitiveState	4	Error	
DataRspTimeoutError	5	Error	
DecoderFailure	6	Error	

6.8.1.22 ST_TcRfc1006SenderEventClass

The table below contains a list of the events of the RFC 1006 event class for the transmission of the data.

Namespace: TC_EVENTS [▶ 570].TcRfc1006SenderEventClass

Type: Global Variable

Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
DataCnfTimeoutError	2	Error	

6.8.1.23 ST_TcRfc1006TransactionEventClass

The table below contains a list of the events of the RFC 1006 event class for the management of the data transactions.

Namespace: TC_EVENTS [▶ 570].TcRfc1006TransactionEventClass

Type: Global Variable

Library: Tc3_Rfc1006 (Tc3_Rfc1006.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	
InvalidTpktObject	2	Error	
EncoderFailure	3	Error	

6.8.1.24 ST_TcRouterEventClass

The table below contains a list of the events of the TwinCAT Router event class.

Namespace: TC_EVENTS [▶ 570].TcRouterEventClass

Type: Global Variable

Library: Not required, TwinCAT System Resource from v3.1.4022.14

Events

Name	ID		Severity	Description
	Hex	Dec		
NoLockedMemory	0x500	1280	Error	Locked memory cannot be allocated.
ResizeMemory	0x501	1281	Error	The size of the router memory cannot be changed.
MailboxFull	0x502	1282	Error	The mailbox has reached the maximum number of possible messages. The currently sent message was rejected.
DebugBoxFull	0x503	1283	Error	The mailbox has reached the maximum number of possible messages. The sent message will not be displayed in the ADS monitor.
UnknownPortType	0x504	1284	Error	The port type is unknown.
NotInitialized	0x505	1285	Error	The router is not initialized.
PortAlreadyInUse	0x506	1286	Error	The desired port number has already been issued.
NotRegistered	0x507	1287	Error	The port is not registered.
NoMoreQueues	0x508	1288	Error	The maximum number of ports has been reached.
InvalidPort	0x509	1289	Error	The port is invalid.
NotActivated	0x50A	1290	Error	The router is not active.
FragmentBoxFull	0x50B	1291	Error	
FragmentTimeout	0x50C	1292	Error	
ToBeRemoved	0x50D	1293	Error	

6.8.1.25 ST_TcRTimeEventClass

The table below contains a list of the events of the TwinCAT RTime event class.

Namespace: TC_EVENTS [▶ 570].TcRTimeEventClass

Type: Global Variable

Library: Not required, TwinCAT System Resource from v3.1.4022.14

Events

Name	ID		Severity	Description
	Hex	Dec		
InternalError	0x1000	4096	Error	Internal error in the TwinCAT real-time system.
BadTimerPeriods	0x1001	4097	Error	Timer value is not valid.
InvalidTaskPtr	0x1002	4098	Error	Task pointer has the invalid value 0 (zero).
InvalidStackPtr	0x1003	4099	Error	Task stack pointer has the invalid value 0.
PrioExists	0x1004	4100	Error	The request task priority is already assigned.
NoMoreTCB	0x1005	4101	Error	No free TCB (Task Control Block) available. The maximum number of TCBs is 64.
NoMoreSemas	0x1006	4102	Error	No free semaphores available. The maximum number of semaphores is 64.
NoMoreQueues	0x1007	4103	Error	No free space available in the queue. The maximum number of positions in the queue is 64.
ExtIRQAlreadyDef	0x100D	4109	Error	An external synchronization interrupt is already applied.
ExtIRQNotDef	0x100E	4110	Error	No external synchronization interrupt applied.
ExtIRQInstallFailed	0x100F	4111	Error	Application of the external synchronization interrupt failed
IRQNotLessOrEqual	0x1010	4112	Error	Call of a service function in the wrong context
VMXNotSupported	0x1017	4119	Error	Intel VT-x extension is not supported.
VMXDisabled	0x1018	4120	Error	Intel VT-x extension is not enabled in the BIOS.
VMXControlsMissing	0x1019	4121	Error	Missing function in Intel VT-x extension.
VMXEnableFails	0x101A	4122	Error	Activation of Intel VT-x fails.

6.8.1.26 ST_TcScsmEventClass

The table below contains a list of the events of the SCSM event class.

Namespace: TC_EVENTS [▶ 570].TcScsmEventClass

Type: Global Variable

Library: Tc3_iec61850_8_1 (Tc3_iec61850_8_1.compiled-library)

Events

Name	ID	Severity	Description
ConfirmedErrorPduReceived	1	Error	
RejectPduReceived	2	Error	
ReqAndCnfInvokeIDiffers	3	Error	
ConfirmationTimeout	4	Error	
InvalidObjectReference	5	Error	
InvalidAssociationObject	6	Error	
InvalidSettingsObject	7	Error	
InvalidMmsObject	8	Error	
InvalidClientObject	9	Error	
InvalidServerObject	10	Error	
InvalidConnectionObject	11	Error	
InvalidIEDObject	12	Error	

6.8.1.27 ST_TcSocketsAbortEventClass

The table below contains a list of the events of the Socket event class for the connection termination.

Namespace: TC_EVENTS [▶ 570].TcSocketsAbortEventClass

Type: Global Variable

Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	
InvalidTcpObject	2	Error	

6.8.1.28 ST_TcSocketsAssociationEventClass

The table below contains a list of the events of the Socket event class for the management of a connection association

Namespace: TC_EVENTS [▶ 570].TcSocketsAssociationEventClass

Type: Global Variable

Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Events

Name	ID	Severity	Description
InvalidStateTransition	1	Error	
InvalidEnvironmentState	2	Error	

6.8.1.29 ST_TcSocketsClientEventClass

The table below contains a list of the events of the Socket event class for the management of a client instance.

Namespace: TC_EVENTS [▶ 570].TcSocketsClientEventClass

Type: Global Variable

Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	

6.8.1.30 ST_TcSocketsIncomingConnectEventClass

The table below contains a list of the events of the Socket event class for the incoming connection.

Namespace: TC_EVENTS [▶ 570].TcSocketsIncomingConnectEventClass

Type: Global Variable

Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidTcpObject	3	Error	
InvalidConnectionObject	4	Error	
InvalidServerObject	5	Error	
InvalidSettingsObject	6	Error	
ConnectRspTimeoutError	7	Error	

6.8.1.31 ST_TcSocketsIncomingDisconnectEventClass

The table below contains a list of the events of the Socket event class for the incoming disconnection.

Namespace: TC_EVENTS [▶ 570].TcSocketsIncomingDisconnectEventClass

Type: Global Variable

Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidTcpObject	3	Error	
DisconnectRspTimeoutError	4	Error	

6.8.1.32 ST_TcSocketsOutgoingConnectEventClass

The table below contains a list of the events of the Socket event class for the outgoing connection.

Namespace: TC_EVENTS [▶ 570].TcSocketsOutgoingConnectEventClass

Type: Global Variable

Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidClientObject	3	Error	
InvalidSettingsObject	4	Error	
InvalidTcpObject	5	Error	

6.8.1.33 ST_TcSocketsOutgoingDisconnectEventClass

The table below contains a list of the events of the Socket event class for the outgoing disconnection.

Namespace: TC_EVENTS [▶ 570].TcSocketsOutgoingDisconnectEventClass

Type: Global Variable

Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidTcpObject	3	Error	

6.8.1.34 ST_TcSocketsReceiverEventClass

The table below contains a list of the events of the Socket event class for the reception of the data.

Namespace: TC_EVENTS [▶ 570].TcSocketsReceiverEventClass

Type: Global Variable

Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
AckTimeoutError	3	Error	
InvalidParameterValue	4	Error	
FatalError	5	Error	
RcvBufferFreeFailure	6	Error	

6.8.1.35 ST_TcSocketsSenderEventClass

The table below contains a list of the events of the Socket event class for the transmission of the data.

Namespace: TC_EVENTS [▶ 570].TcSocketsSenderEventClass

Type: Global Variable

Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Events

Name	ID	Severity	Description
SenderListFailure	1	Error	
InvalidPrimitiveState	2	Error	
InvalidTransactionObject	3	Error	

6.8.1.36 ST_TcSocketsTransactionEventClass

The table below contains a list of the events of the Socket event class for the management of the data transactions.

Namespace: TC_EVENTS [▶ 570].TcSocketsTransactionEventClass

Type: Global Variable

Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	
InvalidDataObject	2	Error	
InvalidPrimitiveState	3	Error	

6.8.1.37 ST_TcSocketsWin32SrvEventClass

The table below contains a list of the events of the TF6310 TCP/IP Server Win32 event class.

Namespace: TC_EVENTS [▶ 570].TcSocketsWin32SrvEventClass

Type: Global Variable

Library: Tc3_Sockets (Tc3_Sockets.compiled-library)

6.8.1.38 ST_TcSystemEventClass

The table below contains a list of the events of the TwinCAT System event class.

Namespace: TC_EVENTS [▶ 570].TcSystemEventClass

Type: Global Variable

Library: Not required, TwinCAT System Resource from v3.1.4022.14

Events

Name	ID		Severity	Description
	Hex	Dec		
InternalError	0x1	1	Error	Internal error
NoRTime	0x2	2	Error	No real-time
AllocationLockedMemoryError	0x3	3	Error	Allocation locked memory error
InsertMailboxError	0x4	4	Error	Mailbox full
WrongReceiveHMSG	0x5	5	Error	Wrong HMSG
TargetPortNotFound	0x6	6	Error	Target port not found
TargetMachineNotFound	0x7	7	Error	Target machine not found
UnknownCommandID	0x8	8	Error	Unknown command ID
BadTaskID	0x9	9	Error	Bad Task ID
NoIO	0xA	10	Error	No IO
UnknownAdsCommand	0xB	11	Error	Unknown ADS command
Win32Error	0xC	12	Error	Win32 error
PortNotConnected	0xD	13	Error	Port not closed
InvalidAdsLength	0xE	14	Error	Invalid ADS length
InvalidAdsNetID	0xF	15	Error	Invalid AMS Net ID
LowInstallationLevel	0x10	16	Error	low installation level
NoDebugAvailable	0x11	17	Error	No debugging available
PortDisabled	0x12	18	Error	Port disabled
PortAlreadyConnected	0x13	19	Error	Port already connected
AdsSyncWin32Error	0x14	20	Error	ADS Sync Win32 error
AdsSyncTimeout	0x15	21	Error	ADS Sync Timeout
AdsSyncAmsError	0x16	22	Error	ADS Sync AMS error
AdsSyncNoIndexMap	0x17	23	Error	No index map exists for ADS Sync
InvalidAdsPort	0x18	24	Error	Invalid ADS port
NoMemory	0x19	25	Error	No memory
TCPSendError	0x1A	26	Error	TCP send error
HostUnreachable	0x1B	27	Error	Host unreachable
InvalidAMSFragment	0x1C	28	Error	Invalid AMS fragment

6.8.1.39 ST_TcTpktAbortEventClass

The table below contains a list of the events of the TPKT event class for the connection termination.

Namespace: TC_EVENTS [▶ 570].TcTpktAbortEventClass

Type: Global Variable

Library: Tc3_Tpkt (Tc3_Tpkt.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	
InvalidSocketObject	2	Error	

6.8.1.40 ST_TcTpktAssociationEventClass

The table below contains a list of the events of the TPKT event class for the management of a connection association.

Namespace: TC_EVENTS [▶ 570].TcTpktAssociationEventClass

Type: Global Variable

Library: Tc3_Tpkt (Tc3_Tpkt.compiled-library)

Events

Name	ID	Severity	Description
InvalidEnvironmentState	1	Error	

6.8.1.41 ST_TcTpktPduEventClass

The table below contains a list of the events of the TPKT event class for the processing of the PDUs.

Namespace: TC_EVENTS [▶ 570].TcTpktPduEventClass

Type: Global Variable

Library: Tc3_Tpkt (Tc3_Tpkt.compiled-library)

Events

Name	ID	Severity	Description
InvalidInputStreamObject	1	Error	
InvalidOutputStreamObject	2	Error	
InvalidHeaderSize	3	Error	
InvalidHeaderVersion	4	Error	
InvalidHeaderReserved	5	Error	
InvalidHeaderLength	6	Error	

6.8.1.42 ST_TcTpktReceiverEventClass

The table below contains a list of the events of the TPKT event class for the reception of the data.

Namespace: TC_EVENTS [▶ 570].TcTpktReceiverEventClass

Type: Global Variable

Library: Tc3_Tpkt (Tc3_Tpkt.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidSocketObject	3	Error	
DataRspTimeoutError	4	Error	
DataResponseAbort	5	Error	

6.8.1.43 ST_TcTpktSenderEventClass

The table below contains a list of the events of the TPKT event class for the transmission of the data.

Namespace: TC_EVENTS [▶ 570].TcTpktSenderEventClass

Type: Global Variable

Library: Tc3_Tpkt (Tc3_Tpkt.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	
InvalidSocketObject	2	Error	
InvalidDataObject	3	Error	

6.8.1.44 ST_TcTpktTransactionEventClass

The table below contains a list of the events of the TPKT event class for the management of the data transactions.

Namespace: TC_EVENTS [▶ 570].TcTpktTransactionEventClass

Type: Global Variable

Library: Tc3_Tpkt (Tc3_Tpkt.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	
InvalidAssociationObject	2	Error	
InvalidSocketObject	3	Error	

6.8.1.45 ST_TcUlosiAbortEventClass

The table below contains a list of the events of the ULOSI event class for the connection termination.

Namespace: TC_EVENTS [▶ 570].TcUlosiAbortEventClass

Type: Global Variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	Invalid association object interface (null)
InvalidRfc1006Object	2	Error	Invalid RFC 1006 interface object (null)
Refuse_spdu_received	128	Error	Refuse-spdu received
Abort_accept_spdu_received	129	Error	Abort-Accept-spdu received
Rejected_permanent	130	Error	Associate response (AARE) result: rejected-permanent.
Rejected_transient	131	Error	Associate response (AARE) result: rejected-transient.

6.8.1.46 ST_TcUlosiAssociationEventClass

The table below contains a list of the events of the ULOSI event class for the management of a connection association.

Namespace: TC_EVENTS [▶ 570].TcUlosiAssociationEventClass

Type: Global Variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
StateTransitionError	1	Error	Environment state transition error
StateValidationError	2	Error	Environment state validation error
AuthenticationFailure	3	Error	Authentication failure

6.8.1.47 ST_TcUlosiClientEventClass

The table below contains a list of the events of the ULOSI event class for the management of a client instance.

Namespace: TC_EVENTS [▶ 570].TcUlosiClientEventClass

Type: Global Variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	Invalid association object interface (null)

6.8.1.48 ST_TcUlosiIncomingConnectEventClass

The table below contains a list of the events of the ULOSI event class for the incoming connection.

Namespace: TC_EVENTS [▶ 570].TcUlosiIncomingConnectEventClass

Type: Global Variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	Invalid primitive state
InvalidAssociationObject	2	Error	Invalid association object interface (null)
InvalidSettingsObject	3	Error	Invalid settings object interface (null)
InvalidRfc1006Object	4	Error	Invalid RFC 1006 object interface (null)
InvalidConnectionObject	5	Error	Invalid connection object interface (null)
InvalidServerObject	6	Error	Invalid server object interface (null)
AARQTimeoutError	7	Error	Application association request (AARQ) timeout error.
AARETimeoutError	8	Error	Application association response (AARE) timeout error.

6.8.1.49 ST_TcUlosiIncomingDisconnectEventClass

The table below contains a list of the events of the ULOSI event class for the incoming disconnection.

Namespace: TC_EVENTS [▶ 570].TcUlosiIncomingDisconnectEventClass

Type: Global Variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	Invalid primitive state.
InvalidAssociationObject	2	Error	Invalid association object interface (null)
InvalidSettingsObject	3	Error	Invalid settings object interface (null)
InvalidRfc1006Object	4	Error	Invalid RFC 1006 object interface (null)
RLRETimeoutError	5	Error	Release response timeout error
ConfirmationTimeoutError	6	Error	Confirmation timeout error

6.8.1.50 ST_TcUlosiOutgoingDisconnectEventClass

The table below contains a list of the events of the ULOSI event class for the outgoing disconnection.

Namespace: TC_EVENTS [▶ 570].TcUlosiOutgoingDisconnectEventClass

Type: Global Variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	Invalid primitive state.
InvalidAssociationObject	2	Error	Invalid association object interface (null).
InvalidRfc1006Object	3	Error	Invalid RFC 1006 object interface (null).
RLRETimeoutError	4	Error	Release response (RLRE) timeout error.

6.8.1.51 ST_TcUlosiReceiverEventClass

The table below contains a list of the events of the ULOSI event class for the reception of the data.

Namespace: TC_EVENTS [▶ 570].TcUlosiReceiverEventClass

Type: Global Variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	Invalid association object interface (null).
InvalidSettingsObject	2	Error	Invalid settings object interface (null).
InvalidRfc1006Object	3	Error	Invalid RFC 1006 object interface (null).
InvalidPrimitiveState	4	Error	Invalid primitive state.
ResponseTimeoutError	5	Error	Response timeout error.
DecoderFailure	6	Error	Decoder failure.

6.8.1.52 ST_TcUlosiSenderEventClass

The table below contains a list of the events of the ULOSI event class for the transmission of the data.

Namespace: TC_EVENTS [▶ 570].TcUlosiSenderEventClass

Type: Global Variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	Invalid primitive state.

6.8.1.53 ST_TcUlosiTransactionEventClass

The table below contains a list of the events of the ULOSI event class for the management of the data transactions.

Namespace: TC_EVENTS [▶ 570].TcUlosiTransactionEventClass

Type: Global Variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
InvalidAssociationObject	1	Error	Invalid association object interface (null).
InvalidRfc1006Object	2	Error	Invalid RFC 1006 object interface (null).
EncoderFailure	3	Error	Encoder failure.

6.8.1.54 ST_TcUlosiServerEventClass

The following table contains a list of the events of the ULOSI event class for the management of a server instance (connection container).

Namespace: TC_EVENTS [▶ 570].TcUlosiServerEventClass

Type: Global variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
InvalidParameterValue	1	Error	Invalid parameter value.
InvalidObjectState	2	Error	Invalid connection object state.
ObjectNotFound	3	Error	Connection object not found.
InvalidRfc1006Object	4	Error	Invalid RFC1006 object interface (null).
InvalidAssociationObject	5	Error	Invalid association object interface (null).

6.8.1.55 ST_TcUlosiConnectionEventClass

The following table contains a list of events of the ULOSI event class for the management of a server connection instance (session).

Namespace: TC_EVENTS [▶ 570].TcUlosiConnectionEventClass

Type: Global variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
InvalidParameterValue	1	Error	Invalid parameter value.
InvalidObjectState	2	Error	Invalid connection object state.
ObjectAlreadyExists	3	Error	Connection object already exists.
ObjectNotFound	4	Error	Connection object not found.

6.8.1.56 ST_TcUlosiAcseAbrtDiagnosticEventClass

The following table contains a list of the events of the ULOSI event class for the connection termination (ACSE-ABRT frame).

Namespace: TC_EVENTS [▶ 570].TcUlosiAcseAbrtDiagnosticEventClass

Type: Global variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
Null	0	Info	
No_reason_given	1	Error	No reason given.
Protocol_error	2	Error	Protocol error.
Authentication_mechanism_name_not_recognized	3	Error	Authentication mechanism name not recognized.
Authentication_mechanism_name_required	4	Error	Authentication mechanism name required.
Authentication_failure	5	Error	Authentication failure.
Authentication_required	6	Error	Authentication required.

6.8.1.57 ST_TcUlosiAcseAareDiagnosticEventClass

The following table contains a list of events of the ULOSI event class for the negative responses during connection establishment (ACSE-AARE frame).

Namespace: TC_EVENTS [▶ 570].TcUlosiAcseAareDiagnosticEventClass

Type: Global variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
Null	0	Info	
No_reason_given	1	Error	No reason given.
Application_context_name_not_supported	2	Error	Application context name not supported.
Calling_AP_title_not_recognized	3	Error	Calling AP title not recognized.
Calling_AP_invocation_identifier_not_recognized	4	Error	Calling AP invocation identifier not recognized.
Calling_AE_qualifier_not_recognized	5	Error	Calling AE qualifier not recognized.
Calling_AE_invocation_identifier_not_recognized	6	Error	Calling AE invocation identifier not recognized.
Called_AP_title_not_recognized	7	Error	Called AP title not recognized.
Called_AP_invocation_identifier_not_recognized	8	Error	Called AP invocation identifier not recognized.
Called_AE_qualifier_not_recognized	9	Error	Called AE qualifier not recognized.
Called_AE_invocation_identifier_not_recognized	10	Error	Called AE invocation identifier not recognized.
Authentication_mechnism_name_not_recognized	11	Error	Authentication mechanism name not recognized.
Authentication_mechanism_name_required	12	Error	Authentication mechanism name required.
Authentication_failure	13	Error	Authentication failure.
Authentication_required	14	Error	Authentication required.

6.8.1.58 ST_TcUlosiSpduReasonCodeEventClass

The following table contains a list of events of the ULOSI event class for rejected session layer telegrams (SPDU refuse frame).

Namespace: TC_EVENTS [▶ 570].TcUlosiSpduReasonCodeEventClass

Type: Global variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
Reason_not_specified	0	Error	Rejection by called SS-user, reason not specified.
Temporary_congestion	1	Error	Rejection by called SS-user due to temporary congestion.
Subsequent	2	Error	Rejection by called SS-user, subsequent octets may be used.
Session_selector_unknown	129	Error	Session selector unknown.
SS_user_not_attached_to_SSAP	130	Error	SS-user not attached to SSAP.
SPM_congestion_at_connect_time	131	Error	SPM congestion at connect time.
Versions_not_supported	132	Error	Proposed protocol versions not supported.
SPM_reason_not_specified	133	Error	Rejection by SPM, reason not specified.
SPM_implementation_restriction	134	Error	Rejection by the SPM, implementation restriction stated in the PICS.

6.8.1.59 ST_Win32EventClass

The table below contains a list of the events of the Win32 event class.

Events (IDs 0..99 [▶ 601]),
Events (IDs 100..199 [▶ 603]),
Events (IDs 200..1000 [▶ 605]),
Events (IDs 1001..1099 [▶ 607]),
Events (IDs 1100..1199 [▶ 609]),
Events (IDs 1200..1399 [▶ 611]),
Events (IDs 1400..1599 [▶ 616]),
Events (IDs 1600..1799 [▶ 618]),
Events (IDs 1800..1999 [▶ 622]),
Events (IDs 2000..2999 [▶ 624]),
Events (IDs 3000..4999 [▶ 625]),
Events (IDs 5000..5999 [▶ 628]),
Events (IDs 6000..7999 [▶ 631]),
Events (IDs 8000..8499 [▶ 633]),
Events (IDs 8500..9000 [▶ 642]),
Events (IDs 9001..10003 [▶ 646]),
Events (IDs 10004..11999 [▶ 648]),
Events (IDs 12000..13884 [▶ 651])

Namespace: TC_EVENTS [▶ 570].Win32EventClass

Type: Global Variable

Library: Not required, TwinCAT System Resource from v3.1.4022.14

Error			Description
decimal	Hexadecimal	Name	
0	0x00000000	ERROR_SUCCESS	The operation completed successfully.
1	0x00000001	ERROR_INVALID_FUNCTION	Incorrect function.
2	0x00000002	ERROR_FILE_NOT_FOUND	The system cannot find the file specified.
3	0x00000003	ERROR_PATH_NOT_FOUND	The system cannot find the path specified.
4	0x00000004	ERROR_TOO_MANY_OPEN_FILES	The system cannot open the file.
5	0x00000005	ERROR_ACCESS_DENIED	Access is denied.
6	0x00000006	ERROR_INVALID_HANDLE	The handle is invalid.
7	0x00000007	ERROR_ARENA_TRASHED	The storage control blocks were destroyed.
8	0x00000008	ERROR_NOT_ENOUGH_MEMORY	Not enough storage is available to process this command.
9	0x00000009	ERROR_INVALID_BLOCK	The storage control block address is invalid.
10	0x0000000A	ERROR_BAD_ENVIRONMENT	The environment is incorrect.
11	0x0000000B	ERROR_BAD_FORMAT	An attempt was made to load a program with an incorrect format.
12	0x0000000C	ERROR_INVALID_ACCESS	The access code is invalid.
13	0x0000000D	ERROR_INVALID_DATA	The data is invalid.
14	0x0000000E	ERROR_OUTOFMEMORY	Not enough storage is available to complete this operation.
15	0x0000000F	ERROR_INVALID_DRIVE	The system cannot find the drive specified.
16	0x00000010	ERROR_CURRENT_DIRECTORY	The directory cannot be removed.
17	0x00000011	ERROR_NOT_SAME_DEVICE	The system cannot move the file to a different disk drive.
18	0x00000012	ERROR_NO_MORE_FILES	There are no more files.
19	0x00000013	ERROR_WRITE_PROTECT	The media is write protected.
20	0x00000014	ERROR_BAD_UNIT	The system cannot find the device specified.
21	0x00000015	ERROR_NOT_READY	The device is not ready.
22	0x00000016	ERROR_BAD_COMMAND	The device does not recognize the command.
23	0x00000017	ERROR_CRC	Data error (cyclic redundancy check).
24	0x00000018	ERROR_BAD_LENGTH	The program issued a command but the command length is incorrect.
25	0x00000019	ERROR_SEEK	The drive cannot locate a specific area or track on the disk.
26	0x0000001A	ERROR_NOT_DOS_DISK	The specified disk or diskette cannot be accessed.
27	0x0000001B	ERROR_SECTOR_NOT_FOUND	The drive cannot find the sector requested.
28	0x0000001C	ERROR_OUT_OF_PAPER	The printer is out of paper.
29	0x0000001D	ERROR_WRITE_FAULT	The system cannot write to the specified device.
30	0x0000001E	ERROR_READ_FAULT	The system cannot read from the specified device.
31	0x0000001F	ERROR_GEN_FAILURE	A device attached to the system is not functioning.
32	0x00000020	ERROR_SHARING_VIOLATION	The process cannot access the file because it is being used by another process.
33	0x00000021	ERROR_LOCK_VIOLATION	The process cannot access the file because another process has locked a portion of the file.
34	0x00000022	ERROR_WRONG_DISK	The wrong diskette is in the drive. Insert %2 (Volume Serial Number: %3) into drive %1.
36	0x00000024	ERROR_SHARING_BUFFER_EXCEEDED	Too many files opened for sharing.
38	0x00000026	ERROR_HANDLE_EOF	Reached the end of the file.
39	0x00000027	ERROR_HANDLE_DISK_FULL	The disk is full.
50	0x00000032	ERROR_NOT_SUPPORTED	The request is not supported.
51	0x00000033	ERROR_REM_NOT_LIST	The remote computer is not available.
52	0x00000034	ERROR_DUP_NAME	A duplicate name exists on the network.
53	0x00000035	ERROR_BAD_NETPATH	The network path was not found.
54	0x00000036	ERROR_NETWORK_BUSY	The network is busy.
55	0x00000037	ERROR_DEV_NOT_EXIST	The specified network resource or device is no longer available.
56	0x00000038	ERROR_TOO_MANY_CMDS	The network BIOS command limit has been reached.
57	0x00000039	ERROR_ADAP_HDW_ERR	A network adapter hardware error occurred.

Error			Description
decimal	Hexadecimal	Name	
58	0x0000003A	ERROR_BAD_NET_RESP	The specified server cannot perform the requested operation.
59	0x0000003B	ERROR_UNEXP_NET_ERR	An unexpected network error occurred.
60	0x0000003C	ERROR_BAD_REM_ADAP	The remote adapter is not compatible.
61	0x0000003D	ERROR_PRINTQ_FULL	The printer queue is full.
62	0x0000003E	ERROR_NO_SPOOL_SPACE	Space to store the file waiting to be printed is not available on the server.
63	0x0000003F	ERROR_PRINT_CANCELLED	Your file waiting to be printed was deleted.
64	0x00000040	ERROR_NETNAME_DELETED	The specified network name is no longer available.
65	0x00000041	ERROR_NETWORK_ACCESS_DENIED	Network access is denied.
66	0x00000042	ERROR_BAD_DEV_TYPE	The network resource type is not correct.
67	0x00000043	ERROR_BAD_NET_NAME	The network name cannot be found.
68	0x00000044	ERROR_TOO_MANY_NAMES	The name limit for the local computer network adapter card was exceeded.
69	0x00000045	ERROR_TOO_MANY_SESS	The network BIOS session limit was exceeded.
70	0x00000046	ERROR_SHARING_PAUSED	The remote server has been paused or is in the process of being started.
71	0x00000047	ERROR_REQ_NOT_ACCEP	No more connections can be made to this remote computer at this time because there are already as many connections as the computer can accept.
72	0x00000048	ERROR_REDIR_PAUSED	The specified printer or disk device has been paused.
80	0x00000050	ERROR_FILE_EXISTS	The file exists.
82	0x00000052	ERROR_CANNOT_MAKE	The directory or file cannot be created.
83	0x00000053	ERROR_FAIL_I24	Fail on INT 24.
84	0x00000054	ERROR_OUT_OF_STRUCTURES	Storage to process this request is not available.
85	0x00000055	ERROR_ALREADY_ASSIGNED	The local device name is already in use.
86	0x00000056	ERROR_INVALID_PASSWORD	The specified network password is not correct.
87	0x00000057	ERROR_INVALID_PARAMETER	The parameter is incorrect.
88	0x00000058	ERROR_NET_WRITE_FAULT	A write fault occurred on the network.
89	0x00000059	ERROR_NO_PROC_SLOTS	The system cannot start another process at this time.

Error			Description
decimal	Hexadecimal	Name	
100	0x00000064	ERROR_TOO_MANY_SEMAPHORES	Cannot create another system semaphore.
101	0x00000065	ERROR_EXCL_SEM_ALREADY_OWNED	The exclusive semaphore is owned by another process.
102	0x00000066	ERROR_SEM_IS_SET	The semaphore is set and cannot be closed.
103	0x00000067	ERROR_TOO_MANY_SEM_REQUESTS	The semaphore cannot be set again.
104	0x00000068	ERROR_INVALID_AT_INTERRUPT_TIME	Cannot request exclusive semaphores at interrupt time.
105	0x00000069	ERROR_SEM_OWNER_DIED	The previous ownership of this semaphore has ended.
106	0x0000006A	ERROR_SEM_USER_LIMIT	Insert the diskette for drive %1.
107	0x0000006B	ERROR_DISK_CHANGE	The program stopped because an alternate diskette was not inserted.
108	0x0000006C	ERROR_DRIVE_LOCKED	The disk is in use or locked by another process.
109	0x0000006D	ERROR_BROKEN_PIPE	The pipe has been ended.
110	0x0000006E	ERROR_OPEN_FAILED	The system cannot open the device or file specified.
111	0x0000006F	ERROR_BUFFER_OVERFLOW	The file name is too long.
112	0x00000070	ERROR_DISK_FULL	There is not enough space on the disk.
113	0x00000071	ERROR_NO_MORE_SEARCH_HANDLES	No more internal file identifiers available.
114	0x00000072	ERROR_INVALID_TARGET_HANDLE	The target internal file identifier is incorrect.
117	0x00000075	ERROR_INVALID_CATEGORY	The IOCTL call made by the application program is not correct.
118	0x00000076	ERROR_INVALID_VERIFY_SWITCH	The verify-on-write switch parameter value is not correct.
119	0x00000077	ERROR_BAD_DRIVER_LEVEL	The system does not support the command requested.
120	0x00000078	ERROR_CALL_NOT_IMPLEMENTED	This function is not supported on this system.
121	0x00000079	ERROR_SEM_TIMEOUT	The semaphore timeout period has expired.
122	0x0000007A	ERROR_INSUFFICIENT_BUFFER	The data area passed to a system call is too small.
123	0x0000007B	ERROR_INVALID_NAME	The filename, directory name, or volume label syntax is incorrect.
124	0x0000007C	ERROR_INVALID_LEVEL	The system call level is not correct.
125	0x0000007D	ERROR_NO_VOLUME_LABEL	The disk has no volume label.
126	0x0000007E	ERROR_MOD_NOT_FOUND	The specified module could not be found.
127	0x0000007F	ERROR_PROC_NOT_FOUND	The specified procedure could not be found.
128	0x00000080	ERROR_WAIT_NO_CHILDREN	There are no child processes to wait for.
129	0x00000081	ERROR_CHILD_NOT_COMPLETE	The %1 application cannot be run in Win32 mode.
130	0x00000082	ERROR_DIRECT_ACCESS_HANDLE	Attempt to use a file handle to an open disk partition for an operation other than raw disk I/O.
131	0x00000083	ERROR_NEGATIVE_SEEK	An attempt was made to move the file pointer before the beginning of the file.
132	0x00000084	ERROR_SEEK_ON_DEVICE	The file pointer cannot be set on the specified device or file.
133	0x00000085	ERROR_IS_JOIN_TARGET	A JOIN or SUBST command cannot be used for a drive that contains previously joined drives.
134	0x00000086	ERROR_IS_JOINED	An attempt was made to use a JOIN or SUBST command on a drive that has already been joined.
135	0x00000087	ERROR_IS_SUBSTED	An attempt was made to use a JOIN or SUBST command on a drive that has already been substituted.
136	0x00000088	ERROR_NOT_JOINED	The system tried to delete the JOIN of a drive that is not joined.
137	0x00000089	ERROR_NOT_SUBSTED	The system tried to delete the substitution of a drive that is not substituted.
138	0x0000008A	ERROR_JOIN_TO_JOIN	The system tried to join a drive to a directory on a joined drive.
139	0x0000008B	ERROR_SUBST_TO_SUBST	The system tried to substitute a drive to a directory on a substituted drive.
140	0x0000008C	ERROR_JOIN_TO_SUBST	The system tried to join a drive to a directory on a substituted drive.
141	0x0000008D	ERROR_SUBST_TO_JOIN	The system tried to SUBST a drive to a directory on a joined drive.

Error			Description
decimal	Hexadecimal	Name	
142	0x0000008E	ERROR_BUSY_DRIVE	The system cannot perform a JOIN or SUBST at this time.
143	0x0000008F	ERROR_SAME_DRIVE	The system cannot join or substitute a drive to or for a directory on the same drive.
144	0x00000090	ERROR_DIR_NOT_ROOT	The directory is not a subdirectory of the root directory.
145	0x00000091	ERROR_DIR_NOT_EMPTY	The directory is not empty.
146	0x00000092	ERROR_IS_SUBST_PATH	The path specified is being used in a substitute.
147	0x00000093	ERROR_IS_JOIN_PATH	Not enough resources are available to process this command.
148	0x00000094	ERROR_PATH_BUSY	The path specified cannot be used at this time.
149	0x00000095	ERROR_IS_SUBST_TARGET	An attempt was made to join or substitute a drive for which a directory on the drive is the target of a previous substitute.
150	0x00000096	ERROR_SYSTEM_TRACE	System trace information was not specified in your CONFIG.SYS file, or tracing is disallowed.
151	0x00000097	ERROR_INVALID_EVENT_COUNT	The number of specified semaphore events for DosMuxSemWait is not correct.
152	0x00000098	ERROR_TOO_MANY_MUXWAITERS	DosMuxSemWait did not execute; too many semaphores are already set.
153	0x00000099	ERROR_INVALID_LIST_FORMAT	The DosMuxSemWait list is not correct.
154	0x0000009A	ERROR_LABEL_TOO_LONG	The volume label you entered exceeds the label character limit of the target file system.
155	0x0000009B	ERROR_TOO_MANY_TCBS	Cannot create another thread.
156	0x0000009C	ERROR_SIGNAL_REFUSED	The recipient process has refused the signal.
157	0x0000009D	ERROR_DISCARDED	The segment is already discarded and cannot be locked.
158	0x0000009E	ERROR_NOT_LOCKED	The segment is already unlocked.
159	0x0000009F	ERROR_BAD_THREADID_ADDR	The address for the thread ID is not correct.
160	0x000000A0	ERROR_BAD_ARGUMENTS	The argument string passed to DosExecPgm is not correct.
161	0x000000A1	ERROR_BAD_PATHNAME	The specified path is invalid.
162	0x000000A2	ERROR_SIGNAL_PENDING	A signal is already pending.
164	0x000000A4	ERROR_MAX_THRDS_REACHED	No more threads can be created in the system.
167	0x000000A7	ERROR_LOCK_FAILED	Unable to lock a region of a file.
170	0x000000AA	ERROR_BUSY	The requested resource is in use.
173	0x000000AD	ERROR_CANCEL_VIOLATION	A lock request was not outstanding for the supplied cancel region.
174	0x000000AE	ERROR_ATOMIC_LOCKS_NOT_SUPPORTED	The file system does not support atomic changes to the lock type.
180	0x000000B4	ERROR_INVALID_SEGMENT_NUMBER	The system detected a segment number that was not correct.
182	0x000000B6	ERROR_INVALID_ORDINAL	The operating system cannot run %1.
183	0x000000B7	ERROR_ALREADY_EXISTS	Cannot create a file when that file already exists.
186	0x000000BA	ERROR_INVALID_FLAG_NUMBER	The flag passed is not correct.
187	0x000000BB	ERROR_SEM_NOT_FOUND	The specified system semaphore name was not found.
188	0x000000BC	ERROR_INVALID_STARTING_CODESEG	The operating system cannot run %1.
189	0x000000BD	ERROR_INVALID_STACKSEG	The operating system cannot run %1.
190	0x000000BE	ERROR_INVALID_MODULETYPE	The operating system cannot run %1.
191	0x000000BF	ERROR_INVALID_EXE_SIGNATURE	Cannot run %1 in Win32 mode.
192	0x000000C0	ERROR_EXE_MARKED_INVALID	The operating system cannot run %1.
193	0x000000C1	ERROR_BAD_EXE_FORMAT	%1 is not a valid Win32 application.
194	0x000000C2	ERROR_ITERATED_DATA_EXCEEDS_64k	The operating system cannot run %1.
195	0x000000C3	ERROR_INVALID_MINALLOCSIZE	The operating system cannot run %1.
196	0x000000C4	ERROR_DYNLINK_FROM_INVALID_RING	The operating system cannot run this application program.
197	0x000000C5	ERROR_IOPL_NOT_ENABLED	The operating system is not presently configured to run this application.
198	0x000000C6	ERROR_INVALID_SEGDPL	The operating system cannot run %1.
199	0x000000C7	ERROR_AUTODATASEG_EXCEEDS_64k	The operating system cannot run this application program.

Error			Description
decimal	Hexadecimal	Name	
200	0x000000C8	ERROR_RING2SEG_MUST_BE_MOVABLE	The code segment cannot be greater than or equal to 64K.
201	0x000000C9	ERROR_RELOC_CHAIN_XEEDS_SEGLIM	The operating system cannot run %1.
202	0x000000CA	ERROR_INFLOOP_IN_RELOC_CHAIN	The operating system cannot run %1.
203	0x000000CB	ERROR_ENVVAR_NOT_FOUND	The system could not find the environment option that was entered.
205	0x000000CD	ERROR_NO_SIGNAL_SENT	No process in the command subtree has a signal handler.
206	0x000000CE	ERROR_FILENAME_EXCED_RANGE	The filename or extension is too long.
207	0x000000CF	ERROR_RING2_STACK_IN_USE	The ring 2 stack is in use.
208	0x000000D0	ERROR_META_EXPANSION_TOO_LONG	The global filename characters, * or ?, are entered incorrectly or too many global filename characters are specified.
209	0x000000D1	ERROR_INVALID_SIGNAL_NUMBER	The signal being posted is not correct.
210	0x000000D2	ERROR_THREAD_1_INACTIVE	The signal handler cannot be set.
212	0x000000D4	ERROR_LOCKED	The segment is locked and cannot be reallocated.
214	0x000000D6	ERROR_TOO_MANY_MODULES	Too many dynamic-link modules are attached to this program or dynamic-link module.
215	0x000000D7	ERROR_NESTING_NOT_ALLOWED	Cannot nest calls to LoadModule.
216	0x000000D8	ERROR_EXE_MACHINE_TYPE_MISMATCH	The image file %1 is valid, but is for a machine type other than the current machine.
230	0x000000E6	ERROR_BAD_PIPE	The pipe state is invalid.
231	0x000000E7	ERROR_PIPE_BUSY	All pipe instances are busy.
232	0x000000E8	ERROR_NO_DATA	The pipe is being closed.
233	0x000000E9	ERROR_PIPE_NOT_CONNECTED	No process is on the other end of the pipe.
234	0x000000EA	ERROR_MORE_DATA	More data is available.
240	0x000000F0	ERROR_VC_DISCONNECTED	The session was canceled.
254	0x000000FE	ERROR_INVALID_EA_NAME	The specified extended attribute name was invalid.
255	0x000000FF	ERROR_EA_LIST_INCONSISTENT	The extended attributes are inconsistent.
258	0x00000102	WAIT_TIMEOUT	The wait operation timed out.
259	0x00000103	ERROR_NO_MORE_ITEMS	No more data is available.
266	0x0000010A	ERROR_CANNOT_COPY	The copy functions cannot be used.
267	0x0000010B	ERROR_DIRECTORY	The directory name is invalid.
275	0x00000113	ERROR_EAS_DIDNT_FIT	The extended attributes did not fit in the buffer.
276	0x00000114	ERROR_EA_FILE_CORRUPT	The extended attribute file on the mounted file system is corrupt.
277	0x00000115	ERROR_EA_TABLE_FULL	The extended attribute table file is full.
278	0x00000116	ERROR_INVALID_EA_HANDLE	The specified extended attribute handle is invalid.
282	0x0000011A	ERROR_EAS_NOT_SUPPORTED	The mounted file system does not support extended attributes.
288	0x00000120	ERROR_NOT_OWNER	Attempt to release mutex not owned by caller.
298	0x0000012A	ERROR_TOO_MANY_POSTS	Too many posts were made to a semaphore.
299	0x0000012B	ERROR_PARTIAL_COPY	Only part of a ReadProcessMemory or WriteProcessMemory request was completed.
300	0x0000012C	ERROR_OPLOCK_NOT_GRANTED	The oplock request is denied.
301	0x0000012D	ERROR_INVALID_OPLOCK_PROTOCOL	An invalid oplock acknowledgment was received by the system.
302	0x0000012E	ERROR_DISK_TOO_FRAGMENTED	The volume is too fragmented to complete this operation.
303	0x0000012F	ERROR_DELETE_PENDING	The file cannot be opened because it is in the process of being deleted.
317	0x0000013D	ERROR_MR_MID_NOT_FOUND	The system cannot find message text for message number 0x%1 in the message file for %2.
487	0x000001E7	ERROR_INVALID_ADDRESS	Attempt to access invalid address.
534	0x00000216	ERROR_ARITHMETIC_OVERFLOW	Arithmetic result exceeded 32 bits.
535	0x00000217	ERROR_PIPE_CONNECTED	There is a process on other end of the pipe.
536	0x00000218	ERROR_PIPE_LISTENING	Waiting for a process to open the other end of the pipe.
994	0x000003E2	ERROR_EA_ACCESS_DENIED	Access to the extended attribute was denied.

Error			Description
decimal	Hexadecimal	Name	
995	0x000003E3	ERROR_OPERATION_ABORTED	The I/O operation has been aborted because of either a thread exit or an application request.
996	0x000003E4	ERROR_IO_INCOMPLETE	Overlapped I/O event is not in a signaled state.
997	0x000003E5	ERROR_IO_PENDING	Overlapped I/O operation is in progress.
998	0x000003E6	ERROR_NOACCESS	Invalid access to memory location.
999	0x000003E7	ERROR_SWAPERROR	Error performing inpage operation.

Error			Description
decimal	Hexadecimal	Name	
1001	0x000003E9	ERROR_STACK_OVERFLOW	Recursion too deep; the stack overflowed.
1002	0x000003EA	ERROR_INVALID_MESSAGE	The window cannot act on the sent message.
1003	0x000003EB	ERROR_CAN_NOT_COMPLETE	Cannot complete this function.
1004	0x000003EC	ERROR_INVALID_FLAGS	Invalid flags.
1005	0x000003ED	ERROR_UNRECOGNIZED_VOLUME	The volume does not contain a recognized file system. Please make sure that all required file system drivers are loaded and that the volume is not corrupted.
1006	0x000003EE	ERROR_FILE_INVALID	The volume for a file has been externally altered so that the opened file is no longer valid.
1007	0x000003EF	ERROR_FULLSCREEN_MODE	The requested operation cannot be performed in full-screen mode.
1008	0x000003F0	ERROR_NO_TOKEN	An attempt was made to reference a token that does not exist.
1009	0x000003F1	ERROR_BADDB	The configuration registry database is corrupt.
1010	0x000003F2	ERROR_BADKEY	The configuration registry key is invalid.
1011	0x000003F3	ERROR_CANTOPEN	The configuration registry key could not be opened.
1012	0x000003F4	ERROR_CANTREAD	The configuration registry key could not be read.
1013	0x000003F5	ERROR_CANTWRITE	The configuration registry key could not be written.
1014	0x000003F6	ERROR_REGISTRY_RECOVERED	One of the files in the registry database had to be recovered by use of a log or alternate copy. The recovery was successful.
1015	0x000003F7	ERROR_REGISTRY_CORRUPT	The registry is corrupted. The structure of one of the files containing registry data is corrupted, or the system's memory image of the file is corrupted, or the file could not be recovered because the alternate copy or log was absent or corrupted.
1016	0x000003F8	ERROR_REGISTRY_IO_FAILED	An I/O operation initiated by the registry failed unrecoverably. The registry could not read in, or write out, or flush, one of the files that contain the system's image of the registry.
1017	0x000003F9	ERROR_NOT_REGISTRY_FILE	The system has attempted to load or restore a file into the registry, but the specified file is not in a registry file format.
1018	0x000003FA	ERROR_KEY_DELETED	Illegal operation attempted on a registry key that has been marked for deletion.
1019	0x000003FB	ERROR_NO_LOG_SPACE	System could not allocate the required space in a registry log.
1020	0x000003FC	ERROR_KEY_HAS_CHILDREN	Cannot create a symbolic link in a registry key that already has subkeys or values.
1021	0x000003FD	ERROR_CHILD_MUST_BE_VOLATILE	Cannot create a stable subkey under a volatile parent key.
1022	0x000003FE	ERROR_NOTIFY_ENUM_DIR	A notify change request is being completed and the information is not being returned in the caller's buffer. The caller now needs to enumerate the files to find the changes.
1051	0x0000041B	ERROR_DEPENDENT_SERVICES_RUNNING	A stop control has been sent to a service that other running services are dependent on.
1052	0x0000041C	ERROR_INVALID_SERVICE_CONTROL	The requested control is not valid for this service.
1053	0x0000041D	ERROR_SERVICE_REQUEST_TIMEOUT	The service did not respond to the start or control request in a timely fashion.
1054	0x0000041E	ERROR_SERVICE_NO_THREAD	A thread could not be created for the service.
1055	0x0000041F	ERROR_SERVICE_DATABASE_LOCKED	The service database is locked.
1056	0x00000420	ERROR_SERVICE_ALREADY_RUNNING	An instance of the service is already running.
1057	0x00000421	ERROR_INVALID_SERVICE_ACCOUNT	The account name is invalid or does not exist, or the password is invalid for the account name specified.
1058	0x00000422	ERROR_SERVICE_DISABLED	The service cannot be started, either because it is disabled or because it has no enabled devices associated with it.
1059	0x00000423	ERROR_CIRCULAR_DEPENDENCY	Circular service dependency was specified.

Error			Description
decimal	Hexadecimal	Name	
1060	0x00000424	ERROR_SERVICE_DOES_NOT_EXIST	The specified service does not exist as an installed service.
1061	0x00000425	ERROR_SERVICE_CANNOT_ACCEPT_CTRL	The service cannot accept control messages at this time.
1062	0x00000426	ERROR_SERVICE_NOT_ACTIVE	The service has not been started.
1063	0x00000427	ERROR_FAILED_SERVICE_CONTROLLER_CONNECT	The service process could not connect to the service controller.
1064	0x00000428	ERROR_EXCEPTION_IN_SERVICE	An exception occurred in the service when handling the control request.
1065	0x00000429	ERROR_DATABASE_DOES_NOT_EXIST	The database specified does not exist.
1066	0x0000042A	ERROR_SERVICE_SPECIFIC_ERROR	The service has returned a service-specific error code.
1067	0x0000042B	ERROR_PROCESS_ABORTED	The process terminated unexpectedly.
1068	0x0000042C	ERROR_SERVICE_DEPENDENCY_FAIL	The dependency service or group failed to start.
1069	0x0000042D	ERROR_SERVICE_LOGON_FAILED	The service did not start due to a logon failure.
1070	0x0000042E	ERROR_SERVICE_START_HANG	After starting, the service hung in a start-pending state.
1071	0x0000042F	ERROR_INVALID_SERVICE_LOCK	The specified service database lock is invalid.
1072	0x00000430	ERROR_SERVICE_MARKED_FOR_DELETE	The specified service has been marked for deletion.
1073	0x00000431	ERROR_SERVICE_EXISTS	The specified service already exists.
1074	0x00000432	ERROR_ALREADY_RUNNING_LKG	The system is currently running with the last-known-good configuration.
1075	0x00000433	ERROR_SERVICE_DEPENDENCY_DELETED	The dependency service does not exist or has been marked for deletion.
1076	0x00000434	ERROR_BOOT_ALREADY_ACCEPTED	The current boot has already been accepted for use as the last-known-good control set.
1077	0x00000435	ERROR_SERVICE_NEVER_STARTED	No attempts to start the service have been made since the last boot.
1078	0x00000436	ERROR_DUPLICATE_SERVICE_NAME	The name is already in use as either a service name or a service display name.
1079	0x00000437	ERROR_DIFFERENT_SERVICE_ACCOUNT	The account specified for this service is different from the account specified for other services running in the same process.
1080	0x00000438	ERROR_CANNOT_DETECT_DRIVER_FAILURE	Failure actions can only be set for Win32 services, not for drivers.
1081	0x00000439	ERROR_CANNOT_DETECT_PROCESS_ABORT	This service runs in the same process as the service control manager. Therefore, the service control manager cannot take action if this service's process terminates unexpectedly.
1082	0x0000043A	ERROR_NO_RECOVERY_PROGRAM	No recovery program has been configured for this service.
1083	0x0000043B	ERROR_SERVICE_NOT_IN_EXE	The executable program that this service is configured to run in does not implement the service.
1084	0x0000043C	ERROR_NOT_SAFEBOOT_SERVICE	This service cannot be started in Safe Mode.

Error			Description
decimal	Hexadecimal	Name	
1100	0x0000044C	ERROR_END_OF_MEDIA	The physical end of the tape has been reached.
1101	0x0000044D	ERROR_FILEMARK_DETECTED	A tape access reached a filemark.
1102	0x0000044E	ERROR_BEGINNING_OF_MEDIA	The beginning of the tape or a partition was encountered.
1103	0x0000044F	ERROR_SETMARK_DETECTED	A tape access reached the end of a set of files.
1104	0x00000450	ERROR_NO_DATA_DETECTED	No more data is on the tape.
1105	0x00000451	ERROR_PARTITION_FAILURE	Tape could not be partitioned.
1106	0x00000452	ERROR_INVALID_BLOCK_LENGTH	When accessing a new tape of a multivolume partition, the current block size is incorrect.
1107	0x00000453	ERROR_DEVICE_NOT_PARTITIONED	Tape partition information could not be found when loading a tape.
1108	0x00000454	ERROR_UNABLE_TO_LOCK_MEDIA	Unable to lock the media eject mechanism.
1109	0x00000455	ERROR_UNABLE_TO_UNLOAD_MEDIA	Unable to unload the media.
1110	0x00000456	ERROR_MEDIA_CHANGED	The media in the drive may have changed.
1111	0x00000457	ERROR_BUS_RESET	The I/O bus was reset.
1112	0x00000458	ERROR_NO_MEDIA_IN_DRIVE	No media in drive.
1113	0x00000459	ERROR_NO_UNICODE_TRANSLATION	No mapping for the Unicode character exists in the target multi-byte code page.
1114	0x0000045A	ERROR_DLL_INIT_FAILED	A dynamic link library (DLL) initialization routine failed.
1115	0x0000045B	ERROR_SHUTDOWN_IN_PROGRESS	A system shutdown is in progress.
1116	0x0000045C	ERROR_NO_SHUTDOWN_IN_PROGRESS	Unable to abort the system shutdown because no shutdown was in progress.
1117	0x0000045D	ERROR_IO_DEVICE	The request could not be performed because of an I/O device error.
1118	0x0000045E	ERROR_SERIAL_NO_DEVICE	No serial device was successfully initialized. The serial driver will unload.
1119	0x0000045F	ERROR_IRQ_BUSY	Unable to open a device that was sharing an interrupt request (IRQ) with other devices. At least one other device that uses that IRQ was already opened.
1120	0x00000460	ERROR_MORE_WRITES	A serial I/O operation was completed by another write to the serial port. (The IOCTL_SERIAL_XOFF_COUNTER reached zero.)
1121	0x00000461	ERROR_COUNTER_TIMEOUT	A serial I/O operation completed because the timeout period expired. (The IOCTL_SERIAL_XOFF_COUNTER did not reach zero.)
1122	0x00000462	ERROR_FLOPPY_ID_MARK_NOT_FOUND	No ID address mark was found on the floppy disk.
1123	0x00000463	ERROR_FLOPPY_WRONG_CYLINDER	Mismatch between the floppy disk sector ID field and the floppy disk controller track address.
1124	0x00000464	ERROR_FLOPPY_UNKNOWN_ERROR	The floppy disk controller reported an error that is not recognized by the floppy disk driver.
1125	0x00000465	ERROR_FLOPPY_BAD_REGISTERS	The floppy disk controller returned inconsistent results in its registers.
1126	0x00000466	ERROR_DISK_RECALIBRATE_FAILED	While accessing the hard disk, a recalibrate operation failed, even after retries.
1127	0x00000467	ERROR_DISK_OPERATION_FAILED	While accessing the hard disk, a disk operation failed even after retries.
1128	0x00000468	ERROR_DISK_RESET_FAILED	While accessing the hard disk, a disk controller reset was needed, but even that failed.
1129	0x00000469	ERROR_EOM_OVERFLOW	Physical end of tape encountered.
1130	0x0000046A	ERROR_NOT_ENOUGH_SERVER_MEMORY	Not enough server storage is available to process this command.
1131	0x0000046B	ERROR_POSSIBLE_DEADLOCK	A potential deadlock condition has been detected.
1132	0x0000046C	ERROR_MAPPED_ALIGNMENT	The base address or the file offset specified does not have the proper alignment.
1140	0x00000474	ERROR_SET_POWER_STATE_VETOED	An attempt to change the system power state was vetoed by another application or driver.
1141	0x00000475	ERROR_SET_POWER_STATE_FAILED	The system BIOS failed an attempt to change the system power state.
1142	0x00000476	ERROR_TOO_MANY_LINKS	An attempt was made to create more links on a file than the file system supports.

Error			Description
decimal	Hexadecimal	Name	
1150	0x0000047E	ERROR_OLD_WIN_VERSION	The specified program requires a newer version of Windows.
1151	0x0000047F	ERROR_APP_WRONG_OS	The specified program is not a Windows or MS-DOS program.
1152	0x00000480	ERROR_SINGLE_INSTANCE_APP	Cannot start more than one instance of the specified program.
1153	0x00000481	ERROR_RMODE_APP	The specified program was written for an earlier version of Windows.
1154	0x00000482	ERROR_INVALID_DLL	One of the library files needed to run this application is damaged.
1155	0x00000483	ERROR_NO_ASSOCIATION	No application is associated with the specified file for this operation.
1156	0x00000484	ERROR_DDE_FAIL	An error occurred in sending the command to the application.
1157	0x00000485	ERROR_DLL_NOT_FOUND	One of the library files needed to run this application cannot be found.
1158	0x00000486	ERROR_NO_MORE_USER_HANDLES	The current process has used all of its system allowance of handles for Window Manager objects.
1159	0x00000487	ERROR_MESSAGE_SYNC_ONLY	The message can be used only with synchronous operations.
1160	0x00000488	ERROR_SOURCE_ELEMENT_EMPTY	The indicated source element has no media.
1161	0x00000489	ERROR_DESTINATION_ELEMENT_FULL	The indicated destination element already contains media.
1162	0x0000048A	ERROR_ILLEGAL_ELEMENT_ADDRESS	The indicated element does not exist.
1163	0x0000048B	ERROR_MAGAZINE_NOT_PRESENT	The indicated element is part of a magazine that is not present.
1164	0x0000048C	ERROR_DEVICE_REINITIALIZATION_NEEDED	The indicated device requires reinitialization due to hardware errors.
1165	0x0000048D	ERROR_DEVICE_REQUIRES_CLEANING	The device has indicated that cleaning is required before further operations are attempted.
1166	0x0000048E	ERROR_DEVICE_DOOR_OPEN	The device has indicated that its door is open.
1167	0x0000048F	ERROR_DEVICE_NOT_CONNECTED	The device is not connected.
1168	0x00000490	ERROR_NOT_FOUND	Element not found.
1169	0x00000491	ERROR_NO_MATCH	There was no match for the specified key in the index.
1170	0x00000492	ERROR_SET_NOT_FOUND	The property set specified does not exist on the object.
1171	0x00000493	ERROR_POINT_NOT_FOUND	The point passed to GetMouseMovePointsEx is not in the buffer.
1172	0x00000494	ERROR_NO_TRACKING_SERVICE	The tracking (workstation) service is not running.
1173	0x00000495	ERROR_NO_VOLUME_ID	The Volume ID could not be found.
1175	0x00000497	ERROR_UNABLE_TO_REMOVE_REPLACED	Unable to remove the file to be replaced.
1176	0x00000498	ERROR_UNABLE_TO_MOVE_REPLACEMENT	Unable to move the replacement file to the file to be replaced. The file to be replaced has retained its original name.
1177	0x00000499	ERROR_UNABLE_TO_MOVE_REPLACEMENT_2	Unable to move the replacement file to the file to be replaced. The file to be replaced has been renamed using the backup name.
1178	0x0000049A	ERROR_JOURNAL_DELETE_IN_PROGRESS	The volume change journal is being deleted.
1179	0x0000049B	ERROR_JOURNAL_NOT_ACTIVE	The volume change journal is not active.
1180	0x0000049C	ERROR_POTENTIAL_FILE_FOUND	A file was found, but it may not be the correct file.
1181	0x0000049D	ERROR_JOURNAL_ENTRY_DELETED	The journal entry has been deleted from the journal.

Error			Description
decimal	Hexadecimal	Name	
1200	0x000004B0	ERROR_BAD_DEVICE	The specified device name is invalid.
1201	0x000004B1	ERROR_CONNECTION_UNAVAIL	The device is not currently connected but it is a remembered connection.
1202	0x000004B2	ERROR_DEVICE_ALREADY_REMEMBERED	The local device name has a remembered connection to another network resource.
1203	0x000004B3	ERROR_NO_NET_OR_BAD_PATH	No network provider accepted the given network path.
1204	0x000004B4	ERROR_BAD_PROVIDER	The specified network provider name is invalid.
1205	0x000004B5	ERROR_CANNOT_OPEN_PROFILE	Unable to open the network connection profile.
1206	0x000004B6	ERROR_BAD_PROFILE	The network connection profile is corrupted.
1207	0x000004B7	ERROR_NOT_CONTAINER	Cannot enumerate a noncontainer.
1208	0x000004B8	ERROR_EXTENDED_ERROR	An extended error has occurred.
1209	0x000004B9	ERROR_INVALID_GROUPNAME	The format of the specified group name is invalid.
1210	0x000004BA	ERROR_INVALID_COMPUTERNAME	The format of the specified computer name is invalid.
1211	0x000004BB	ERROR_INVALID_EVENTNAME	The format of the specified event name is invalid.
1212	0x000004BC	ERROR_INVALID_DOMAINNAME	The format of the specified domain name is invalid.
1213	0x000004BD	ERROR_INVALID_SERVICENAME	The format of the specified service name is invalid.
1214	0x000004BE	ERROR_INVALID_NETNAME	The format of the specified network name is invalid.
1215	0x000004BF	ERROR_INVALID_SHARENAME	The format of the specified share name is invalid.
1216	0x000004C0	ERROR_INVALID_PASSWORDNAME	The format of the specified password is invalid.
1217	0x000004C1	ERROR_INVALID_MESSAGENAME	The format of the specified message name is invalid.
1218	0x000004C2	ERROR_INVALID_MESSAGEDEST	The format of the specified message destination is invalid.
1219	0x000004C3	ERROR_SESSION_CREDENTIAL_CONFLICT	The credentials supplied conflict with an existing set of credentials.
1220	0x000004C4	ERROR_REMOTE_SESSION_LIMIT_EXCEEDED	An attempt was made to establish a session to a network server, but there are already too many sessions established to that server.
1221	0x000004C5	ERROR_DUP_DOMAINNAME	The workgroup or domain name is already in use by another computer on the network.
1222	0x000004C6	ERROR_NO_NETWORK	The network is not present or not started.
1223	0x000004C7	ERROR_CANCELLED	The operation was canceled by the user.
1224	0x000004C8	ERROR_USER_MAPPED_FILE	The requested operation cannot be performed on a file with a user-mapped section open.
1225	0x000004C9	ERROR_CONNECTION_REFUSED	The remote system refused the network connection.
1226	0x000004CA	ERROR_GRACEFUL_DISCONNECT	The network connection was gracefully closed.
1227	0x000004CB	ERROR_ADDRESS_ALREADY_ASSOCIATED	The network transport endpoint already has an address associated with it.
1228	0x000004CC	ERROR_ADDRESS_NOT_ASSOCIATED	An address has not yet been associated with the network endpoint.
1229	0x000004CD	ERROR_CONNECTION_INVALID	An operation was attempted on a nonexistent network connection.
1230	0x000004CE	ERROR_CONNECTION_ACTIVE	An invalid operation was attempted on an active network connection.
1231	0x000004CF	ERROR_NETWORK_UNREACHABLE	The network location cannot be reached. For information about network troubleshooting, see Windows Help.
1232	0x000004D0	ERROR_HOST_UNREACHABLE	The network location cannot be reached. For information about network troubleshooting, see Windows Help.
1233	0x000004D1	ERROR_PROTOCOL_UNREACHABLE	The network location cannot be reached. For information about network troubleshooting, see Windows Help.
1234	0x000004D2	ERROR_PORT_UNREACHABLE	No service is operating at the destination network endpoint on the remote system.
1235	0x000004D3	ERROR_REQUEST_ABORTED	The request was aborted.
1236	0x000004D4	ERROR_CONNECTION_ABORTED	The network connection was aborted by the local system.

Error			Description
decimal	Hexadecimal	Name	
1237	0x000004D5	ERROR_RETRY	The operation could not be completed. A retry should be performed.
1238	0x000004D6	ERROR_CONNECTION_COUNT_LIMIT	A connection to the server could not be made because the limit on the number of concurrent connections for this account has been reached.
1239	0x000004D7	ERROR_LOGIN_TIME_RESTRICTION	Attempting to log in during an unauthorized time of day for this account.
1240	0x000004D8	ERROR_LOGIN_WKSTA_RESTRICTION	The account is not authorized to log in from this station.
1241	0x000004D9	ERROR_INCORRECT_ADDRESS	The network address could not be used for the operation requested.
1242	0x000004DA	ERROR_ALREADY_REGISTERED	The service is already registered.
1243	0x000004DB	ERROR_SERVICE_NOT_FOUND	The specified service does not exist.
1244	0x000004DC	ERROR_NOT_AUTHENTICATED	The operation being requested was not performed because the user has not been authenticated.
1245	0x000004DD	ERROR_NOT_LOGGED_ON	The operation being requested was not performed because the user has not logged on to the network. The specified service does not exist.
1246	0x000004DE	ERROR_CONTINUE	Continue with work in progress.
1247	0x000004DF	ERROR_ALREADY_INITIALIZED	An attempt was made to perform an initialization operation when initialization has already been completed.
1248	0x000004E0	ERROR_NO_MORE_DEVICES	No more local devices.
1249	0x000004E1	ERROR_NO_SUCH_SITE	The specified site does not exist.
1250	0x000004E2	ERROR_DOMAIN_CONTROLLER_EXISTS	A domain controller with the specified name already exists.
1251	0x000004E3	ERROR_ONLY_IF_CONNECTED	This operation is supported only when you are connected to the server.
1252	0x000004E4	ERROR_OVERRIDE_NOCHANGES	The group policy framework should call the extension even if there are no changes.
1253	0x000004E5	ERROR_BAD_USER_PROFILE	The specified user does not have a valid profile.
1254	0x000004E6	ERROR_NOT_SUPPORTED_ON_SBS	This operation is not supported on a Microsoft Small Business Server.
1255	0x000004E7	ERROR_SERVER_SHUTDOWN_IN_PROGRESS	The server machine is shutting down.
1256	0x000004E8	ERROR_HOST_DOWN	The remote system is not available. For information about network troubleshooting, see Windows Help.
1257	0x000004E9	ERROR_NON_ACCOUNT_SID	The security identifier provided is not from an account domain.
1258	0x000004EA	ERROR_NON_DOMAIN_SID	The security identifier provided does not have a domain component.
1259	0x000004EB	ERROR_APPHELP_BLOCK	AppHelp dialog canceled thus preventing the application from starting.
1260	0x000004EC	ERROR_ACCESS_DISABLED_BY_POLICY	Access to the requested resource has been disabled by your administrator.
1261	0x000004ED	ERROR_REG_NAT_CONSUMPTION	A program attempt to use an invalid register value. Normally caused by an uninitialized register. This error is Itanium specific.
1262	0x000004EE	ERROR_CSCSHARE_OFFLINE	The share is currently offline or does not exist.
1300	0x00000514	ERROR_NOT_ALL_ASSIGNED	Not all privileges referenced are assigned to the caller.
1301	0x00000515	ERROR_SOME_NOT_MAPPED	Some mapping between account names and security IDs was not done.
1302	0x00000516	ERROR_NO_QUOTAS_FOR_ACCOUNT	No system quota limits are specifically set for this account.
1303	0x00000517	ERROR_LOCAL_USER_SESSION_KEY	No encryption key is available. A well-known encryption key was returned.
1304	0x00000518	ERROR_NULL_LM_PASSWORD	The password is too complex to be converted to a LAN Manager password. The LAN Manager password returned is a NULL string.
1305	0x00000519	ERROR_UNKNOWN_REVISION	The revision level is unknown.
1306	0x0000051A	ERROR_REVISION_MISMATCH	Indicates two revision levels are incompatible.
1307	0x0000051B	ERROR_INVALID_OWNER	This security ID may not be assigned as the owner of this object.

Error			Description
decimal	Hexadecimal	Name	
1308	0x0000051C	ERROR_INVALID_PRIMARY_GROUP	This security ID may not be assigned as the primary group of an object.
1309	0x0000051D	ERROR_NO_IMPERSONATION_TOKEN	An attempt has been made to operate on an impersonation token by a thread that is not currently impersonating a client.
1310	0x0000051E	ERROR_CANT_DISABLE_MANDATORY	The group may not be disabled.
1311	0x0000051F	ERROR_NO_LOGON_SERVERS	There are currently no logon servers available to service the logon request.
1312	0x00000520	ERROR_NO_SUCH_LOGON_SESSION	A specified logon session does not exist. It may already have been terminated.
1313	0x00000521	ERROR_NO_SUCH_PRIVILEGE	A specified privilege does not exist.
1314	0x00000522	ERROR_PRIVILEGE_NOT_HELD	A required privilege is not held by the client.
1315	0x00000523	ERROR_INVALID_ACCOUNT_NAME	The name provided is not a properly formed account name.
1316	0x00000524	ERROR_USER_EXISTS	The specified user already exists.
1317	0x00000525	ERROR_NO_SUCH_USER	The specified user does not exist.
1318	0x00000526	ERROR_GROUP_EXISTS	The specified group already exists.
1319	0x00000527	ERROR_NO_SUCH_GROUP	The specified group does not exist.
1320	0x00000528	ERROR_MEMBER_IN_GROUP	Either the specified user account is already a member of the specified group, or the specified group cannot be deleted because it contains a member.
1321	0x00000529	ERROR_MEMBER_NOT_IN_GROUP	The specified user account is not a member of the specified group account.
1322	0x0000052A	ERROR_LAST_ADMIN	The last remaining administration account cannot be disabled or deleted.
1323	0x0000052B	ERROR_WRONG_PASSWORD	Unable to update the password. The value provided as the current password is incorrect.
1324	0x0000052C	ERROR_ILL_FORMED_PASSWORD	Unable to update the password. The value provided for the new password contains values that are not allowed in passwords.
1325	0x0000052D	ERROR_PASSWORD_RESTRICTION	Unable to update the password. The value provided for the new password does not meet the length, complexity, or history requirement of the domain.
1326	0x0000052E	ERROR_LOGON_FAILURE	Logon failure: unknown user name or bad password.
1327	0x0000052F	ERROR_ACCOUNT_RESTRICTION	Logon failure: user account restriction.
1328	0x00000530	ERROR_INVALID_LOGON_HOURS	Logon failure: account logon time restriction violation.
1329	0x00000531	ERROR_INVALID_WORKSTATION	Logon failure: user not allowed to log on to this computer.
1330	0x00000532	ERROR_PASSWORD_EXPIRED	Logon failure: the specified account password has expired.
1331	0x00000533	ERROR_ACCOUNT_DISABLED	Logon failure: account currently disabled.
1332	0x00000534	ERROR_NONE_MAPPED	No mapping between account names and security IDs was done.
1333	0x00000535	ERROR_TOO_MANY_LUIDS_REQUESTED	Too many local user identifiers (LUIDs) were requested at one time.
1334	0x00000536	ERROR_LUIDS_EXHAUSTED	No more local user identifiers (LUIDs) are available.
1335	0x00000537	ERROR_INVALID_SUB_AUTHORITY	The subauthority part of a security ID is invalid for this particular use.
1336	0x00000538	ERROR_INVALID_ACL	The access control list (ACL) structure is invalid.
1337	0x00000539	ERROR_INVALID_SID	The security ID structure is invalid.
1338	0x0000053A	ERROR_INVALID_SECURITY_DESCR	The security descriptor structure is invalid.
1340	0x0000053C	ERROR_BAD_INHERITANCE_ACL	The inherited access control list (ACL) or access control entry (ACE) could not be built.
1341	0x0000053D	ERROR_SERVER_DISABLED	The server is currently disabled.
1342	0x0000053E	ERROR_SERVER_NOT_DISABLED	The server is currently enabled.
1343	0x0000053F	ERROR_INVALID_ID_AUTHORITY	The value provided was an invalid value for an identifier authority.
1344	0x00000540	ERROR_ALLOTTED_SPACE_EXCEEDED	No more memory is available for security information updates.

Error			Description
decimal	Hexadecimal	Name	
1345	0x00000541	ERROR_INVALID_GROUP_ATTRIBUTES	The specified attributes are invalid, or incompatible with the attributes for the group as a whole.
1346	0x00000542	ERROR_BAD_IMPERSONATION_LEVEL	Either a required impersonation level was not provided, or the provided impersonation level is invalid.
1347	0x00000543	ERROR_CANT_OPEN_ANONYMOUS	Cannot open an anonymous level security token.
1348	0x00000544	ERROR_BAD_VALIDATION_CLASS	The validation information class requested was invalid.
1349	0x00000545	ERROR_BAD_TOKEN_TYPE	The type of the token is inappropriate for its attempted use.
1350	0x00000546	ERROR_NO_SECURITY_ON_OBJECT	Unable to perform a security operation on an object that has no associated security.
1351	0x00000547	ERROR_CANT_ACCESS_DOMAIN_INFO	Configuration information could not be read from the domain controller, either because the machine is unavailable, or access has been denied.
1352	0x00000548	ERROR_INVALID_SERVER_STATE	The security account manager (SAM) or local security authority (LSA) server was in the wrong state to perform the security operation.
1353	0x00000549	ERROR_INVALID_DOMAIN_STATE	The domain was in the wrong state to perform the security operation.
1354	0x0000054A	ERROR_INVALID_DOMAIN_ROLE	This operation is only allowed for the Primary Domain Controller of the domain.
1355	0x0000054B	ERROR_NO_SUCH_DOMAIN	The specified domain either does not exist or could not be contacted.
1356	0x0000054C	ERROR_DOMAIN_EXISTS	The specified domain already exists.
1357	0x0000054D	ERROR_DOMAIN_LIMIT_EXCEEDED	An attempt was made to exceed the limit on the number of domains per server.
1358	0x0000054E	ERROR_INTERNAL_DB_CORRUPTION	Unable to complete the requested operation because of either a catastrophic media failure or a data structure corruption on the disk.
1359	0x0000054F	ERROR_INTERNAL_ERROR	An internal error occurred.
1360	0x00000550	ERROR_GENERIC_NOT_MAPPED	Generic access types were contained in an access mask which should already be mapped to nongeneric types.
1361	0x00000551	ERROR_BAD_DESCRIPTOR_FORMAT	A security descriptor is not in the right format (absolute or self-relative).
1362	0x00000552	ERROR_NOT_LOGON_PROCESS	The requested action is restricted for use by logon processes only. The calling process has not registered as a logon process.
1363	0x00000553	ERROR_LOGON_SESSION_EXISTS	Cannot start a new logon session with an ID that is already in use.
1364	0x00000554	ERROR_NO_SUCH_PACKAGE	A specified authentication package is unknown.
1365	0x00000555	ERROR_BAD_LOGON_SESSION_STATE	The logon session is not in a state that is consistent with the requested operation.
1366	0x00000556	ERROR_LOGON_SESSION_COLLISION	The logon session ID is already in use.
1367	0x00000557	ERROR_INVALID_LOGON_TYPE	A logon request contained an invalid logon type value.
1368	0x00000558	ERROR_CANNOT_IMPERSONATE	Unable to impersonate using a named pipe until data has been read from that pipe.
1369	0x00000559	ERROR_RXACT_INVALID_STATE	The transaction state of a registry subtree is incompatible with the requested operation.
1370	0x0000055A	ERROR_RXACT_COMMIT_FAILURE	An internal security database corruption has been encountered.
1371	0x0000055B	ERROR_SPECIAL_ACCOUNT	Cannot perform this operation on built-in accounts.
1372	0x0000055C	ERROR_SPECIAL_GROUP	Cannot perform this operation on this built-in special group.
1373	0x0000055D	ERROR_SPECIAL_USER	Cannot perform this operation on this built-in special user.
1374	0x0000055E	ERROR_MEMBERS_PRIMARY_GROUP	The user cannot be removed from a group because the group is currently the user's primary group.
1375	0x0000055F	ERROR_TOKEN_ALREADY_IN_USE	The token is already in use as a primary token.
1376	0x00000560	ERROR_NO_SUCH_ALIAS	The specified local group does not exist.

Error			Description
decimal	Hexadecimal	Name	
1377	0x00000561	ERROR_MEMBER_NOT_IN_ALIAS	The specified account name is not a member of the local group.
1378	0x00000562	ERROR_MEMBER_IN_ALIAS	The specified account name is already a member of the local group.
1379	0x00000563	ERROR_ALIAS_EXISTS	The specified local group already exists.
1380	0x00000564	ERROR_LOGON_NOT_GRANTED	Logon failure: the user has not been granted the requested logon type at this computer.
1381	0x00000565	ERROR_TOO_MANY_SECRETS	The maximum number of secrets that may be stored in a single system has been exceeded.
1382	0x00000566	ERROR_SECRET_TOO_LONG	The length of a secret exceeds the maximum length allowed.
1383	0x00000567	ERROR_INTERNAL_DB_ERROR	The local security authority database contains an internal inconsistency.
1384	0x00000568	ERROR_TOO_MANY_CONTEXT_IDS	During a logon attempt, the user's security context accumulated too many security IDs.
1385	0x00000569	ERROR_LOGON_TYPE_NOT_GRANTED	Logon failure: the user has not been granted the requested logon type at this computer.
1386	0x0000056A	ERROR_NT_CROSS_ENCRYPTION_REQUIRED	A cross-encrypted password is necessary to change a user password.
1387	0x0000056B	ERROR_NO_SUCH_MEMBER	A new member could not be added to or removed from the local group because the member does not exist.
1388	0x0000056C	ERROR_INVALID_MEMBER	A new member could not be added to a local group because the member has the wrong account type.
1389	0x0000056D	ERROR_TOO_MANY_SIDS	Too many security IDs have been specified.
1390	0x0000056E	ERROR_LM_CROSS_ENCRYPTION_REQUIRED	A cross-encrypted password is necessary to change this user password.
1391	0x0000056F	ERROR_NO_INHERITANCE	Indicates an ACL contains no inheritable components.
1392	0x00000570	ERROR_FILE_CORRUPT	The file or directory is corrupted and unreadable.
1393	0x00000571	ERROR_DISK_CORRUPT	The disk structure is corrupted and unreadable.
1394	0x00000572	ERROR_NO_USER_SESSION_KEY	There is no user session key for the specified logon session.
1395	0x00000573	ERROR_LICENSE_QUOTA_EXCEEDED	The service being accessed is licensed for a particular number of connections. No more connections can be made to the service at this time because there are already as many connections as the service can accept.
1396	0x00000574	ERROR_WRONG_TARGET_NAME	Logon Failure: The target account name is incorrect.
1397	0x00000575	ERROR_MUTUAL_AUTH_FAILED	Mutual Authentication failed. The server's password is out of date at the domain controller.
1398	0x00000576	ERROR_TIME_SKEW	There is a time difference between the client and server.
1399	0x00000577	ERROR_CURRENT_DOMAIN_NOT_ALLOWED	This operation can not be performed on the current domain.

Error			Description
decimal	Hexadecimal	Name	
1400	0x00000578	ERROR_INVALID_WINDOW_HANDLE	Invalid window handle.
1401	0x00000579	ERROR_INVALID_MENU_HANDLE	Invalid menu handle.
1402	0x0000057A	ERROR_INVALID_CURSOR_HANDLE	Invalid cursor handle.
1403	0x0000057B	ERROR_INVALID_ACCEL_HANDLE	Invalid accelerator table handle.
1404	0x0000057C	ERROR_INVALID_HOOK_HANDLE	Invalid hook handle.
1405	0x0000057D	ERROR_INVALID_DWP_HANDLE	Invalid handle to a multiple-window position structure.
1406	0x0000057E	ERROR_TLW_WITH_WSCHILD	Cannot create a top-level child window.
1407	0x0000057F	ERROR_CANNOT_FIND_WND_CLASS	Cannot find window class.
1408	0x00000580	ERROR_WINDOW_OF_OTHER_THREAD	Invalid window; it belongs to other thread.
1409	0x00000581	ERROR_HOTKEY_ALREADY_REGISTERED	Hot key is already registered.
1410	0x00000582	ERROR_CLASS_ALREADY_EXISTS	Class already exists.
1411	0x00000583	ERROR_CLASS_DOES_NOT_EXIST	Class does not exist.
1412	0x00000584	ERROR_CLASS_HAS_WINDOWS	Class still has open windows.
1413	0x00000585	ERROR_INVALID_INDEX	Invalid index.
1414	0x00000586	ERROR_INVALID_ICON_HANDLE	Invalid icon handle.
1415	0x00000587	ERROR_PRIVATE_DIALOG_INDEX	Using private DIALOG window words.
1416	0x00000588	ERROR_LISTBOX_ID_NOT_FOUND	The list box identifier was not found.
1417	0x00000589	ERROR_NO_WILDCARD_CHARACTERS	No wildcards were found.
1418	0x0000058A	ERROR_CLIPBOARD_NOT_OPEN	Thread does not have a clipboard open.
1419	0x0000058B	ERROR_HOTKEY_NOT_REGISTERED	Hot key is not registered.
1420	0x0000058C	ERROR_WINDOW_NOT_DIALOG	The window is not a valid dialog window.
1421	0x0000058D	ERROR_CONTROL_ID_NOT_FOUND	Control ID not found.
1422	0x0000058E	ERROR_INVALID_COMBOBOX_MESSAGE	Invalid message for a combo box because it does not have an edit control.
1423	0x0000058F	ERROR_WINDOW_NOT_COMBOBOX	The window is not a combo box.
1424	0x00000590	ERROR_INVALID_EDIT_HEIGHT	Height must be less than 256.
1425	0x00000591	ERROR_DC_NOT_FOUND	Invalid device context (DC) handle.
1426	0x00000592	ERROR_INVALID_HOOK_FILTER	Invalid hook procedure type.
1427	0x00000593	ERROR_INVALID_FILTER_PROC	Invalid hook procedure.
1428	0x00000594	ERROR_HOOK_NEEDS_HMOD	Cannot set nonlocal hook without a module handle.
1429	0x00000595	ERROR_GLOBAL_ONLY_HOOK	This hook procedure can only be set globally.
1430	0x00000596	ERROR_JOURNAL_HOOK_SET	The journal hook procedure is already installed.
1431	0x00000597	ERROR_HOOK_NOT_INSTALLED	The hook procedure is not installed.
1432	0x00000598	ERROR_INVALID_LB_MESSAGE	Invalid message for single-selection list box.
1433	0x00000599	ERROR_SETCOUNT_ON_BAD_LB	LB_SETCOUNT sent to non-lazy list box.
1434	0x0000059A	ERROR_LB_WITHOUT_TABSTOPS	This list box does not support tab stops.
1435	0x0000059B	ERROR_DESTROY_OBJECT_OF_OTHER_THREAD	Cannot destroy object created by another thread.
1436	0x0000059C	ERROR_CHILD_WINDOW_MENU	Child windows cannot have menus.
1437	0x0000059D	ERROR_NO_SYSTEM_MENU	The window does not have a system menu.
1438	0x0000059E	ERROR_INVALID_MSGBOX_STYLE	Invalid message box style.
1439	0x0000059F	ERROR_INVALID_SPI_VALUE	Invalid system-wide (SPI_*) parameter.
1440	0x000005A0	ERROR_SCREEN_ALREADY_LOCKED	Screen already locked.
1441	0x000005A1	ERROR_HWNDS_HAVE_DIFF_PARENT	All handles to windows in a multiple-window position structure must have the same parent.
1442	0x000005A2	ERROR_NOT_CHILD_WINDOW	The window is not a child window.
1443	0x000005A3	ERROR_INVALID_GW_COMMAND	Invalid GW_* command.
1444	0x000005A4	ERROR_INVALID_THREAD_ID	Invalid thread identifier.
1445	0x000005A5	ERROR_NON_MDICHILD_WINDOW	Cannot process a message from a window that is not a multiple document interface (MDI) window.
1446	0x000005A6	ERROR_POPUP_ALREADY_ACTIVE	Popup menu already active.
1447	0x000005A7	ERROR_NO_SCROLLBARS	The window does not have scroll bars.
1448	0x000005A8	ERROR_INVALID_SCROLLBAR_RANGE	Scroll bar range cannot be greater than MAXLONG.
1449	0x000005A9	ERROR_INVALID_SHOWWIN_COMMAND	Cannot show or remove the window in the way specified.

Error			Description
decimal	Hexadecimal	Name	
1450	0x000005AA	ERROR_NO_SYSTEM_RESOURCES	Insufficient system resources exist to complete the requested service.
1451	0x000005AB	ERROR_NONPAGED_SYSTEM_RESOURCES	Insufficient system resources exist to complete the requested service.
1452	0x000005AC	ERROR_PAGED_SYSTEM_RESOURCES	Insufficient system resources exist to complete the requested service.
1453	0x000005AD	ERROR_WORKING_SET_QUOTA	Insufficient quota to complete the requested service.
1454	0x000005AE	ERROR_PAGEFILE_QUOTA	Insufficient quota to complete the requested service.
1455	0x000005AF	ERROR_COMMITMENT_LIMIT	The paging file is too small for this operation to complete.
1456	0x000005B0	ERROR_MENU_ITEM_NOT_FOUND	A menu item was not found.
1457	0x000005B1	ERROR_INVALID_KEYBOARD_HANDLE	Invalid keyboard layout handle.
1458	0x000005B2	ERROR_HOOK_TYPE_NOT_ALLOWED	Hook type not allowed.
1459	0x000005B3	ERROR_REQUIRES_INTERACTIVE_WINDOWSTATION	This operation requires an interactive window station.
1460	0x000005B4	ERROR_TIMEOUT	This operation returned because the timeout period expired.
1461	0x000005B5	ERROR_INVALID_MONITOR_HANDLE	Invalid monitor handle.
1500	0x000005DC	ERROR_EVENTLOG_FILE_CORRUPT	The event log file is corrupted.
1501	0x000005DD	ERROR_EVENTLOG_CANT_START	No event log file could be opened, so the event logging service did not start.
1502	0x000005DE	ERROR_LOG_FILE_FULL	The event log file is full.
1503	0x000005DF	ERROR_EVENTLOG_FILE_CHANGED	The event log file has changed between read operations.

Error			Description
decimal	Hexadecimal	Name	
1601	0x00000641	ERROR_INSTALL_SERVICE_FAILURE	The Windows Installer service could not be accessed. Contact your support personnel to verify that the Windows Installer service is properly registered.
1602	0x00000642	ERROR_INSTALL_USEREXIT	User cancelled installation.
1603	0x00000643	ERROR_INSTALL_FAILURE	Fatal error during installation.
1604	0x00000644	ERROR_INSTALL_SUSPEND	Installation suspended, incomplete.
1605	0x00000645	ERROR_UNKNOWN_PRODUCT	This action is only valid for products that are currently installed.
1606	0x00000646	ERROR_UNKNOWN_FEATURE	Feature ID not registered.
1607	0x00000647	ERROR_UNKNOWN_COMPONENT	Component ID not registered.
1608	0x00000648	ERROR_UNKNOWN_PROPERTY	Unknown property.
1609	0x00000649	ERROR_INVALID_HANDLE_STATE	Handle is in an invalid state.
1610	0x0000064A	ERROR_BAD_CONFIGURATION	The configuration data for this product is corrupt. Contact your support personnel.
1611	0x0000064B	ERROR_INDEX_ABSENT	Component qualifier not present.
1612	0x0000064C	ERROR_INSTALL_SOURCE_ABSENT	The installation source for this product is not available. Verify that the source exists and that you can access it.
1613	0x0000064D	ERROR_INSTALL_PACKAGE_VERSION	This installation package cannot be installed by the Windows Installer service. You must install a Windows service pack that contains a newer version of the Windows Installer service.
1614	0x0000064E	ERROR_PRODUCT_UNINSTALLED	Product is uninstalled.
1615	0x0000064F	ERROR_BAD_QUERY_SYNTAX	SQL query syntax invalid or unsupported.
1616	0x00000650	ERROR_INVALID_FIELD	Record field does not exist.
1617	0x00000651	ERROR_DEVICE_REMOVED	The device has been removed.
1618	0x00000652	ERROR_INSTALL_ALREADY_RUNNING	Another installation is already in progress. Complete that installation before proceeding with this install.
1619	0x00000653	ERROR_INSTALL_PACKAGE_OPEN_FAILED	This installation package could not be opened. Verify that the package exists and that you can access it, or contact the application vendor to verify that this is a valid Windows Installer package.
1620	0x00000654	ERROR_INSTALL_PACKAGE_INVALID	This installation package could not be opened. Contact the application vendor to verify that this is a valid Windows Installer package.
1621	0x00000655	ERROR_INSTALL_UI_FAILURE	There was an error starting the Windows Installer service user interface. Contact your support personnel.
1622	0x00000656	ERROR_INSTALL_LOG_FAILURE	Error opening installation log file. Verify that the specified log file location exists and that you can write to it.
1623	0x00000657	ERROR_INSTALL_LANGUAGE_UNSUPPORTED	The language of this installation package is not supported by your system.
1624	0x00000658	ERROR_INSTALL_TRANSFORM_FAILURE	Error applying transforms. Verify that the specified transform paths are valid.
1625	0x00000659	ERROR_INSTALL_PACKAGE_REJECTED	This installation is forbidden by system policy. Contact your system administrator.
1626	0x0000065A	ERROR_FUNCTION_NOT_CALLED	Function could not be executed.
1627	0x0000065B	ERROR_FUNCTION_FAILED	Function failed during execution.
1628	0x0000065C	ERROR_INVALID_TABLE	Invalid or unknown table specified.
1629	0x0000065D	ERROR_DATATYPE_MISMATCH	Data supplied is of wrong type.
1630	0x0000065E	ERROR_UNSUPPORTED_TYPE	Data of this type is not supported.
1631	0x0000065F	ERROR_CREATE_FAILED	The Windows Installer service failed to start. Contact your support personnel.
1632	0x00000660	ERROR_INSTALL_TEMP_UNWRITABLE	The temp folder is either full or inaccessible. Verify that the temp folder exists and that you can write to it.
1633	0x00000661	ERROR_INSTALL_PLATFORM_UNSUPPORTED	This installation package is not supported by this processor type. Contact your product vendor.
1634	0x00000662	ERROR_INSTALL_NOTUSED	Component not used on this computer.

Error			Description
decimal	Hexadecimal	Name	
1635	0x00000663	ERROR_PATCH_PACKAGE_OPEN_FAILED	This patch package could not be opened. Verify that the patch package exists and that you can access it, or contact the application vendor to verify that this is a valid Windows Installer patch package.
1636	0x00000664	ERROR_PATCH_PACKAGE_INVALID	This patch package could not be opened. Contact the application vendor to verify that this is a valid Windows Installer patch package.
1637	0x00000665	ERROR_PATCH_PACKAGE_UNSUPPORTED.	This patch package cannot be processed by the Windows Installer service. You must install a Windows service pack that contains a newer version of the Windows Installer service.
1638	0x00000666	ERROR_PRODUCT_VERSION	Another version of this product is already installed. Installation of this version cannot continue. To configure or remove the existing version of this product, use Add/Remove Programs on the Control Panel.
1639	0x00000667	ERROR_INVALID_COMMAND_LINE	Invalid command line argument. Consult the Windows Installer SDK for detailed command line help.
1640	0x00000668	ERROR_INSTALL_REMOTE_DISALLOWED	Only administrators have permission to add, remove, or configure server software during a Terminal Services remote session. If you want to install or configure software on the server, contact your network administrator.
1641	0x00000669	ERROR_SUCCESS_REBOOT_INITIATED	The requested operation completed successfully. The system will be restarted so the changes can take effect.
1642	0x0000066A	ERROR_PATCH_TARGET_NOT_FOUND	The upgrade patch cannot be installed by the Windows Installer service because the program to be upgraded may be missing, or the upgrade patch may update a different version of the program. Verify that the program to be upgraded exists on your computer and that you have the correct upgrade patch.
1643	0x0000066B	ERROR_PATCH_PACKAGE_REJECTED	The patch package is not permitted by system policy. It is not signed with an appropriate certificate.
1644	0x0000066C	ERROR_INSTALL_TRANSFORM_REJECTED	One or more customizations are not permitted by system policy. They are not signed with an appropriate certificate.
1700	0x000006A4	RPC_S_INVALID_STRING_BINDING	The string binding is invalid.
1701	0x000006A5	RPC_S_WRONG_KIND_OF_BINDING	The binding handle is not the correct type.
1702	0x000006A6	RPC_S_INVALID_BINDING	The binding handle is invalid.
1703	0x000006A7	RPC_S_PROTSEQ_NOT_SUPPORTED	The RPC protocol sequence is not supported.
1704	0x000006A8	RPC_S_INVALID_RPC_PROTSEQ	The RPC protocol sequence is invalid.
1705	0x000006A9	RPC_S_INVALID_STRING_UUID	The string universal unique identifier (UUID) is invalid.
1706	0x000006AA	RPC_S_INVALID_ENDPOINT_FORMAT	The endpoint format is invalid.
1707	0x000006AB	RPC_S_INVALID_NET_ADDR	The network address is invalid.
1708	0x000006AC	RPC_S_NO_ENDPOINT_FOUND	No endpoint was found.
1709	0x000006AD	RPC_S_INVALID_TIMEOUT	The timeout value is invalid.
1710	0x000006AE	RPC_S_OBJECT_NOT_FOUND	The object universal unique identifier (UUID) was not found.
1711	0x000006AF	RPC_S_ALREADY_REGISTERED	The object universal unique identifier (UUID) has already been registered.
1712	0x000006B0	RPC_S_TYPE_ALREADY_REGISTERED	The type universal unique identifier (UUID) has already been registered.
1713	0x000006B1	RPC_S_ALREADY_LISTENING	The RPC server is already listening.
1714	0x000006B2	RPC_S_NO_PROTSEQS_REGISTERED	No protocol sequences have been registered.
1715	0x000006B3	RPC_S_NOT_LISTENING	The RPC server is not listening.
1716	0x000006B4	RPC_S_UNKNOWN_MGR_TYPE	The manager type is unknown.
1717	0x000006B5	RPC_S_UNKNOWN_IF	The interface is unknown.
1718	0x000006B6	RPC_S_NO_BINDINGS	There are no bindings.
1719	0x000006B7	RPC_S_NO_PROTSEQS	There are no protocol sequences.
1720	0x000006B8	RPC_S_CANT_CREATE_ENDPOINT	The endpoint cannot be created.

Error			Description
decimal	Hexadecimal	Name	
1721	0x000006B9	RPC_S_OUT_OF_RESOURCES	Not enough resources are available to complete this operation.
1722	0x000006BA	RPC_S_SERVER_UNAVAILABLE	The RPC server is unavailable.
1723	0x000006BB	RPC_S_SERVER_TOO_BUSY	The RPC server is too busy to complete this operation.
1724	0x000006BC	RPC_S_INVALID_NETWORK_OPTIONS	The network options are invalid.
1725	0x000006BD	RPC_S_NO_CALL_ACTIVE	There are no remote procedure calls active on this thread.
1726	0x000006BE	RPC_S_CALL_FAILED	The remote procedure call failed.
1727	0x000006BF	RPC_S_CALL_FAILED_DNE	The remote procedure call failed and did not execute.
1728	0x000006C0	RPC_S_PROTOCOL_ERROR	A remote procedure call (RPC) protocol error occurred.
1730	0x000006C2	RPC_S_UNSUPPORTED_TRANS_SYN	The transfer syntax is not supported by the RPC server.
1732	0x000006C4	RPC_S_UNSUPPORTED_TYPE	The universal unique identifier (UUID) type is not supported.
1733	0x000006C5	RPC_S_INVALID_TAG	The tag is invalid.
1734	0x000006C6	RPC_S_INVALID_BOUND	The array bounds are invalid.
1735	0x000006C7	RPC_S_NO_ENTRY_NAME	The binding does not contain an entry name.
1736	0x000006C8	RPC_S_INVALID_NAME_SYNTAX	The name syntax is invalid.
1737	0x000006C9	RPC_S_UNSUPPORTED_NAME_SYNTAX	The name syntax is not supported.
1739	0x000006CB	RPC_S_UUID_NO_ADDRESS	No network address is available to use to construct a universal unique identifier (UUID).
1740	0x000006CC	RPC_S_DUPLICATE_ENDPOINT	The endpoint is a duplicate.
1741	0x000006CD	RPC_S_UNKNOWN_AUTHN_TYPE	The authentication type is unknown.
1742	0x000006CE	RPC_S_MAX_CALLS_TOO_SMALL	The maximum number of calls is too small.
1743	0x000006CF	RPC_S_STRING_TOO_LONG	The string is too long.
1744	0x000006D0	RPC_S_PROTSEQ_NOT_FOUND	The RPC protocol sequence was not found.
1745	0x000006D1	RPC_S_PROCNUM_OUT_OF_RANGE	The procedure number is out of range.
1746	0x000006D2	RPC_S_BINDING_HAS_NO_AUTH	The binding does not contain any authentication information.
1747	0x000006D3	RPC_S_UNKNOWN_AUTHN_SERVICE	The authentication service is unknown.
1748	0x000006D4	RPC_S_UNKNOWN_AUTHN_LEVEL	The authentication level is unknown.
1749	0x000006D5	RPC_S_INVALID_AUTH_IDENTITY	The security context is invalid.
1750	0x000006D6	RPC_S_UNKNOWN_AUTHZ_SERVICE	The authorization service is unknown.
1751	0x000006D7	EPT_S_INVALID_ENTRY	The entry is invalid.
1752	0x000006D8	EPT_S_CANT_PERFORM_OP	The server endpoint cannot perform the operation.
1753	0x000006D9	EPT_S_NOT_REGISTERED	There are no more endpoints available from the endpoint mapper.
1754	0x000006DA	RPC_S_NOTHING_TO_EXPORT	No interfaces have been exported.
1755	0x000006DB	RPC_S_INCOMPLETE_NAME	The entry name is incomplete.
1756	0x000006DC	RPC_S_INVALID_VERS_OPTION	The version option is invalid.
1757	0x000006DD	RPC_S_NO_MORE_MEMBERS	There are no more members.
1758	0x000006DE	RPC_S_NOT_ALL_OBJS_UNEXPORTED	There is nothing to unexport.
1759	0x000006DF	RPC_S_INTERFACE_NOT_FOUND	The interface was not found.
1760	0x000006E0	RPC_S_ENTRY_ALREADY_EXISTS	The entry already exists.
1761	0x000006E1	RPC_S_ENTRY_NOT_FOUND	The entry is not found.
1762	0x000006E2	RPC_S_NAME_SERVICE_UNAVAILABLE	The name service is unavailable.
1763	0x000006E3	RPC_S_INVALID_NAF_ID	The network address family is invalid.
1764	0x000006E4	RPC_S_CANNOT_SUPPORT	The requested operation is not supported.
1765	0x000006E5	RPC_S_NO_CONTEXT_AVAILABLE	No security context is available to allow impersonation.
1766	0x000006E6	RPC_S_INTERNAL_ERROR	An internal error occurred in a remote procedure call (RPC).
1767	0x000006E7	RPC_S_ZERO_DIVIDE	The RPC server attempted an integer division by zero.
1768	0x000006E8	RPC_S_ADDRESS_ERROR	An addressing error occurred in the RPC server.
1769	0x000006E9	RPC_S_FP_DIV_ZERO	A floating-point operation at the RPC server caused a division by zero.

Error			Description
decimal	Hexadecimal	Name	
1770	0x000006EA	RPC_S_FP_UNDERFLOW	A floating-point underflow occurred at the RPC server.
1771	0x000006EB	RPC_S_FP_OVERFLOW	A floating-point overflow occurred at the RPC server.
1772	0x000006EC	RPC_X_NO_MORE_ENTRIES	The list of RPC servers available for the binding of auto handles has been exhausted.
1773	0x000006ED	RPC_X_SS_CHAR_TRANS_OPEN_FAIL	Unable to open the character translation table file.
1774	0x000006EE	RPC_X_SS_CHAR_TRANS_SHORT_FILE	The file containing the character translation table has fewer than 512 bytes.
1775	0x000006EF	RPC_X_SS_IN_NULL_CONTEXT	A null context handle was passed from the client to the host during a remote procedure call.
1777	0x000006F1	RPC_X_SS_CONTEXT_DAMAGED	The context handle changed during a remote procedure call.
1778	0x000006F2	RPC_X_SS_HANDLES_MISMATCH	The binding handles passed to a remote procedure call do not match.
1779	0x000006F3	RPC_X_SS_CANNOT_GET_CALL_HANDLE	The stub is unable to get the remote procedure call handle.
1780	0x000006F4	RPC_X_NULL_REF_POINTER	A null reference pointer was passed to the stub.
1781	0x000006F5	RPC_X_ENUM_VALUE_OUT_OF_RANGE	The enumeration value is out of range.
1782	0x000006F6	RPC_X_BYTE_COUNT_TOO_SMALL	The byte count is too small.
1783	0x000006F7	RPC_X_BAD_STUB_DATA	The stub received bad data.
1784	0x000006F8	ERROR_INVALID_USER_BUFFER	The supplied user buffer is not valid for the requested operation.
1785	0x000006F9	ERROR_UNRECOGNIZED_MEDIA	The disk media is not recognized. It may not be formatted.
1786	0x000006FA	ERROR_NO_TRUST_LSA_SECRET	The workstation does not have a trust secret.
1787	0x000006FB	ERROR_NO_TRUST_SAM_ACCOUNT	The security database on the server does not have a computer account for this workstation trust relationship.
1788	0x000006FC	ERROR_TRUSTED_DOMAIN_FAILURE	The trust relationship between the primary domain and the trusted domain failed.
1789	0x000006FD	ERROR_TRUSTED_RELATIONSHIP_FAILURE	The trust relationship between this workstation and the primary domain failed.
1790	0x000006FE	ERROR_TRUST_FAILURE	The network logon failed.
1791	0x000006FF	RPC_S_CALL_IN_PROGRESS	A remote procedure call is already in progress for this thread.
1792	0x00000700	ERROR_NETLOGON_NOT_STARTED	An attempt was made to logon, but the network logon service was not started.
1793	0x00000701	ERROR_ACCOUNT_EXPIRED	The user's account has expired.
1794	0x00000702	ERROR_REDIRECTOR_HAS_OPEN_HANDLES	The redirector is in use and cannot be unloaded.
1795	0x00000703	ERROR_PRINTER_DRIVER_ALREADY_INSTALLED	The specified printer driver is already installed.
1796	0x00000704	ERROR_UNKNOWN_PORT	The specified port is unknown.
1797	0x00000705	ERROR_UNKNOWN_PRINTER_DRIVER	The printer driver is unknown.
1798	0x00000706	ERROR_UNKNOWN_PRINTPROCESSOR	The print processor is unknown.
1799	0x00000707	ERROR_INVALID_SEPARATOR_FILE	The specified separator file is invalid.

Error			Description
decimal	Hexadecimal	Name	
1800	0x00000708	ERROR_INVALID_PRIORITY	The specified priority is invalid.
1801	0x00000709	ERROR_INVALID_PRINTER_NAME	The printer name is invalid.
1802	0x0000070A	ERROR_PRINTER_ALREADY_EXISTS	The printer already exists.
1803	0x0000070B	ERROR_INVALID_PRINTER_COMMAND	The printer command is invalid.
1804	0x0000070C	ERROR_INVALID_DATATYPE	The specified datatype is invalid.
1805	0x0000070D	ERROR_INVALID_ENVIRONMENT	The environment specified is invalid.
1806	0x0000070E	RPC_S_NO_MORE_BINDINGS	There are no more bindings.
1807	0x0000070F	ERROR_NOLOGON_INTERDOMAIN_TRUST_ACCOUNT	The account used is an interdomain trust account. Use your global user account or local user account to access this server.
1808	0x00000710	ERROR_NOLOGON_WORKSTATION_TRUST_ACCOUNT	The account used is a computer account. Use your global user account or local user account to access this server.
1809	0x00000711	ERROR_NOLOGON_SERVER_TRUST_ACCOUNT	The account used is a server trust account. Use your global user account or local user account to access this server.
1810	0x00000712	ERROR_DOMAIN_TRUST_INCONSISTENT	The name or security ID (SID) of the domain specified is inconsistent with the trust information for that domain.
1811	0x00000713	ERROR_SERVER_HAS_OPEN_HANDLES	The server is in use and cannot be unloaded.
1812	0x00000714	ERROR_RESOURCE_DATA_NOT_FOUND	The specified image file did not contain a resource section.
1813	0x00000715	ERROR_RESOURCE_TYPE_NOT_FOUND	The specified resource type cannot be found in the image file.
1814	0x00000716	ERROR_RESOURCE_NAME_NOT_FOUND	The specified resource name cannot be found in the image file.
1815	0x00000717	ERROR_RESOURCE_LANG_NOT_FOUND	The specified resource language ID cannot be found in the image file.
1816	0x00000718	ERROR_NOT_ENOUGH_QUOTA	Not enough quota is available to process this command.
1817	0x00000719	RPC_S_NO_INTERFACES	No interfaces have been registered.
1818	0x0000071A	RPC_S_CALL_CANCELLED	The remote procedure call was cancelled.
1819	0x0000071B	RPC_S_BINDING_INCOMPLETE	The binding handle does not contain all required information.
1820	0x0000071C	RPC_S_COMM_FAILURE	A communications failure occurred during a remote procedure call.
1821	0x0000071D	RPC_S_UNSUPPORTED_AUTHN_LEVEL	The requested authentication level is not supported.
1822	0x0000071E	RPC_S_NO_PRINC_NAME	No principal name registered.
1823	0x0000071F	RPC_S_NOT_RPC_ERROR	The error specified is not a valid Windows RPC error code.
1824	0x00000720	RPC_S_UUID_LOCAL_ONLY	A UUID that is valid only on this computer has been allocated.
1825	0x00000721	RPC_S_SEC_PKG_ERROR	A security package specific error occurred.
1826	0x00000722	RPC_S_NOT_CANCELLED	Thread is not canceled.
1827	0x00000723	RPC_X_INVALID_ES_ACTION	Invalid operation on the encoding/decoding handle.
1828	0x00000724	RPC_X_WRONG_ES_VERSION	Incompatible version of the serializing package.
1829	0x00000725	RPC_X_WRONG_STUB_VERSION	Incompatible version of the RPC stub.
1830	0x00000726	RPC_X_INVALID_PIPE_OBJECT	The RPC pipe object is invalid or corrupted.
1831	0x00000727	RPC_X_WRONG_PIPE_ORDER	An invalid operation was attempted on an RPC pipe object.
1832	0x00000728	RPC_X_WRONG_PIPE_VERSION	Unsupported RPC pipe version.
1898	0x0000076A	RPC_S_GROUP_MEMBER_NOT_FOUND	The group member was not found.
1899	0x0000076B	EPT_S_CANT_CREATE	The endpoint mapper database entry could not be created.
1900	0x0000076C	RPC_S_INVALID_OBJECT	The object universal unique identifier (UUID) is the nil UUID.
1901	0x0000076D	ERROR_INVALID_TIME	The specified time is invalid.
1902	0x0000076E	ERROR_INVALID_FORM_NAME	The specified form name is invalid.
1903	0x0000076F	ERROR_INVALID_FORM_SIZE	The specified form size is invalid.
1904	0x00000770	ERROR_ALREADY_WAITING	The specified printer handle is already being waited on

Error			Description
decimal	Hexadecimal	Name	
1905	0x00000771	ERROR_PRINTER_DELETED	The specified printer has been deleted.
1906	0x00000772	ERROR_INVALID_PRINTER_STATE	The state of the printer is invalid.
1907	0x00000773	ERROR_PASSWORD_MUST_CHANGE	The user's password must be changed before logging on the first time.
1908	0x00000774	ERROR_DOMAIN_CONTROLLER_NOT_FOUND	Could not find the domain controller for this domain.
1909	0x00000775	ERROR_ACCOUNT_LOCKED_OUT	The referenced account is currently locked out and may not be logged on to.
1910	0x00000776	OR_INVALID_OXID	The object exporter specified was not found.
1911	0x00000777	OR_INVALID_OID	The object specified was not found.
1912	0x00000778	OR_INVALID_SET	The object resolver set specified was not found.
1913	0x00000779	RPC_S_SEND_INCOMPLETE	Some data remains to be sent in the request buffer.
1914	0x0000077A	RPC_S_INVALID_ASYNC_HANDLE	Invalid asynchronous remote procedure call handle.
1915	0x0000077B	RPC_S_INVALID_ASYNC_CALL	Invalid asynchronous RPC call handle for this operation.
1916	0x0000077C	RPC_X_PIPE_CLOSED	The RPC pipe object has already been closed.
1917	0x0000077D	RPC_X_PIPE_DISCIPLINE_ERROR	The RPC call completed before all pipes were processed.
1918	0x0000077E	RPC_X_PIPE_EMPTY	No more data is available from the RPC pipe.
1919	0x0000077F	ERROR_NO_SITENAME	No site name is available for this machine.
1920	0x00000780	ERROR_CANT_ACCESS_FILE	The file can not be accessed by the system.
1921	0x00000781	ERROR_CANT_RESOLVE_FILENAME	The name of the file cannot be resolved by the system.
1922	0x00000782	RPC_S_ENTRY_TYPE_MISMATCH	The entry is not of the expected type.
1923	0x00000783	RPC_S_NOT_ALL_OBJS_EXPORTED	Not all object UUIDs could be exported to the specified entry.
1924	0x00000784	RPC_S_INTERFACE_NOT_EXPORTED	Interface could not be exported to the specified entry.
1925	0x00000785	RPC_S_PROFILE_NOT_ADDED	The specified profile entry could not be added.
1926	0x00000786	RPC_S_PRF_ELT_NOT_ADDED	The specified profile element could not be added.
1927	0x00000787	RPC_S_PRF_ELT_NOT_REMOVED	The specified profile element could not be removed.
1928	0x00000788	RPC_S_GRP_ELT_NOT_ADDED	The group element could not be added.
1929	0x00000789	RPC_S_GRP_ELT_NOT_REMOVED	The group element could not be removed.
1930	0x0000078A	ERROR_KM_DRIVER_BLOCKED	The printer driver is not compatible with a policy enabled on your computer that blocks NT 4.0 drivers.

Error			Description
decimal	Hexadecimal	Name	
2000	0x000007D0	ERROR_INVALID_PIXEL_FORMAT	The pixel format is invalid.
2001	0x000007D1	ERROR_BAD_DRIVER	The specified driver is invalid.
2002	0x000007D2	ERROR_INVALID_WINDOW_STYLE	The window style or class attribute is invalid for this operation.
2003	0x000007D3	ERROR_METAFILE_NOT_SUPPORTED	The requested metafile operation is not supported.
2004	0x000007D4	ERROR_TRANSFORM_NOT_SUPPORTED	The requested transformation operation is not supported.
2005	0x000007D5	ERROR_CLIPPING_NOT_SUPPORTED	The requested clipping operation is not supported.
2010	0x000007DA	ERROR_INVALID_CMM	The specified color management module is invalid.
2011	0x000007DB	ERROR_INVALID_PROFILE	The specified color profile is invalid.
2012	0x000007DC	ERROR_TAG_NOT_FOUND	The specified tag was not found.
2013	0x000007DD	ERROR_TAG_NOT_PRESENT	A required tag is not present.
2014	0x000007DE	ERROR_DUPLICATE_TAG	The specified tag is already present.
2015	0x000007DF	ERROR_PROFILE_NOT_ASSOCIATED_WITH_DEVICE	The specified color profile is not associated with any device.
2016	0x000007E0	ERROR_PROFILE_NOT_FOUND	The specified color profile was not found.
2017	0x000007E1	ERROR_INVALID_COLORSPACE	The specified color space is invalid.
2018	0x000007E2	ERROR_ICM_NOT_ENABLED	Image Color Management is not enabled.
2019	0x000007E3	ERROR_DELETING_ICM_XFORM	There was an error while deleting the color transform.
2020	0x000007E4	ERROR_INVALID_TRANSFORM	The specified color transform is invalid.
2021	0x000007E5	ERROR_COLORSPACE_MISMATCH	The specified transform does not match the bitmap's color space.
2022	0x000007E6	ERROR_INVALID_COLORINDEX	The specified named color index is not present in the profile.
2108	0x0000083C	ERROR_CONNECTED_OTHER_PASSWORD	The network connection was made successfully, but the user had to be prompted for a password other than the one originally specified.
2202	0x0000089A	ERROR_BAD_USERNAME	The specified username is invalid.
2250	0x000008CA	ERROR_NOT_CONNECTED	This network connection does not exist.
2401	0x00000961	ERROR_OPEN_FILES	This network connection has files open or requests pending.
2402	0x00000962	ERROR_ACTIVE_CONNECTIONS	Active connections still exist.
2404	0x00000964	ERROR_DEVICE_IN_USE	The device is in use by an active process and cannot be disconnected.
2500	0x000009C4	ERROR_PKINIT_FAILURE	The kerberos protocol encountered an error while validating the KDC certificate during smartcard logon.
2501	0x000009C5	ERROR_SMARTCARD_SUBSYSTEM_FAILURE	The kerberos protocol encountered an error while attempting to utilize the smartcard subsystem.

Error			Description
decimal	Hexadecimal	Name	
3000	0x00000BB8	ERROR_UNKNOWN_PRINT_MONITOR	The specified print monitor is unknown.
3001	0x00000BB9	ERROR_PRINTER_DRIVER_IN_USE	The specified printer driver is currently in use.
3002	0x00000BBA	ERROR_SPOOL_FILE_NOT_FOUND	The spool file was not found.
3003	0x00000BBB	ERROR_SPL_NO_STARTDOC	A StartDocPrinter call was not issued.
3004	0x00000BBC	ERROR_SPL_NO_ADDJOB	An AddJob call was not issued.
3005	0x00000BBD	ERROR_PRINT_PROCESSOR_ALREADY_INSTALLED	The specified print processor has already been installed.
3006	0x00000BBE	ERROR_PRINT_MONITOR_ALREADY_INSTALLED	The specified print monitor has already been installed.
3007	0x00000BBF	ERROR_INVALID_PRINT_MONITOR	The specified print monitor does not have the required functions.
3008	0x00000BC0	ERROR_PRINT_MONITOR_IN_USE	The specified print monitor is currently in use.
3009	0x00000BC1	ERROR_PRINTER_HAS_JOBS_QUEUED	The requested operation is not allowed when there are jobs queued to the printer.
3010	0x00000BC2	ERROR_SUCCESS_REBOOT_REQUIRED	The requested operation is successful. Changes will not be effective until the system is rebooted.
3011	0x00000BC3	ERROR_SUCCESS_RESTART_REQUIRED	The requested operation is successful. Changes will not be effective until the service is restarted.
3012	0x00000BC4	ERROR_PRINTER_NOT_FOUND	No printers were found.
4000	0x00000FA0	ERROR_WINS_INTERNAL	WINS encountered an error while processing the command.
4001	0x00000FA1	ERROR_CAN_NOT_DEL_LOCAL_WINS	The local WINS can not be deleted.
4002	0x00000FA2	ERROR_STATIC_INIT	The importation from the file failed.
4003	0x00000FA3	ERROR_INC_BACKUP	The backup failed. Was a full backup done before?
4004	0x00000FA4	ERROR_FULL_BACKUP	The backup failed. Check the directory to which you are backing the database.
4005	0x00000FA5	ERROR_REC_NON_EXISTENT	The name does not exist in the WINS database.
4006	0x00000FA6	ERROR_RPL_NOT_ALLOWED	Replication with a nonconfigured partner is not allowed.
4100	0x00001004	ERROR_DHCP_ADDRESS_CONFLICT	The DHCP client has obtained an IP address that is already in use on the network. The local interface will be disabled until the DHCP client can obtain a new address.
4200	0x00001068	ERROR_WMI_GUID_NOT_FOUND	The GUID passed was not recognized as valid by a WMI data provider.
4201	0x00001069	ERROR_WMI_INSTANCE_NOT_FOUND	The instance name passed was not recognized as valid by a WMI data provider.
4202	0x0000106A	ERROR_WMI_ITEMID_NOT_FOUND	The data item ID passed was not recognized as valid by a WMI data provider.
4203	0x0000106B	ERROR_WMI_TRY_AGAIN	The WMI request could not be completed and should be retried.
4204	0x0000106C	ERROR_WMI_DP_NOT_FOUND	The WMI data provider could not be located.
4205	0x0000106D	ERROR_WMI_UNRESOLVED_INSTANCE_REF	The WMI data provider references an instance set that has not been registered.
4206	0x0000106E	ERROR_WMI_ALREADY_ENABLED	The WMI data block or event notification has already been enabled.
4207	0x0000106F	ERROR_WMI_GUID_DISCONNECTED	The WMI data block is no longer available.
4208	0x00001070	ERROR_WMI_SERVER_UNAVAILABLE	The WMI data service is not available.
4209	0x00001071	ERROR_WMI_DP_FAILED	The WMI data provider failed to carry out the request.
4210	0x00001072	ERROR_WMI_INVALID_MOF	The WMI MOF information is not valid.
4211	0x00001073	ERROR_WMI_INVALID_REGINFO	The WMI registration information is not valid.
4212	0x00001074	ERROR_WMI_ALREADY_DISABLED	The WMI data block or event notification has already been disabled.
4213	0x00001075	ERROR_WMI_READ_ONLY	The WMI data item or data block is read only.
4214	0x00001076	ERROR_WMI_SET_FAILURE	The WMI data item or data block could not be changed.
4300	0x000010CC	ERROR_INVALID_MEDIA	The media identifier does not represent a valid medium.
4301	0x000010CD	ERROR_INVALID_LIBRARY	The library identifier does not represent a valid library.

Error			Description
decimal	Hexadecimal	Name	
4302	0x000010CE	ERROR_INVALID_MEDIA_POOL	The media pool identifier does not represent a valid media pool.
4303	0x000010CF	ERROR_DRIVE_MEDIA_MISMATCH	The drive and medium are not compatible or exist in different libraries.
4304	0x000010D0	ERROR_MEDIA_OFFLINE	The medium currently exists in an offline library and must be online to perform this operation.
4305	0x000010D1	ERROR_LIBRARY_OFFLINE	The operation cannot be performed on an offline library.
4306	0x000010D2	ERROR_EMPTY	The library, drive, or media pool is empty.
4307	0x000010D3	ERROR_NOT_EMPTY	The library, drive, or media pool must be empty to perform this operation.
4308	0x000010D4	ERROR_MEDIA_UNAVAILABLE	No media is currently available in this media pool or library.
4309	0x000010D5	ERROR_RESOURCE_DISABLED	A resource required for this operation is disabled.
4310	0x000010D6	ERROR_INVALID_CLEANER	The media identifier does not represent a valid cleaner.
4311	0x000010D7	ERROR_UNABLE_TO_CLEAN	The drive cannot be cleaned or does not support cleaning.
4312	0x000010D8	ERROR_OBJECT_NOT_FOUND	The object identifier does not represent a valid object.
4313	0x000010D9	ERROR_DATABASE_FAILURE	Unable to read from or write to the database.
4314	0x000010DA	ERROR_DATABASE_FULL	The database is full.
4315	0x000010DB	ERROR_MEDIA_INCOMPATIBLE	The medium is not compatible with the device or media pool.
4316	0x000010DC	ERROR_RESOURCE_NOT_PRESENT	The resource required for this operation does not exist.
4317	0x000010DD	ERROR_INVALID_OPERATION	The operation identifier is not valid.
4318	0x000010DE	ERROR_MEDIA_NOT_AVAILABLE	The media is not mounted or ready for use.
4319	0x000010DF	ERROR_DEVICE_NOT_AVAILABLE	The device is not ready for use.
4320	0x000010E0	ERROR_REQUEST_REFUSED	The operator or administrator has refused the request.
4321	0x000010E1	ERROR_INVALID_DRIVE_OBJECT	The drive identifier does not represent a valid drive.
4322	0x000010E2	ERROR_LIBRARY_FULL	Library is full. No slot is available for use.
4323	0x000010E3	ERROR_MEDIUM_NOT_ACCESSIBLE	The transport cannot access the medium.
4324	0x000010E4	ERROR_UNABLE_TO_LOAD_MEDIUM	Unable to load the medium into the drive.
4325	0x000010E5	ERROR_UNABLE_TO_INVENTORY_DRIVE	Unable to retrieve status about the drive.
4326	0x000010E6	ERROR_UNABLE_TO_INVENTORY_SLOT	Unable to retrieve status about the slot.
4327	0x000010E7	ERROR_UNABLE_TO_INVENTORY_TRANSPORT	Unable to retrieve status about the transport.
4328	0x000010E8	ERROR_TRANSPORT_FULL	Cannot use the transport because it is already in use.
4329	0x000010E9	ERROR_CONTROLLING_IEPORT	Unable to open or close the inject/eject port.
4330	0x000010EA	ERROR_UNABLE_TO_EJECT_MOUNTED_MEDIA	Unable to eject the media because it is in a drive.
4331	0x000010EB	ERROR_CLEANER_SLOT_SET	A cleaner slot is already reserved.
4332	0x000010EC	ERROR_CLEANER_SLOT_NOT_SET	A cleaner slot is not reserved.
4333	0x000010ED	ERROR_CLEANER_CARTRIDGE_SPENT	The cleaner cartridge has performed the maximum number of drive cleanings.
4334	0x000010EE	ERROR_UNEXPECTED_OMID	Unexpected on-medium identifier.
4335	0x000010EF	ERROR_CANT_DELETE_LAST_ITEM	The last remaining item in this group or resource cannot be deleted.
4336	0x000010F0	ERROR_MESSAGE_EXCEEDS_MAX_SIZE	The message provided exceeds the maximum size allowed for this parameter.
4337	0x000010F1	ERROR_VOLUME_CONTAINS_SYS_FILES	The volume contains system or paging files.
4338	0x000010F2	ERROR_INDIGENOUS_TYPE	The media type cannot be removed from this library since at least one drive in the library reports it can support this media type.
4339	0x000010F3	ERROR_NO_SUPPORTING_DRIVES	This offline media cannot be mounted on this system since no enabled drives are present which can be used.
4340	0x000010F4	ERROR_CLEANER_CARTRIDGE_INSTALLED	A cleaner cartridge is present in the tape library.
4350	0x000010FE	ERROR_FILE_OFFLINE	The remote storage service was not able to recall the file.

Error			Description
decimal	Hexadecimal	Name	
4351	0x000010FF	ERROR_REMOTE_STORAGE_NOT_ACTIVE	The remote storage service is not operational at this time.
4352	0x00001100	ERROR_REMOTE_STORAGE_MEDIA_ERROR	The remote storage service encountered a media error.
4390	0x00001126	ERROR_NOT_A_REPARSE_POINT	The file or directory is not a reparse point.
4391	0x00001127	ERROR_REPARSE_ATTRIBUTE_CONFLICT	The reparse point attribute cannot be set because it conflicts with an existing attribute.
4392	0x00001128	ERROR_INVALID_REPARSE_DATA	The data present in the reparse point buffer is invalid.
4393	0x00001129	ERROR_REPARSE_TAG_INVALID	The tag present in the reparse point buffer is invalid.
4394	0x0000112A	ERROR_REPARSE_TAG_MISMATCH	There is a mismatch between the tag specified in the request and the tag present in the reparse point.
4500	0x00001194	ERROR_VOLUME_NOT_SIS_ENABLED	Single Instance Storage is not available on this volume.

Error			Description
decimal	Hexadecimal	Name	
5001	0x00001389	ERROR_DEPENDENT_RESOURCE_EXISTS	The cluster resource cannot be moved to another group because other resources are dependent on it.
5002	0x0000138A	ERROR_DEPENDENCY_NOT_FOUND	The cluster resource dependency cannot be found.
5003	0x0000138B	ERROR_DEPENDENCY_ALREADY_EXISTS	The cluster resource cannot be made dependent on the specified resource because it is already dependent.
5004	0x0000138C	ERROR_RESOURCE_NOT_ONLINE	The cluster resource is not online.
5005	0x0000138D	ERROR_HOST_NODE_NOT_AVAILABLE	A cluster node is not available for this operation.
5006	0x0000138E	ERROR_RESOURCE_NOT_AVAILABLE	The cluster resource is not available.
5007	0x0000138F	ERROR_RESOURCE_NOT_FOUND	The cluster resource could not be found.
5008	0x00001390	ERROR_SHUTDOWN_CLUSTER	The cluster is being shut down.
5009	0x00001391	ERROR_CANT_EVICT_ACTIVE_NODE	A cluster node cannot be evicted from the cluster unless the node is down.
5010	0x00001392	ERROR_OBJECT_ALREADY_EXISTS	The object already exists.
5011	0x00001393	ERROR_OBJECT_IN_LIST	The object is already in the list.
5012	0x00001394	ERROR_GROUP_NOT_AVAILABLE	The cluster group is not available for any new requests.
5013	0x00001395	ERROR_GROUP_NOT_FOUND	The cluster group could not be found.
5014	0x00001396	ERROR_GROUP_NOT_ONLINE	The operation could not be completed because the cluster group is not online.
5015	0x00001397	ERROR_HOST_NODE_NOT_RESOURCE_OWNER	The cluster node is not the owner of the resource.
5016	0x00001398	ERROR_HOST_NODE_NOT_GROUP_OWNER	The cluster node is not the owner of the group.
5017	0x00001399	ERROR_RESMON_CREATE_FAILED	The cluster resource could not be created in the specified resource monitor.
5018	0x0000139A	ERROR_RESMON_ONLINE_FAILED	The cluster resource could not be brought online by the resource monitor.
5019	0x0000139B	ERROR_RESOURCE_ONLINE	The operation could not be completed because the cluster resource is online.
5020	0x0000139C	ERROR_QUORUM_RESOURCE	The cluster resource could not be deleted or brought offline because it is the quorum resource.
5021	0x0000139D	ERROR_NOT_QUORUM_CAPABLE	The cluster could not make the specified resource a quorum resource because it is not capable of being a quorum resource.
5022	0x0000139E	ERROR_CLUSTER_SHUTTING_DOWN	The cluster software is shutting down.
5023	0x0000139F	ERROR_INVALID_STATE	The group or resource is not in the correct state to perform the requested operation.
5024	0x000013A0	ERROR_RESOURCE_PROPERTIES_STORED	The properties were stored but not all changes will take effect until the next time the resource is brought online.
5025	0x000013A1	ERROR_NOT_QUORUM_CLASS	The cluster could not make the specified resource a quorum resource because it does not belong to a shared storage class.
5026	0x000013A2	ERROR_CORE_RESOURCE	The cluster resource could not be deleted since it is a core resource.
5027	0x000013A3	ERROR_QUORUM_RESOURCE_ONLINE_FAILED	The quorum resource failed to come online.
5028	0x000013A4	ERROR_QUORUMLOG_OPEN_FAILED	The quorum log could not be created or mounted successfully.
5029	0x000013A5	ERROR_CLUSTERLOG_CORRUPT	The cluster log is corrupt.
5030	0x000013A6	ERROR_CLUSTERLOG_RECORD_EXCEEDS_MAXSIZE	The record could not be written to the cluster log since it exceeds the maximum size.
5031	0x000013A7	ERROR_CLUSTERLOG_EXCEEDS_MAXSIZE	The cluster log exceeds its maximum size.
5032	0x000013A8	ERROR_CLUSTERLOG_CHKPOINT_NOT_FOUND	No checkpoint record was found in the cluster log.
5033	0x000013A9	ERROR_CLUSTERLOG_NOT_ENOUGH_SPACE	The minimum required disk space needed for logging is not available.
5034	0x000013AA	ERROR_QUORUM_OWNER_ALIVE	The cluster node failed to take control of the quorum resource because the resource is owned by another active node.
5035	0x000013AB	ERROR_NETWORK_NOT_AVAILABLE	A cluster network is not available for this operation.
5036	0x000013AC	ERROR_NODE_NOT_AVAILABLE	A cluster node is not available for this operation.

Error			Description
decimal	Hexadecimal	Name	
5037	0x000013AD	ERROR_ALL_NODES_NOT_AVAILABLE	All cluster nodes must be running to perform this operation.
5038	0x000013AE	ERROR_RESOURCE_FAILED	A cluster resource failed.
5039	0x000013AF	ERROR_CLUSTER_INVALID_NODE	The cluster node is not valid.
5040	0x000013B0	ERROR_CLUSTER_NODE_EXISTS	The cluster node already exists.
5041	0x000013B1	ERROR_CLUSTER_JOIN_IN_PROGRESS	A node is in the process of joining the cluster.
5042	0x000013B2	ERROR_CLUSTER_NODE_NOT_FOUND	The cluster node was not found.
5043	0x000013B3	ERROR_CLUSTER_LOCAL_NODE_NOT_FOUND	The cluster local node information was not found.
5044	0x000013B4	ERROR_CLUSTER_NETWORK_EXISTS	The cluster network already exists.
5045	0x000013B5	ERROR_CLUSTER_NETWORK_NOT_FOUND	The cluster network was not found.
5046	0x000013B6	ERROR_CLUSTER_NETINTERFACE_EXISTS	The cluster network interface already exists.
5047	0x000013B7	ERROR_CLUSTER_NETINTERFACE_NOT_FOUND	The cluster network interface was not found.
5048	0x000013B8	ERROR_CLUSTER_INVALID_REQUEST	The cluster request is not valid for this object.
5049	0x000013B9	ERROR_CLUSTER_INVALID_NETWORK_PROVIDER	The cluster network provider is not valid.
5050	0x000013BA	ERROR_CLUSTER_NODE_DOWN	The cluster node is down.
5051	0x000013BB	ERROR_CLUSTER_NODE_UNREACHABLE	The cluster node is not reachable.
5052	0x000013BC	ERROR_CLUSTER_NODE_NOT_MEMBER	The cluster node is not a member of the cluster.
5053	0x000013BD	ERROR_CLUSTER_JOIN_NOT_IN_PROGRESS	A cluster join operation is not in progress.
5054	0x000013BE	ERROR_CLUSTER_INVALID_NETWORK	The cluster network is not valid.
5056	0x000013C0	ERROR_CLUSTER_NODE_UP	The cluster node is up.
5057	0x000013C1	ERROR_CLUSTER_IPADDR_IN_USE	The cluster IP address is already in use.
5058	0x000013C2	ERROR_CLUSTER_NODE_NOT_PAUSED	The cluster node is not paused.
5059	0x000013C3	ERROR_CLUSTER_NO_SECURITY_CONTEXT	No cluster security context is available.
5060	0x000013C4	ERROR_CLUSTER_NETWORK_NOT_INTERNAL	The cluster network is not configured for internal cluster communication.
5061	0x000013C5	ERROR_CLUSTER_NODE_ALREADY_UP	The cluster node is already up.
5062	0x000013C6	ERROR_CLUSTER_NODE_ALREADY_DOWN	The cluster node is already down.
5063	0x000013C7	ERROR_CLUSTER_NETWORK_ALREADY_ONLINE	The cluster network is already online.
5064	0x000013C8	ERROR_CLUSTER_NETWORK_ALREADY_OFFLINE	The cluster network is already offline.
5065	0x000013C9	ERROR_CLUSTER_NODE_ALREADY_MEMBER	The cluster node is already a member of the cluster.
5066	0x000013CA	ERROR_CLUSTER_LAST_INTERNAL_NETWORK	The cluster network is the only one configured for internal cluster communication between two or more active cluster nodes. The internal communication capability cannot be removed from the network.
5067	0x000013CB	ERROR_CLUSTER_NETWORK_HAS_DEPENDENTS	One or more cluster resources depend on the network to provide service to clients. The client access capability cannot be removed from the network.
5068	0x000013CC	ERROR_INVALID_OPERATION_ON_QUORUM	This operation cannot be performed on the cluster resource as it the quorum resource. You may not bring the quorum resource offline or modify its possible owners list.
5069	0x000013CD	ERROR_DEPENDENCY_NOT_ALLOWED	The cluster quorum resource is not allowed to have any dependencies.
5070	0x000013CE	ERROR_CLUSTER_NODE_PAUSED	The cluster node is paused.
5071	0x000013CF	ERROR_NODE_CANT_HOST_RESOURCE	The cluster resource cannot be brought online. The owner node cannot run this resource.
5072	0x000013D0	ERROR_CLUSTER_NODE_NOT_READY	The cluster node is not ready to perform the requested operation.
5073	0x000013D1	ERROR_CLUSTER_NODE_SHUTTING_DOWN	The cluster node is shutting down.
5074	0x000013D2	ERROR_CLUSTER_JOIN_ABORTED	The cluster join operation was aborted.
5075	0x000013D3	ERROR_CLUSTER_INCOMPATIBLE_VERSIONS	The cluster join operation failed due to incompatible software versions between the joining node and its sponsor.
5076	0x000013D4	ERROR_CLUSTER_MAXNUM_OF_RESOURCES_EXCEEDED	This resource cannot be created because the cluster has reached the limit on the number of resources it can monitor.

Error			Description
decimal	Hexadecimal	Name	
5077	0x000013D5	ERROR_CLUSTER_SYSTEM_CONFIG_CHANGED	The system configuration changed during the cluster join or form operation. The join or form operation was aborted.
5078	0x000013D6	ERROR_CLUSTER_RESOURCE_TYPE_NOT_FOUND	The specified resource type was not found.
5079	0x000013D7	ERROR_CLUSTER_RESTYPE_NOT_SUPPORTED	The specified node does not support a resource of this type. This may be due to version inconsistencies or due to the absence of the resource DLL on this node.
5080	0x000013D8	ERROR_CLUSTER_RESNAME_NOT_FOUND	The specified resource name is supported by this resource DLL. This may be due to a bad (or changed) name supplied to the resource DLL.
5081	0x000013D9	ERROR_CLUSTER_NO_RPC_PACKAGES_REGISTERED	No authentication package could be registered with the RPC server.
5082	0x000013DA	ERROR_CLUSTER_OWNER_NOT_IN_PREFLIST	You cannot bring the group online because the owner of the group is not in the preferred list for the group. To change the owner node for the group, move the group.
5083	0x000013DB	ERROR_CLUSTER_DATABASE_SEQMISMATCH	The join operation failed because the cluster database sequence number has changed or is incompatible with the locker node. This may happen during a join operation if the cluster database was changing during the join.
5084	0x000013DC	ERROR_RESMON_INVALID_STATE	The resource monitor will not allow the fail operation to be performed while the resource is in its current state. This may happen if the resource is in a pending state.
5085	0x000013DD	ERROR_CLUSTER_GUM_NOT_LOCKER	A non locker code got a request to reserve the lock for making global updates.
5086	0x000013DE	ERROR_QUORUM_DISK_NOT_FOUND	The quorum disk could not be located by the cluster service.
5087	0x000013DF	ERROR_DATABASE_BACKUP_CORRUPT	The backup up cluster database is possibly corrupt.
5088	0x000013E0	ERROR_CLUSTER_NODE_ALREADY_HAS_DFS_ROOT	A DFS root already exists in this cluster node.
5089	0x000013E1	ERROR_RESOURCE_PROPERTY_UNCHANGEABLE	An attempt to modify a resource property failed because it conflicts with another existing property.
5890	0x00001702	ERROR_CLUSTER_MEMBERSHIP_INVALID_STATE	An operation was attempted that is incompatible with the current membership state of the node.
5891	0x00001703	ERROR_CLUSTER_QUORUMLOG_NOT_FOUND	The quorum resource does not contain the quorum log.
5892	0x00001704	ERROR_CLUSTER_MEMBERSHIP_HALT	The membership engine requested shutdown of the cluster service on this node.
5893	0x00001705	ERROR_CLUSTER_INSTANCE_ID_MISMATCH	The join operation failed because the cluster instance ID of the joining node does not match the cluster instance ID of the sponsor node.
5894	0x00001706	ERROR_CLUSTER_NETWORK_NOT_FOUND_FOR_IP	A matching network for the specified IP address could not be found. Please also specify a subnet mask and a cluster network.
5895	0x00001707	ERROR_CLUSTER_PROPERTY_DATA_TYPE_MISMATCH	The actual data type of the property did not match the expected data type of the property.
5896	0x00001708	ERROR_CLUSTER_EVICT_WITHOUT_CLEANUP	The cluster node was evicted from the cluster successfully. The node was not cleaned up because it does not support the evict cleanup functionality.

Error			Description
decimal	Hexadecimal	Name	
6000	0x00001770	ERROR_ENCRYPTION_FAILED	The specified file could not be encrypted.
6001	0x00001771	ERROR_DECRYPTION_FAILED	The specified file could not be decrypted.
6002	0x00001772	ERROR_FILE_ENCRYPTED	The specified file is encrypted and the user does not have the ability to decrypt it.
6003	0x00001773	ERROR_NO_RECOVERY_POLICY	There is no valid encryption recovery policy configured for this system.
6004	0x00001774	ERROR_NO_EFS	The required encryption driver is not loaded for this system.
6005	0x00001775	ERROR_WRONG_EFS	The file was encrypted with a different encryption driver than is currently loaded.
6006	0x00001776	ERROR_NO_USER_KEYS	There are no EFS keys defined for the user.
6007	0x00001777	ERROR_FILE_NOT_ENCRYPTED	The specified file is not encrypted.
6008	0x00001778	ERROR_NOT_EXPORT_FORMAT	The specified file is not in the defined EFS export format.
6009	0x00001779	ERROR_FILE_READ_ONLY	The specified file is read only.
6010	0x0000177A	ERROR_DIR_EFS_DISALLOWED	The directory has been disabled for encryption.
6011	0x0000177B	ERROR_EFS_SERVER_NOT_TRUSTED	The server is not trusted for remote encryption operation.
6012	0x0000177C	ERROR_BAD_RECOVERY_POLICY	Recovery policy configured for this system contains invalid recovery certificate.
6013	0x0000177D	ERROR_EFS_ALG_BLOB_TOO_BIG	The encryption algorithm used on the source file needs a bigger key buffer than the one on the destination file.
6014	0x0000177E	ERROR_VOLUME_NOT_SUPPORT_EFS	The disk partition does not support file encryption.
6118	0x000017E6	ERROR_NO_BROWSER_SERVERS_FOUND	The list of servers for this workgroup is not currently available.
6200	0x00001838	SCHED_E_SERVICE_NOT_LOCALSYSTEM	The Task Scheduler service must be configured to run in the System account to function properly. Individual tasks may be configured to run in other accounts.
7001	0x00001B59	ERROR_CTX_WINSTATION_NAME_INVALID	The specified session name is invalid.
7002	0x00001B5A	ERROR_CTX_INVALID_PD	The specified protocol driver is invalid.
7003	0x00001B5B	ERROR_CTX_PD_NOT_FOUND	The specified protocol driver was not found in the system path.
7004	0x00001B5C	ERROR_CTX_WD_NOT_FOUND	The specified terminal connection driver was not found in the system path.
7005	0x00001B5D	ERROR_CTX_CANNOT_MAKE_EVENTLOG_ENTRY	A registry key for event logging could not be created for this session.
7006	0x00001B5E	ERROR_CTX_SERVICE_NAME_COLLISION	A service with the same name already exists on the system.
7007	0x00001B5F	ERROR_CTX_CLOSE_PENDING	A close operation is pending on the session.
7008	0x00001B60	ERROR_CTX_NO_OUTBUF	There are no free output buffers available.
7009	0x00001B61	ERROR_CTX_MODEM_INF_NOT_FOUND	The MODEM.INF file was not found.
7010	0x00001B62	ERROR_CTX_INVALID_MODEMNAME	The modem name was not found in MODEM.INF.
7011	0x00001B63	ERROR_CTX_MODEM_RESPONSE_ERROR	The modem did not accept the command sent to it. Verify that the configured modem name matches the attached modem.
7012	0x00001B64	ERROR_CTX_MODEM_RESPONSE_TIMEOUT	The modem did not respond to the command sent to it. Verify that the modem is properly cabled and powered on.
7013	0x00001B65	ERROR_CTX_MODEM_RESPONSE_NO_CARRIER	Carrier detect has failed or carrier has been dropped due to disconnect.
7014	0x00001B66	ERROR_CTX_MODEM_RESPONSE_NO_DIALTONE	Dial tone not detected within the required time. Verify that the phone cable is properly attached and functional.
7015	0x00001B67	ERROR_CTX_MODEM_RESPONSE_BUSY	Busy signal detected at remote site on callback.
7016	0x00001B68	ERROR_CTX_MODEM_RESPONSE_VOICE	Voice detected at remote site on callback.
7017	0x00001B69	ERROR_CTX_TD_ERROR	Transport driver error
7022	0x00001B6E	ERROR_CTX_WINSTATION_NOT_FOUND	The specified session cannot be found.
7023	0x00001B6F	ERROR_CTX_WINSTATION_ALREADY_EXISTS	The specified session name is already in use.
7024	0x00001B70	ERROR_CTX_WINSTATION_BUSY	The requested operation cannot be completed because the terminal connection is currently busy processing a connect, disconnect, reset, or delete operation.

Error			Description
decimal	Hexadecimal	Name	
7025	0x00001B71	ERROR_CTX_BAD_VIDEO_MODE	An attempt has been made to connect to a session whose video mode is not supported by the current client.
7035	0x00001B7B	ERROR_CTX_GRAPHICS_INVALID	The application attempted to enable DOS graphics mode. DOS graphics mode is not supported.
7037	0x00001B7D	ERROR_CTX_LOGON_DISABLED	Your interactive logon privilege has been disabled. Please contact your administrator.
7038	0x00001B7E	ERROR_CTX_NOT_CONSOLE	The requested operation can be performed only on the system console. This is most often the result of a driver or system DLL requiring direct console access.
7040	0x00001B80	ERROR_CTX_CLIENT_QUERY_TIMEOUT	The client failed to respond to the server connect message.
7041	0x00001B81	ERROR_CTX_CONSOLE_DISCONNECT	Disconnecting the console session is not supported.
7042	0x00001B82	ERROR_CTX_CONSOLE_CONNECT	Reconnecting a disconnected session to the console is not supported.
7044	0x00001B84	ERROR_CTX_SHADOW_DENIED	The request to control another session remotely was denied.
7045	0x00001B85	ERROR_CTX_WINSTATION_ACCESS_DENIED	The requested session access is denied.
7049	0x00001B89	ERROR_CTX_INVALID_WD	The specified terminal connection driver is invalid.
7050	0x00001B8A	ERROR_CTX_SHADOW_INVALID	The requested session cannot be controlled remotely. This may be because the session is disconnected or does not currently have a user logged on.
7051	0x00001B8B	ERROR_CTX_SHADOW_DISABLED	The requested session is not configured to allow remote control.
7052	0x00001B8C	ERROR_CTX_CLIENT_LICENSE_IN_USE	Your request to connect to this Terminal Server has been rejected. Your Terminal Server client license number is currently being used by another user. Please call your system administrator to obtain a unique license number.
7053	0x00001B8D	ERROR_CTX_CLIENT_LICENSE_NOT_SET	Your request to connect to this Terminal Server has been rejected. Your Terminal Server client license number has not been entered for this copy of the Terminal Server client. Please contact your system administrator.
7054	0x00001B8E	ERROR_CTX_LICENSE_NOT_AVAILABLE	The system has reached its licensed logon limit. Please try again later.
7055	0x00001B8F	ERROR_CTX_LICENSE_CLIENT_INVALID	The client you are using is not licensed to use this system. Your logon request is denied.
7056	0x00001B90	ERROR_CTX_LICENSE_EXPIRED	The system license has expired. Your logon request is denied.
7057	0x00001B91	ERROR_CTX_SHADOW_NOT_RUNNING	Remote control could not be terminated because the specified session is not currently being remotely controlled.

Error			Description
decimal	Hexadecimal	Name	
8001	0x00001F41	FRS_ERR_INVALID_API_SEQUENCE	The file replication service API was called incorrectly.
8002	0x00001F42	FRS_ERR_STARTING_SERVICE	The file replication service cannot be started.
8003	0x00001F43	FRS_ERR_STOPPING_SERVICE	The file replication service cannot be stopped.
8004	0x00001F44	FRS_ERR_INTERNAL_API	The file replication service API terminated the request. The event log may have more information.
8005	0x00001F45	FRS_ERR_INTERNAL	The file replication service terminated the request. The event log may have more information.
8006	0x00001F46	FRS_ERR_SERVICE_COMM	The file replication service cannot be contacted. The event log may have more information.
8007	0x00001F47	FRS_ERR_INSUFFICIENT_PRIV	The file replication service cannot satisfy the request because the user has insufficient privileges. The event log may have more information.
8008	0x00001F48	FRS_ERR_AUTHENTICATION	The file replication service cannot satisfy the request because authenticated RPC is not available. The event log may have more information.
8009	0x00001F49	FRS_ERR_PARENT_INSUFFICIENT_PRIV	The file replication service cannot satisfy the request because the user has insufficient privileges on the domain controller. The event log may have more information.
8010	0x00001F4A	FRS_ERR_PARENT_AUTHENTICATION	The file replication service cannot satisfy the request because authenticated RPC is not available on the domain controller. The event log may have more information.
8011	0x00001F4B	FRS_ERR_CHILD_TO_PARENT_COMM	The file replication service cannot communicate with the file replication service on the domain controller. The event log may have more information.
8012	0x00001F4C	FRS_ERR_PARENT_TO_CHILD_COMM	The file replication service on the domain controller cannot communicate with the file replication service on this computer. The event log may have more information.
8013	0x00001F4D	FRS_ERR_SYSVOL_POPULATE	The file replication service cannot populate the system volume because of an internal error. The event log may have more information.
8014	0x00001F4E	FRS_ERR_SYSVOL_POPULATE_TIMEOUT	The file replication service cannot populate the system volume because of an internal timeout. The event log may have more information.
8015	0x00001F4F	FRS_ERR_SYSVOL_IS_BUSY	The file replication service cannot process the request. The system volume is busy with a previous request.
8016	0x00001F50	FRS_ERR_SYSVOL_DEMOTE	The file replication service cannot stop replicating the system volume because of an internal error. The event log may have more information.
8017	0x00001F51	FRS_ERR_INVALID_SERVICE_PARAMETER	The file replication service detected an invalid parameter.
8200	0x00002008	ERROR_DS_NOT_INSTALLED	An error occurred while installing the directory service. For more information, see the event log.
8201	0x00002009	ERROR_DS_MEMBERSHIP_EVALUATED_LOCALLY	The directory service evaluated group memberships locally.
8202	0x0000200A	ERROR_DS_NO_ATTRIBUTE_OR_VALUE	The specified directory service attribute or value does not exist.
8203	0x0000200B	ERROR_DS_INVALID_ATTRIBUTE_SYNTAX	The attribute syntax specified to the directory service is invalid.
8204	0x0000200C	ERROR_DS_ATTRIBUTE_TYPE_UNDEFINED	The attribute type specified to the directory service is not defined.
8205	0x0000200D	ERROR_DS_ATTRIBUTE_OR_VALUE_EXISTS	The specified directory service attribute or value already exists.
8206	0x0000200E	ERROR_DS_BUSY	The directory service is busy.
8207	0x0000200F	ERROR_DS_UNAVAILABLE	The directory service is unavailable.
8208	0x00002010	ERROR_DS_NO_RIDS_ALLOCATED	The directory service was unable to allocate a relative identifier.
8209	0x00002011	ERROR_DS_NO_MORE_RIDS	The directory service has exhausted the pool of relative identifiers.

Error			Description
decimal	Hexadecimal	Name	
8210	0x00002012	ERROR_DS_INCORRECT_ROLE_OWNER	The requested operation could not be performed because the directory service is not the master for that type of operation.
8211	0x00002013	ERROR_DS_RIDMGR_INIT_ERROR	The directory service was unable to initialize the subsystem that allocates relative identifiers.
8212	0x00002014	ERROR_DS_OBJ_CLASS_VIOLATION	The requested operation did not satisfy one or more constraints associated with the class of the object.
8213	0x00002015	ERROR_DS_CANT_ON_NON_LEAF	The directory service can perform the requested operation only on a leaf object.
8214	0x00002016	ERROR_DS_CANT_ON_RDN	The directory service cannot perform the requested operation on the RDN attribute of an object.
8215	0x00002017	ERROR_DS_CANT_MOD_OBJ_CLASS	The directory service detected an attempt to modify the object class of an object.
8216	0x00002018	ERROR_DS_CROSS_DOM_MOVE_ERROR	The requested cross-domain move operation could not be performed.
8217	0x00002019	ERROR_DS_GC_NOT_AVAILABLE	Unable to contact the global catalog server.
8218	0x0000201A	ERROR_SHARED_POLICY	The policy object is shared and can only be modified at the root.
8219	0x0000201B	ERROR_POLICY_OBJECT_NOT_FOUND	The policy object does not exist.
8220	0x0000201C	ERROR_POLICY_ONLY_IN_DS	The requested policy information is only in the directory service.
8221	0x0000201D	ERROR_PROMOTION_ACTIVE	A domain controller promotion is currently active.
8222	0x0000201E	ERROR_NO_PROMOTION_ACTIVE	A domain controller promotion is not currently active
8224	0x00002020	ERROR_DS_OPERATIONS_ERROR	An operations error occurred.
8225	0x00002021	ERROR_DS_PROTOCOL_ERROR	A protocol error occurred.
8226	0x00002022	ERROR_DS_TIMELIMIT_EXCEEDED	The time limit for this request was exceeded.
8227	0x00002023	ERROR_DS_SIZELIMIT_EXCEEDED	The size limit for this request was exceeded.
8228	0x00002024	ERROR_DS_ADMIN_LIMIT_EXCEEDED	The administrative limit for this request was exceeded.
8229	0x00002025	ERROR_DS_COMPARE_FALSE	The compare response was false.
8230	0x00002026	ERROR_DS_COMPARE_TRUE	The compare response was true.
8231	0x00002027	ERROR_DS_AUTH_METHOD_NOT_SUPPORTED	The requested authentication method is not supported by the server.
8232	0x00002028	ERROR_DS_STRONG_AUTH_REQUIRED	A more secure authentication method is required for this server.
8233	0x00002029	ERROR_DS_INAPPROPRIATE_AUTH	Inappropriate authentication.
8234	0x0000202A	ERROR_DS_AUTH_UNKNOWN	The authentication mechanism is unknown.
8235	0x0000202B	ERROR_DS_REFERRAL	A referral was returned from the server.
8236	0x0000202C	ERROR_DS_UNAVAILABLE_CRIT_EXTENSION	The server does not support the requested critical extension.
8237	0x0000202D	ERROR_DS_CONFIDENTIALITY_REQUIRED	This request requires a secure connection.
8238	0x0000202E	ERROR_DS_INAPPROPRIATE_MATCHING	Inappropriate matching.
8239	0x0000202F	ERROR_DS_CONSTRAINT_VIOLATION	A constraint violation occurred.
8240	0x00002030	ERROR_DS_NO_SUCH_OBJECT	There is no such object on the server.
8241	0x00002031	ERROR_DS_ALIAS_PROBLEM	There is an alias problem.
8242	0x00002032	ERROR_DS_INVALID_DN_SYNTAX	An invalid dn syntax has been specified.
8243	0x00002033	ERROR_DS_IS_LEAF	The object is a leaf object.
8244	0x00002034	ERROR_DS_ALIAS_DEREF_PROBLEM	There is an alias dereferencing problem.
8245	0x00002035	ERROR_DS_UNWILLING_TO_PERFORM	The server is unwilling to process the request.
8246	0x00002036	ERROR_DS_LOOP_DETECT	A loop has been detected.
8247	0x00002037	ERROR_DS_NAMING_VIOLATION	There is a naming violation.
8248	0x00002038	ERROR_DS_OBJECT_RESULTS_TOO_LARGE	The result set is too large.
8249	0x00002039	ERROR_DS_AFFECTS_MULTIPLE_DSAS	The operation affects multiple DSAs
8250	0x0000203A	ERROR_DS_SERVER_DOWN	The server is not operational.
8251	0x0000203B	ERROR_DS_LOCAL_ERROR	A local error has occurred.
8252	0x0000203C	ERROR_DS_ENCODING_ERROR	An encoding error has occurred.
8253	0x0000203D	ERROR_DS_DECODING_ERROR	A decoding error has occurred.
8254	0x0000203E	ERROR_DS_FILTER_UNKNOWN	The search filter cannot be recognized.

Error			Description
decimal	Hexadecimal	Name	
8255	0x0000203F	ERROR_DS_PARAM_ERROR	One or more parameters are illegal.
8256	0x00002040	ERROR_DS_NOT_SUPPORTED	The specified method is not supported.
8257	0x00002041	ERROR_DS_NO_RESULTS_RETURNED	No results were returned.
8258	0x00002042	ERROR_DS_CONTROL_NOT_FOUND	The specified control is not supported by the server.
8259	0x00002043	ERROR_DS_CLIENT_LOOP	A referral loop was detected by the client.
8260	0x00002044	ERROR_DS_REFERRAL_LIMIT_EXCEEDED	The preset referral limit was exceeded.
8261	0x00002045	ERROR_DS_SORT_CONTROL_MISSING	The search requires a SORT control.
8262	0x00002046	ERROR_DS_OFFSET_RANGE_ERROR	The search results exceed the offset range specified.
8301	0x0000206D	ERROR_DS_ROOT_MUST_BE_NC	The root object must be the head of a naming context. The root object cannot have an instantiated parent.
8302	0x0000206E	ERROR_DS_ADD_REPLICA_INHIBITED	The add replica operation cannot be performed. The naming context must be writable in order to create the replica.
8303	0x0000206F	ERROR_DS_ATT_NOT_DEF_IN_SCHEMA	A reference to an attribute that is not defined in the schema occurred.
8304	0x00002070	ERROR_DS_MAX_OBJ_SIZE_EXCEEDED	The maximum size of an object has been exceeded.
8305	0x00002071	ERROR_DS_OBJ_STRING_NAME_EXISTS	An attempt was made to add an object to the directory with a name that is already in use.
8306	0x00002072	ERROR_DS_NO_RDN_DEFINED_IN_SCHEMA	An attempt was made to add an object of a class that does not have an RDN defined in the schema.
8307	0x00002073	ERROR_DS_RDN_DOESNT_MATCH_SCHEMA	An attempt was made to add an object using an RDN that is not the RDN defined in the schema.
8308	0x00002074	ERROR_DS_NO_REQUESTED_ATTS_FOUND	None of the requested attributes were found on the objects.
8309	0x00002075	ERROR_DS_USER_BUFFER_TO_SMALL	The user buffer is too small.
8310	0x00002076	ERROR_DS_ATT_IS_NOT_ON_OBJ	The attribute specified in the operation is not present on the object.
8311	0x00002077	ERROR_DS_ILLEGAL_MOD_OPERATION	Illegal modify operation. Some aspect of the modification is not permitted.
8312	0x00002078	ERROR_DS_OBJ_TOO_LARGE	The specified object is too large.
8313	0x00002079	ERROR_DS_BAD_INSTANCE_TYPE	The specified instance type is not valid.
8314	0x0000207A	ERROR_DS_MASTERDSA_REQUIRED	The operation must be performed at a master DSA.
8315	0x0000207B	ERROR_DS_OBJECT_CLASS_REQUIRED	The object class attribute must be specified.
8316	0x0000207C	ERROR_DS_MISSING_REQUIRED_ATT	A required attribute is missing.
8317	0x0000207D	ERROR_DS_ATT_NOT_DEF_FOR_CLASS	An attempt was made to modify an object to include an attribute that is not legal for its class
8318	0x0000207E	ERROR_DS_ATT_ALREADY_EXISTS	The specified attribute is already present on the object.
8320	0x00002080	ERROR_DS_CANT_ADD_ATT_VALUES	The specified attribute is not present, or has no values.
8321	0x00002081	ERROR_DS_SINGLE_VALUE_CONSTRAINT	Multiple values were specified for an attribute that can have only one value.
8322	0x00002082	ERROR_DS_RANGE_CONSTRAINT	A value for the attribute was not in the acceptable range of values.
8323	0x00002083	ERROR_DS_ATT_VAL_ALREADY_EXISTS	The specified value already exists.
8324	0x00002084	ERROR_DS_CANT_REM_MISSING_ATT	The attribute cannot be removed because it is not present on the object.
8325	0x00002085	ERROR_DS_CANT_REM_MISSING_ATT_VAL	The attribute value cannot be removed because it is not present on the object.
8326	0x00002086	ERROR_DS_ROOT_CANT_BE_SUBREF	The specified root object cannot be a subref.
8327	0x00002087	ERROR_DS_NO_CHAINING	Chaining is not permitted.
8328	0x00002088	ERROR_DS_NO_CHAINED_EVAL	Chained evaluation is not permitted.
8329	0x00002089	ERROR_DS_NO_PARENT_OBJECT	The operation could not be performed because the object's parent is either uninstantiated or deleted.
8330	0x0000208A	ERROR_DS_PARENT_IS_AN_ALIAS	Having a parent that is an alias is not permitted. Aliases are leaf objects.

Error			Description
decimal	Hexadecimal	Name	
8331	0x0000208B	ERROR_DS_CANT_MIX_MASTER_AND_REPS	The object and parent must be of the same type, either both masters or both replicas.
8332	0x0000208C	ERROR_DS_CHILDREN_EXIST	The operation cannot be performed because child objects exist. This operation can only be performed on a leaf object.
8333	0x0000208D	ERROR_DS_OBJ_NOT_FOUND	Directory object not found.
8334	0x0000208E	ERROR_DS_ALIASED_OBJ_MISSING	The aliased object is missing.
8335	0x0000208F	ERROR_DS_BAD_NAME_SYNTAX	The object name has bad syntax.
8336	0x00002090	ERROR_DS_ALIAS_POINTS_TO_ALIAS	It is not permitted for an alias to refer to another alias.
8337	0x00002091	ERROR_DS_CANT_DEREF_ALIAS	The alias cannot be dereferenced.
8338	0x00002092	ERROR_DS_OUT_OF_SCOPE	The operation is out of scope.
8339	0x00002093	ERROR_DS_OBJECT_BEING_REMOVED	The operation cannot continue because the object is in the process of being removed.
8340	0x00002094	ERROR_DS_CANT_DELETE_DSA_OBJ	The DSA object cannot be deleted.
8341	0x00002095	ERROR_DS_GENERIC_ERROR	A directory service error has occurred.
8342	0x00002096	ERROR_DS_DSA_MUST_BE_INT_MASTER	The operation can only be performed on an internal master DSA object.
8343	0x00002097	ERROR_DS_CLASS_NOT_DSA	The object must be of class DSA.
8344	0x00002098	ERROR_DS_INSUFF_ACCESS_RIGHTS	Insufficient access rights to perform the operation.
8345	0x00002099	ERROR_DS_ILLEGAL_SUPERIOR	The object cannot be added because the parent is not on the list of possible superiors.
8346	0x0000209A	ERROR_DS_ATTRIBUTE_OWNED_BY_SAM	Access to the attribute is not permitted because the attribute is owned by the Security Accounts Manager (SAM).
8347	0x0000209B	ERROR_DS_NAME_TOO_MANY_PARTS	The name has too many parts.
8348	0x0000209C	ERROR_DS_NAME_TOO_LONG	The name is too long.
8349	0x0000209D	ERROR_DS_NAME_VALUE_TOO_LONG	The name value is too long.
8350	0x0000209E	ERROR_DS_NAME_UNPARSEABLE	The directory service encountered an error parsing a name.
8351	0x0000209F	ERROR_DS_NAME_TYPE_UNKNOWN	The directory service cannot get the attribute type for a name.
8352	0x000020A0	ERROR_DS_NOT_AN_OBJECT	The name does not identify an object; the name identifies a phantom.
8353	0x000020A1	ERROR_DS_SEC_DESC_TOO_SHORT	The security descriptor is too short.
8354	0x000020A2	ERROR_DS_SEC_DESC_INVALID	The security descriptor is invalid.
8355	0x000020A3	ERROR_DS_NO_DELETED_NAME	Failed to create name for deleted object.
8356	0x000020A4	ERROR_DS_SUBREF_MUST_HAVE_PARENT	The parent of a new subref must exist.
8357	0x000020A5	ERROR_DS_NCNAME_MUST_BE_NC	The object must be a naming context.
8358	0x000020A6	ERROR_DS_CANT_ADD_SYSTEM_ONLY	It is not permitted to add an attribute which is owned by the system.
8359	0x000020A7	ERROR_DS_CLASS_MUST_BE_CONCRETE	The class of the object must be structural; you cannot instantiate an abstract class.
8360	0x000020A8	ERROR_DS_INVALID_DMD	The schema object could not be found.
8361	0x000020A9	ERROR_DS_OBJ_GUID_EXISTS	A local object with this GUID (dead or alive) already exists.
8362	0x000020AA	ERROR_DS_NOT_ON_BACKLINK	The operation cannot be performed on a back link.
8363	0x000020AB	ERROR_DS_NO_CROSSREF_FOR_NC	The cross reference for the specified naming context could not be found.
8364	0x000020AC	ERROR_DS_SHUTTING_DOWN	The operation could not be performed because the directory service is shutting down.
8365	0x000020AD	ERROR_DS_UNKNOWN_OPERATION	The directory service request is invalid.
8366	0x000020AE	ERROR_DS_INVALID_ROLE_OWNER	The role owner attribute could not be read.
8367	0x000020AF	ERROR_DS_COULDNT_CONTACT_FSMO	The requested FSMO operation failed. The current FSMO holder could not be reached.
8368	0x000020B0	ERROR_DS_CROSS_NC_DN_RENAME	Modification of a DN across a naming context is not permitted.
8369	0x000020B1	ERROR_DS_CANT_MOD_SYSTEM_ONLY	The attribute cannot be modified because it is owned by the system.
8370	0x000020B2	ERROR_DS_REPLICATOR_ONLY	Only the replicator can perform this function.
8371	0x000020B3	ERROR_DS_OBJ_CLASS_NOT_DEFINED	The specified class is not defined.

Error			Description
decimal	Hexadecimal	Name	
8372	0x000020B4	ERROR_DS_OBJ_CLASS_NOT_SUBCLASS	The specified class is not a subclass.
8373	0x000020B5	ERROR_DS_NAME_REFERENCE_INVALID	The name reference is invalid.
8374	0x000020B6	ERROR_DS_CROSS_REF_EXISTS	A cross reference already exists.
8375	0x000020B7	ERROR_DS_CANT_DEL_MASTER_CROSSREF	It is not permitted to delete a master cross reference.
8376	0x000020B8	ERROR_DS_SUBTREE_NOTIFY_NOT_NC_HEAD	Subtree notifications are only supported on NC heads.
8377	0x000020B9	ERROR_DS_NOTIFY_FILTER_TOO_COMPLEX	Notification filter is too complex.
8378	0x000020BA	ERROR_DS_DUP_RDN	Schema update failed: duplicate RDN.
8379	0x000020BB	ERROR_DS_DUP_OID	Schema update failed: duplicate OID
8380	0x000020BC	ERROR_DS_DUP_MAPI_ID	Schema update failed: duplicate MAPI identifier.
8381	0x000020BD	ERROR_DS_DUP_SCHEMA_ID_GUID	Schema update failed: duplicate schema-id GUID.
8382	0x000020BE	ERROR_DS_DUP_LDAP_DISPLAY_NAME	Schema update failed: duplicate LDAP display name.
8383	0x000020BF	ERROR_DS_SEMANTIC_ATT_TEST	Schema update failed: range-lower less than range upper
8384	0x000020C0	ERROR_DS_SYNTAX_MISMATCH	Schema update failed: syntax mismatch
8385	0x000020C1	ERROR_DS_EXISTS_IN_MUST_HAVE	Schema deletion failed: attribute is used in must-contain
8386	0x000020C2	ERROR_DS_EXISTS_IN_MAY_HAVE	Schema deletion failed: attribute is used in may-contain
8387	0x000020C3	ERROR_DS_NONEXISTENT_MAY_HAVE	Schema update failed: attribute in may-contain does not exist
8388	0x000020C4	ERROR_DS_NONEXISTENT_MUST_HAVE	Schema update failed: attribute in must-contain does not exist
8389	0x000020C5	ERROR_DS_AUX_CLS_TEST_FAIL	Schema update failed: class in aux-class list does not exist or is not an auxiliary class
8390	0x000020C6	ERROR_DS_NONEXISTENT_POSS_SUP	Schema update failed: class in poss-superiors does not exist
8391	0x000020C7	ERROR_DS_SUB_CLS_TEST_FAIL	Schema update failed: class in subclassof list does not exist or does not satisfy hierarchy rules
8392	0x000020C8	ERROR_DS_BAD_RDN_ATT_ID_SYNTAX	Schema update failed: Rdn-Att-Id has wrong syntax
8393	0x000020C9	ERROR_DS_EXISTS_IN_AUX_CLS	Schema deletion failed: class is used as auxiliary class
8394	0x000020CA	ERROR_DS_EXISTS_IN_SUB_CLS	Schema deletion failed: class is used as sub class
8395	0x000020CB	ERROR_DS_EXISTS_IN_POSS_SUP	Schema deletion failed: class is used as poss superior
8396	0x000020CC	ERROR_DS_RECALCSCHEMA_FAILED	Schema update failed in recalculating validation cache.
8397	0x000020CD	ERROR_DS_TREE_DELETE_NOT_FINISHED	The tree deletion is not finished.
8398	0x000020CE	ERROR_DS_CANT_DELETE	The requested delete operation could not be performed.
8399	0x000020CF	ERROR_DS_ATT_SCHEMA_REQ_ID	Cannot read the governs class identifier for the schema record.
8400	0x000020D0	ERROR_DS_BAD_ATT_SCHEMA_SYNTAX	The attribute schema has bad syntax.
8401	0x000020D1	ERROR_DS_CANT_CACHE_ATT	The attribute could not be cached.
8402	0x000020D2	ERROR_DS_CANT_CACHE_CLASS	The class could not be cached.
8403	0x000020D3	ERROR_DS_CANT_REMOVE_ATT_CACHE	The attribute could not be removed from the cache.
8404	0x000020D4	ERROR_DS_CANT_REMOVE_CLASS_CACHE	The class could not be removed from the cache.
8405	0x000020D5	ERROR_DS_CANT_RETRIEVE_DN	The distinguished name attribute could not be read.
8406	0x000020D6	ERROR_DS_MISSING_SUPREF	A required subref is missing.
8407	0x000020D7	ERROR_DS_CANT_RETRIEVE_INSTANCE	The instance type attribute could not be retrieved.
8408	0x000020D8	ERROR_DS_CODE_INCONSISTENCY	An internal error has occurred.
8409	0x000020D9	ERROR_DS_DATABASE_ERROR	A database error has occurred.
8410	0x000020DA	ERROR_DS_GOVERNSID_MISSING	The attribute GOVERNSID is missing.
8411	0x000020DB	ERROR_DS_MISSING_EXPECTED_ATT	An expected attribute is missing.
8412	0x000020DC	ERROR_DS_NCNAME_MISSING_CR_REF	The specified naming context is missing a cross reference.
8413	0x000020DD	ERROR_DS_SECURITY_CHECKING_ERROR	A security checking error has occurred.

Error			Description
decimal	Hexadecimal	Name	
8414	0x000020DE	ERROR_DS_SCHEMA_NOT_LOADED	The schema is not loaded.
8415	0x000020DF	ERROR_DS_SCHEMA_ALLOC_FAILED	Schema allocation failed. Please check if the machine is running low on memory.
8416	0x000020E0	ERROR_DS_ATT_SCHEMA_REQ_SYNTAX	Failed to obtain the required syntax for the attribute schema.
8417	0x000020E1	ERROR_DS_GCVERIFY_ERROR	The global catalog verification failed. The global catalog is not available or does not support the operation. Some part of the directory is currently not available.
8418	0x000020E2	ERROR_DS_DRA_SCHEMA_MISMATCH	The replication operation failed because of a schema mismatch between the servers involved.
8419	0x000020E3	ERROR_DS_CANT_FIND_DSA_OBJ	The DSA object could not be found.
8420	0x000020E4	ERROR_DS_CANT_FIND_EXPECTED_NC	The naming context could not be found.
8421	0x000020E5	ERROR_DS_CANT_FIND_NC_IN_CACHE	The naming context could not be found in the cache.
8422	0x000020E6	ERROR_DS_CANT_RETRIEVE_CHILD	The child object could not be retrieved.
8423	0x000020E7	ERROR_DS_SECURITY_ILLEGAL_MODIFY	The modification was not permitted for security reasons.
8424	0x000020E8	ERROR_DS_CANT_REPLACE_HIDDEN_REC	The operation cannot replace the hidden record.
8425	0x000020E9	ERROR_DS_BAD_HIERARCHY_FILE	The hierarchy file is invalid.
8426	0x000020EA	ERROR_DS_BUILD_HIERARCHY_TABLE_FAILED	The attempt to build the hierarchy table failed.
8427	0x000020EB	ERROR_DS_CONFIG_PARAM_MISSING	The directory configuration parameter is missing from the registry.
8428	0x000020EC	ERROR_DS_COUNTING_AB_INDICES_FAILED	The attempt to count the address book indices failed.
8429	0x000020ED	ERROR_DS_HIERARCHY_TABLE_MALLOC_FAILED	The allocation of the hierarchy table failed.
8430	0x000020EE	ERROR_DS_INTERNAL_FAILURE	The directory service encountered an internal failure.
8431	0x000020EF	ERROR_DS_UNKNOWN_ERROR	The directory service encountered an unknown failure.
8432	0x000020F0	ERROR_DS_ROOT_REQUIRES_CLASS_TOP	A root object requires a class of 'top'.
8433	0x000020F1	ERROR_DS_REFUSING_FSMO_ROLES	This directory server is shutting down, and cannot take ownership of new floating single-master operation roles.
8434	0x000020F2	ERROR_DS_MISSING_FSMO_SETTINGS	The directory service is missing mandatory configuration information, and is unable to determine the ownership of floating single-master operation roles.
8435	0x000020F3	ERROR_DS_UNABLE_TO_SURRENDER_ROLES	The directory service was unable to transfer ownership of one or more floating single-master operation roles to other servers.
8436	0x000020F4	ERROR_DS_DRA_GENERIC	The replication operation failed.
8437	0x000020F5	ERROR_DS_DRA_INVALID_PARAMETER	An invalid parameter was specified for this replication operation.
8438	0x000020F6	ERROR_DS_DRA_BUSY	The directory service is too busy to complete the replication operation at this time.
8439	0x000020F7	ERROR_DS_DRA_BAD_DN	The distinguished name specified for this replication operation is invalid.
8440	0x000020F8	ERROR_DS_DRA_BAD_NC	The naming context specified for this replication operation is invalid.
8441	0x000020F9	ERROR_DS_DRA_DN_EXISTS	The distinguished name specified for this replication operation already exists.
8442	0x000020FA	ERROR_DS_DRA_INTERNAL_ERROR	The replication system encountered an internal error.
8443	0x000020FB	ERROR_DS_DRA_INCONSISTENT_DIT	The replication operation encountered a database inconsistency.
8444	0x000020FC	ERROR_DS_DRA_CONNECTION_FAILED	The server specified for this replication operation could not be contacted.
8445	0x000020FD	ERROR_DS_DRA_BAD_INSTANCE_TYPE	The replication operation encountered an object with an invalid instance type.
8446	0x000020FE	ERROR_DS_DRA_OUT_OF_MEM	The replication operation failed to allocate memory.
8447	0x000020FF	ERROR_DS_DRA_MAIL_PROBLEM	The replication operation encountered an error with the mail system.

Error			Description
decimal	Hexadecimal	Name	
8448	0x00002100	ERROR_DS_DRA_REF_ALREADY_EXISTS	The replication reference information for the target server already exists.
8449	0x00002101	ERROR_DS_DRA_REF_NOT_FOUND	The replication reference information for the target server does not exist.
8450	0x00002102	ERROR_DS_DRA_OBJ_IS_REP_SOURCE	The naming context cannot be removed because it is replicated to another server.
8451	0x00002103	ERROR_DS_DRA_DB_ERROR	The replication operation encountered a database error.
8452	0x00002104	ERROR_DS_DRA_NO_REPLICA	The naming context is in the process of being removed or is not replicated from the specified server.
8453	0x00002105	ERROR_DS_DRA_ACCESS_DENIED	Replication access was denied.
8454	0x00002106	ERROR_DS_DRA_NOT_SUPPORTED	The requested operation is not supported by this version of the directory service.
8455	0x00002107	ERROR_DS_DRA_RPC_CANCELLED	The replication remote procedure call was cancelled.
8456	0x00002108	ERROR_DS_DRA_SOURCE_DISABLED	The source server is currently rejecting replication requests.
8457	0x00002109	ERROR_DS_DRA_SINK_DISABLED	The destination server is currently rejecting replication requests.
8458	0x0000210A	ERROR_DS_DRA_NAME_COLLISION	The replication operation failed due to a collision of object names.
8459	0x0000210B	ERROR_DS_DRA_SOURCE_REINSTALLED	The replication source has been reinstalled.
8460	0x0000210C	ERROR_DS_DRA_MISSING_PARENT	The replication operation failed because a required parent object is missing.
8461	0x0000210D	ERROR_DS_DRA_PREEMPTED	The replication operation was preempted.
8462	0x0000210E	ERROR_DS_DRA_ABANDON_SYNC	The replication synchronization attempt was abandoned because of a lack of updates.
8463	0x0000210F	ERROR_DS_DRA_SHUTDOWN	The replication operation was terminated because the system is shutting down.
8464	0x00002110	ERROR_DS_DRA_INCOMPATIBLE_PARTIAL_SET	The replication synchronization attempt failed as the destination partial attribute set is not a subset of source partial attribute set.
8465	0x00002111	ERROR_DS_DRA_SOURCE_IS_PARTIAL_REPLICA	The replication synchronization attempt failed because a master replica attempted to sync from a partial replica.
8466	0x00002112	ERROR_DS_DRA_EXTN_CONNECTION_FAILED	The server specified for this replication operation was contacted, but that server was unable to contact an additional server needed to complete the operation.
8467	0x00002113	ERROR_DS_INSTALL_SCHEMA_MISMATCH	The version of the Active Directory schema of the source forest is not compatible with the version of Active Directory on this computer. You must upgrade the operating system on a domain controller in the source forest before this computer can be added as a domain controller to that forest.
8468	0x00002114	ERROR_DS_DUP_LINK_ID	Schema update failed: An attribute with the same link identifier already exists.
8469	0x00002115	ERROR_DS_NAME_ERROR_RESOLVING	Name translation: Generic processing error.
8470	0x00002116	ERROR_DS_NAME_ERROR_NOT_FOUND	Name translation: Could not find the name or insufficient right to see name.
8471	0x00002117	ERROR_DS_NAME_ERROR_NOT_UNIQUE	Name translation: Input name mapped to more than one output name.
8472	0x00002118	ERROR_DS_NAME_ERROR_NO_MAPPING	Name translation: Input name found, but not the associated output format.
8473	0x00002119	ERROR_DS_NAME_ERROR_DOMAIN_ONLY	Name translation: Unable to resolve completely, only the domain was found.
8474	0x0000211A	ERROR_DS_NAME_ERROR_NO_SYNTACTICAL_MAPPING	Name translation: Unable to perform purely syntactical mapping at the client without going out to the wire.
8475	0x0000211B	ERROR_DS_CONSTRUCTED_ATT_MOD	Modification of a constructed att is not allowed.
8476	0x0000211C	ERROR_DS_WRONG_OM_OBJ_CLASS	The OM-Object-Class specified is incorrect for an attribute with the specified syntax.
8477	0x0000211D	ERROR_DS_DRA_REPL_PENDING	The replication request has been posted; waiting for reply.

Error			Description
decimal	Hexadecimal	Name	
8478	0x0000211E	ERROR_DS_DS_REQUIRED	The requested operation requires a directory service, and none was available.
8479	0x0000211F	ERROR_DS_INVALID_LDAP_DISPLAY_NAME	The LDAP display name of the class or attribute contains non-ASCII characters.
8480	0x00002120	ERROR_DS_NON_BASE_SEARCH	The requested search operation is only supported for base searches.
8481	0x00002121	ERROR_DS_CANT_RETRIEVE_ATTS	The search failed to retrieve attributes from the database.
8482	0x00002122	ERROR_DS_BACKLINK_WITHOUT_LINK	The schema update operation tried to add a backward link attribute that has no corresponding forward link.
8483	0x00002123	ERROR_DS_EPOCH_MISMATCH	Source and destination of a cross domain move do not agree on the object's epoch number. Either source or destination does not have the latest version of the object.
8484	0x00002124	ERROR_DS_SRC_NAME_MISMATCH	Source and destination of a cross domain move do not agree on the object's current name. Either source or destination does not have the latest version of the object.
8485	0x00002125	ERROR_DS_SRC_AND_DST_NC_IDENTICAL	Source and destination of a cross domain move operation are identical. Caller should use local move operation instead of cross domain move operation.
8486	0x00002126	ERROR_DS_DST_NC_MISMATCH	Source and destination for a cross domain move are not in agreement on the naming contexts in the forest. Either source or destination does not have the latest version of the Partitions container.
8487	0x00002127	ERROR_DS_NOT_AUTHORITIVE_FOR_DST_NC	Destination of a cross domain move is not authoritative for the destination naming context.
8488	0x00002128	ERROR_DS_SRC_GUID_MISMATCH	Source and destination of a cross domain move do not agree on the identity of the source object. Either source or destination does not have the latest version of the source object.
8489	0x00002129	ERROR_DS_CANT_MOVE_DELETED_OBJECT	Object being moved across domains is already known to be deleted by the destination server. The source server does not have the latest version of the source object.
8490	0x0000212A	ERROR_DS_PDC_OPERATION_IN_PROGRESS	Another operation which requires exclusive access to the PDC PSMO is already in progress.
8491	0x0000212B	ERROR_DS_CROSS_DOMAIN_CLEANUP_REQD	A cross domain move operation failed such that the two versions of the moved object exist - one each in the source and destination domains. The destination object needs to be removed to restore the system to a consistent state.
8492	0x0000212C	ERROR_DS_ILLEGAL_XDOM_MOVE_OPERATION	This object may not be moved across domain boundaries either because cross domain moves for this class are disallowed, or the object has some special characteristics, eg: trust account or restricted RID, which prevent its move.
8493	0x0000212D	ERROR_DS_CANT_WITH_ACCT_GROUP_MEMBERSHPS	Can't move objects with memberships across domain boundaries as once moved, this would violate the membership conditions of the account group. Remove the object from any account group memberships and retry.
8494	0x0000212E	ERROR_DS_NC_MUST_HAVE_NC_PARENT	A naming context head must be the immediate child of another naming context head, not of an interior node.
8495	0x0000212F	ERROR_DS_CR_IMPOSSIBLE_TO_VALIDATE	The directory cannot validate the proposed naming context name because it does not hold a replica of the naming context above the proposed naming context. Please ensure that the domain naming master role is held by a server that is configured as a global catalog server, and that the server is up to date with its replication partners. (Applies only to Windows 2000 Domain Naming masters)
8496	0x00002130	ERROR_DS_DST_DOMAIN_NOT_NATIVE	Destination domain must be in native mode.

Error			Description
decimal	Hexadecimal	Name	
8497	0x00002131	ERROR_DS_MISSING_INFRASTRUCTURE_CONTAINER	The operation can not be performed because the server does not have an infrastructure container in the domain of interest.
8498	0x00002132	ERROR_DS_CANT_MOVE_ACCOUNT_GROUP	Cross-domain move of non-empty account groups is not allowed.
8499	0x00002133	ERROR_DS_CANT_MOVE_RESOURCE_GROUP	Cross-domain move of non-empty resource groups is not allowed.

Error			Description
decimal	Hexadecimal	Name	
8500	0x00002134	ERROR_DS_INVALID_SEARCH_FLAG	The search flags for the attribute are invalid. The ANR bit is valid only on attributes of Unicode or Teletex strings.
8501	0x00002135	ERROR_DS_NO_TREE_DELETE_ABOVE_NC	Tree deletions starting at an object which has an NC head as a descendant are not allowed.
8502	0x00002136	ERROR_DS_COULDNT_LOCK_TREE_FOR_DELETE	The directory service failed to lock a tree in preparation for a tree deletion because the tree was in use.
8503	0x00002137	ERROR_DS_COULDNT_IDENTIFY_OBJECTS_FOR_TREE_DELETE	The directory service failed to identify the list of objects to delete while attempting a tree deletion.
8504	0x00002138	ERROR_DS_SAM_INIT_FAILURE	Security Accounts Manager initialization failed because of the following error: %1. Error Status: 0x%2. Click OK to shut down the system and reboot into Directory Services Restore Mode. Check the event log for detailed information.
8505	0x00002139	ERROR_DS_SENSITIVE_GROUP_VIOLATION	Only an administrator can modify the membership list of an administrative group.
8506	0x0000213A	ERROR_DS_CANT_MOD_PRIMARYGROUPID	Cannot change the primary group ID of a domain controller account.
8507	0x0000213B	ERROR_DS_ILLEGAL_BASE_SCHEMA_MOD	An attempt is made to modify the base schema.
8508	0x0000213C	ERROR_DS_NONSAFE_SCHEMA_CHANGE	Adding a new mandatory attribute to an existing class, deleting a mandatory attribute from an existing class, or adding an optional attribute to the special class Top that is not a backlink attribute (directly or through inheritance, for example, by adding or deleting an auxiliary class) is not allowed.
8509	0x0000213D	ERROR_DS_SCHEMA_UPDATE_DISALLOWED	Schema update is not allowed on this DC because the DC is not the schema FSMO Role Owner.
8510	0x0000213E	ERROR_DS_CANT_CREATE_UNDER_SCHEMA	An object of this class cannot be created under the schema container. You can only create attribute-schema and class-schema objects under the schema container.
8511	0x0000213F	ERROR_DS_INSTALL_NO_SRC_SCH_VERSION	The replica/child install failed to get the objectVersion attribute on the schema container on the source DC. Either the attribute is missing on the schema container or the credentials supplied do not have permission to read it.
8512	0x00002140	ERROR_DS_INSTALL_NO_SCH_VERSION_IN_INIFILE	The replica/child install failed to read the objectVersion attribute in the SCHEMA section of the file schema.ini in the system32 directory.
8513	0x00002141	ERROR_DS_INVALID_GROUP_TYPE	The specified group type is invalid.
8514	0x00002142	ERROR_DS_NO_NEST_GLOBALGROUP_IN_MIXEDDOMAIN	Cannot nest global groups in a mixed domain if the group is security-enabled.
8515	0x00002143	ERROR_DS_NO_NEST_LOCALGROUP_IN_MIXEDDOMAIN	Cannot nest local groups in a mixed domain if the group is security-enabled.
8516	0x00002144	ERROR_DS_GLOBAL_CANT_HAVE_LOCAL_MEMBER	A global group cannot have a local group as a member.
8517	0x00002145	ERROR_DS_GLOBAL_CANT_HAVE_UNIVERSAL_MEMBER	A global group cannot have a universal group as a member.
8518	0x00002146	ERROR_DS_UNIVERSAL_CANT_HAVE_LOCAL_MEMBER	A universal group cannot have a local group as a member.
8519	0x00002147	ERROR_DS_GLOBAL_CANT_HAVE_CROSSDOMAIN_MEMBER	A global group cannot have a cross-domain member.
8520	0x00002148	ERROR_DS_LOCAL_CANT_HAVE_CROSSDOMAIN_LOCAL_MEMBER	A local group cannot have another cross-domain local group as a member.
8521	0x00002149	ERROR_DS_HAVE_PRIMARY_MEMBERS	A group with primary members cannot change to a security-disabled group.
8522	0x0000214A	ERROR_DS_STRING_SD_CONVERSION_FAILED	The schema cache load failed to convert the string default SD on a class-schema object.
8523	0x0000214B	ERROR_DS_NAMING_MASTER_GC	Only DSAs configured to be Global Catalog servers should be allowed to hold the Domain Naming Master FSMO role. (Applies only to Windows 2000 servers)
8524	0x0000214C	ERROR_DS_LOOKUP_FAILURE	The DSA operation is unable to proceed because of a DNS lookup failure.

Error			Description
decimal	Hexadecimal	Name	
8525	0x0000214D	ERROR_DS_COULDNT_UPDATE_SPNS	While processing a change to the DNS Host Name for an object, the Service Principal Name values could not be kept in sync.
8526	0x0000214E	ERROR_DS_CANT_RETRIEVE_SD	The Security Descriptor attribute could not be read.
8527	0x0000214F	ERROR_DS_KEY_NOT_UNIQUE.	The object requested was not found, but an object with that key was found.
8528	0x00002150	ERROR_DS_WRONG_LINKED_ATT_SYNTAX	The syntax of the linked attributed being added is incorrect. Forward links can only have syntax 2.5.5.1, 2.5.5.7, and 2.5.5.14, and backlinks can only have syntax 2.5.5.1.
8529	0x00002151	ERROR_DS_SAM_NEED_BOOTKEY_PASSWORD	Security Account Manager needs to get the boot password.
8530	0x00002152	ERROR_DS_SAM_NEED_BOOTKEY_FLOPPY	Security Account Manager needs to get the boot key from floppy disk.
8531	0x00002153	ERROR_DS_CANT_START	Directory Service cannot start.
8532	0x00002154	ERROR_DS_INIT_FAILURE	Directory Services could not start.
8533	0x00002155	ERROR_DS_NO_PKT_PRIVACY_ON_CONNECTION	The connection between client and server requires packet privacy or better.
8534	0x00002156	ERROR_DS_SOURCE_DOMAIN_IN_FOREST	The source domain may not be in the same forest as destination.
8535	0x00002157	ERROR_DS_DESTINATION_DOMAIN_NOT_IN_FOREST	The destination domain must be in the forest.
8536	0x00002158	ERROR_DS_DESTINATION_AUDITING_NOT_ENABLED	The operation requires that destination domain auditing be enabled.
8537	0x00002159	ERROR_DS_CANT_FIND_DC_FOR_SRC_DOMAIN	The operation couldn't locate a DC for the source domain.
8538	0x0000215A	ERROR_DS_SRC_OBJ_NOT_GROUP_OR_USER	The source object must be a group or user.
8539	0x0000215B	ERROR_DS_SRC_SID_EXISTS_IN_FOREST	The source object's SID already exists in destination forest.
8540	0x0000215C	ERROR_DS_SRC_AND_DST_OBJECT_CLASS_MISMATCH	The source and destination object must be of the same type.
8541	0x0000215D	ERROR_SAM_INIT_FAILURE	Security Accounts Manager initialization failed because of the following error: %1. Error Status: 0x%2. Click OK to shut down the system and reboot into Safe Mode. Check the event log for detailed information.
8542	0x0000215E	ERROR_DS_DRA_SCHEMA_INFO_SHIP	Schema information could not be included in the replication request.
8543	0x0000215F	ERROR_DS_DRA_SCHEMA_CONFLICT	The replication operation could not be completed due to a schema incompatibility.
8544	0x00002160	ERROR_DS_DRA_EARLIER_SCHEMA_CONLICT	The replication operation could not be completed due to a previous schema incompatibility.
8545	0x00002161	ERROR_DS_DRA_OBJ_NC_MISMATCH	The replication update could not be applied because either the source or the destination has not yet received information regarding a recent cross-domain move operation.
8546	0x00002162	ERROR_DS_NC_STILL_HAS_DSAS	The requested domain could not be deleted because there exist domain controllers that still host this domain.
8547	0x00002163	ERROR_DS_GC_REQUIRED	The requested operation can be performed only on a global catalog server.
8548	0x00002164	ERROR_DS_LOCAL_MEMBER_OF_LOCAL_ONLY	A local group can only be a member of other local groups in the same domain.
8549	0x00002165	ERROR_DS_NO_FPO_IN_UNIVERSAL_GROUPS	Foreign security principals cannot be members of universal groups.
8550	0x00002166	ERROR_DS_CANT_ADD_TO_GC	The attribute is not allowed to be replicated to the GC because of security reasons.
8551	0x00002167	ERROR_DS_NO_CHECKPOINT_WITH_PDC	The checkpoint with the PDC could not be taken because there are too many modifications being processed currently.
8552	0x00002168	ERROR_DS_SOURCE_AUDITING_NOT_ENABLED	The operation requires that source domain auditing be enabled.
8553	0x00002169	ERROR_DS_CANT_CREATE_IN_NONDOMAIN_NC	Security principal objects can only be created inside domain naming contexts.

Error			Description
decimal	Hexadecimal	Name	
8554	0x0000216A	ERROR_DS_INVALID_NAME_FOR_SPN	A Service Principal Name (SPN) could not be constructed because the provided hostname is not in the necessary format.
8555	0x0000216B	ERROR_DS_FILTER_USES_CONTRUCTED_ATTRS	A Filter was passed that uses constructed attributes.
8556	0x0000216C	ERROR_DS_UNICODEPWD_NOT_IN_QUOTES	The unicodePwd attribute value must be enclosed in double quotes.
8557	0x0000216D	ERROR_DS_MACHINE_ACCOUNT_QUOTA_EXCEEDED	Your computer could not be joined to the domain. You have exceeded the maximum number of computer accounts you are allowed to create in this domain. Contact your system administrator to have this limit reset or increased.
8558	0x0000216E	ERROR_DS_MUST_BE_RUN_ON_DST_DC	For security reasons, the operation must be run on the destination DC.
8559	0x0000216F	ERROR_DS_SRC_DC_MUST_BE_SP4_OR_GREATER	For security reasons, the source DC must be NT4SP4 or greater.
8560	0x00002170	ERROR_DS_CANT_TREE_DELETE_CRITICAL_OBJ	Critical Directory Service System objects cannot be deleted during tree delete operations. The tree delete may have been partially performed.
8561	0x00002171	ERROR_DS_INIT_FAILURE_CONSOLE	Directory Services could not start because of the following error: %1. Error Status: 0x%2. Please click OK to shutdown the system. You can use the recovery console to diagnose the system further.
8562	0x00002172	ERROR_DS_SAM_INIT_FAILURE_CONSOLE	Security Accounts Manager initialization failed because of the following error: %1. Error Status: 0x%2. Please click OK to shutdown the system. You can use the recovery console to diagnose the system further.
8563	0x00002173	ERROR_DS_FOREST_VERSION_TOO_HIGH	This version of Windows is too old to support the current directory forest behavior. You must upgrade the operating system on this server before it can become a domain controller in this forest.
8564	0x00002174	ERROR_DS_DOMAIN_VERSION_TOO_HIGH	This version of Windows is too old to support the current domain behavior. You must upgrade the operating system on this server before it can become a domain controller in this domain.
8565	0x00002175	ERROR_DS_FOREST_VERSION_TOO_LOW	This version of Windows no longer supports the behavior version in use in this directory forest. You must advance the forest behavior version before this server can become a domain controller in the forest.
8566	0x00002176	ERROR_DS_DOMAIN_VERSION_TOO_LOW	This version of Windows no longer supports the behavior version in use in this domain. You must advance the domain behavior version before this server can become a domain controller in the domain.
8567	0x00002177	ERROR_DS_INCOMPATIBLE_VERSION	The version of Windows is incompatible with the behavior version of the domain or forest.
8568	0x00002178	ERROR_DS_LOW_DSA_VERSION	The behavior version cannot be increased to the requested value because Domain Controllers still exist with versions lower than the requested value.
8569	0x00002179	ERROR_DS_NO_BEHAVIOR_VERSION_IN_MIXEDDOMAIN	The behavior version value cannot be increased while the domain is still in mixed domain mode. You must first change the domain to native mode before increasing the behavior version.
8570	0x0000217A	ERROR_DS_NOT_SUPPORTED_SORT_ORDER	The sort order requested is not supported.
8571	0x0000217B	ERROR_DS_NAME_NOT_UNIQUE	Found an object with a non unique name.
8572	0x0000217C	ERROR_DS_MACHINE_ACCOUNT_CREATED_PRENT4	The machine account was created pre-NT4. The account needs to be recreated.
8573	0x0000217D	ERROR_DS_OUT_OF_VERSION_STORE	The database is out of version store.
8574	0x0000217E	ERROR_DS_INCOMPATIBLE_CONTROLS_USED	Unable to continue operation because multiple conflicting controls were used.
8575	0x0000217F	ERROR_DS_NO_REF_DOMAIN	Unable to find a valid security descriptor reference domain for this partition.

Error			Description
decimal	Hexadecimal	Name	
8576	0x00002180	ERROR_DS_RESERVED_LINK_ID	Schema update failed: The link identifier is reserved.
8577	0x00002181	ERROR_DS_LINK_ID_NOT_AVAILABLE	Schema update failed: There are no link identifiers available.
8578	0x00002182	ERROR_DS_AG_CANT_HAVE_UNIVERSAL_MEMBER	An account group can not have a universal group as a member.
8579	0x00002183	ERROR_DS_MODIFYDN_DISALLOWED_BY_INSTANCE_TYPE	Rename or move operations on naming context heads or read-only objects are not allowed.
8580	0x00002184	ERROR_DS_NO_OBJECT_MOVE_IN_SCHEMA_NC	Move operations on objects in the schema naming context are not allowed.
8581	0x00002185	ERROR_DS_MODIFYDN_DISALLOWED_BY_FLAG	A system flag has been set on the object and does not allow the object to be moved or renamed.
8582	0x00002186	ERROR_DS_MODIFYDN_WRONG_GRANDPARENT	This object is not allowed to change its grandparent container. Moves are not forbidden on this object, but are restricted to sibling containers.
8583	0x00002187	ERROR_DS_NAME_ERROR_TRUST_REFERRAL	Unable to resolve completely, a referral to another forest is generated.
8584	0x00002188	ERROR_NOT_SUPPORTED_ON_STANDARD_SERVER	The requested action is not supported on standard server.
8585	0x00002189	ERROR_DS_CANT_ACCESS_REMOTE_PART_OF_AD	Could not access a partition of the Active Directory located on a remote server. Make sure at least one server is running for the partition in question.
8586	0x0000218A	ERROR_DS_CR_IMPOSSIBLE_TO_VALIDATE_V2	The directory cannot validate the proposed naming context (or partition) name because it does not hold a replica nor can it contact a replica of the naming context above the proposed naming context. Please ensure that the parent naming context is properly registered in DNS, and at least one replica of this naming context is reachable by the Domain Naming master.
8587	0x0000218B	ERROR_DS_THREAD_LIMIT_EXCEEDED	The thread limit for this request was exceeded.
8588	0x0000218C	ERROR_DS_NOT_CLOSEST	The Global catalog server is not in the closet site.

Error			Description
decimal	Hexadecimal	Name	
9001	0x00002329	DNS_ERROR_RCODE_FORMAT_ERROR	DNS server unable to interpret format.
9002	0x0000232A	DNS_ERROR_RCODE_SERVER_FAILURE	DNS server failure.
9003	0x0000232B	DNS_ERROR_RCODE_NAME_ERROR	DNS name does not exist.
9004	0x0000232C	DNS_ERROR_RCODE_NOT_IMPLEMENTED	DNS request not supported by name server.
9005	0x0000232D	DNS_ERROR_RCODE_REFUSED	DNS operation refused.
9006	0x0000232E	DNS_ERROR_RCODE_YXDOMAIN	DNS name that ought not exist, does exist.
9007	0x0000232F	DNS_ERROR_RCODE_YXRRSET	DNS RR set that ought not exist, does exist.
9008	0x00002330	DNS_ERROR_RCODE_NXRRSET	DNS RR set that ought to exist, does not exist.
9009	0x00002331	DNS_ERROR_RCODE_NOTAUTH	DNS server not authoritative for zone.
9010	0x00002332	DNS_ERROR_RCODE_NOTZONE	DNS name in update or prereq is not in zone.
9016	0x00002338	DNS_ERROR_RCODE_BADSIG	DNS signature failed to verify.
9017	0x00002339	DNS_ERROR_RCODE_BADKEY	DNS bad key.
9018	0x0000233A	DNS_ERROR_RCODE_BADTIME	DNS signature validity expired.
9501	0x0000251D	DNS_INFO_NO_RECORDS	No records found for given DNS query.
9502	0x0000251E	DNS_ERROR_BAD_PACKET	Bad DNS packet.
9503	0x0000251F	DNS_ERROR_NO_PACKET	No DNS packet.
9504	0x00002520	DNS_ERROR_RCODE	DNS error, check rcode.
9505	0x00002521	DNS_ERROR_UNSECURE_PACKET	Unsecured DNS packet.
9551	0x0000254F	DNS_ERROR_INVALID_TYPE	Invalid DNS type.
9552	0x00002550	DNS_ERROR_INVALID_IP_ADDRESS	Invalid IP address.
9553	0x00002551	DNS_ERROR_INVALID_PROPERTY	Invalid property.
9554	0x00002552	DNS_ERROR_TRY_AGAIN_LATER	Try DNS operation again later.
9555	0x00002553	DNS_ERROR_NOT_UNIQUE	Record for given name and type is not unique.
9556	0x00002554	DNS_ERROR_NON_RFC_NAME	DNS name does not comply with RFC specifications.
9557	0x00002555	DNS_STATUS_FQDN	DNS name is a fully-qualified DNS name.
9558	0x00002556	DNS_STATUS_DOTTED_NAME	DNS name is dotted (multi-label).
9559	0x00002557	DNS_STATUS_SINGLE_PART_NAME	DNS name is a single-part name.
9560	0x00002558	DNS_ERROR_INVALID_NAME_CHAR	DSN name contains an invalid character.
9561	0x00002559	DNS_ERROR_NUMERIC_NAME	DNS name is entirely numeric.
9601	0x00002581	DNS_ERROR_ZONE_DOES_NOT_EXIST	DNS zone does not exist.
9602	0x00002582	DNS_ERROR_NO_ZONE_INFO	DNS zone information not available.
9603	0x00002583	DNS_ERROR_INVALID_ZONE_OPERATION	Invalid operation for DNS zone.
9604	0x00002584	DNS_ERROR_ZONE_CONFIGURATION_ERROR	Invalid DNS zone configuration.
9605	0x00002585	DNS_ERROR_ZONE_HAS_NO_SOA_RECORD	DNS zone has no start of authority (SOA) record.
9606	0x00002586	DNS_ERROR_ZONE_HAS_NO_NS_RECORDS	DNS zone has no name server (NS) record.
9607	0x00002587	DNS_ERROR_ZONE_LOCKED	DNS zone is locked.
9608	0x00002588	DNS_ERROR_ZONE_CREATION_FAILED	DNS zone creation failed.
9609	0x00002589	DNS_ERROR_ZONE_ALREADY_EXISTS	DNS zone already exists.
9610	0x0000258A	DNS_ERROR_AUTOZONE_ALREADY_EXISTS	DNS automatic zone already exists.
9611	0x0000258B	DNS_ERROR_INVALID_ZONE_TYPE	Invalid DNS zone type.
9612	0x0000258C	DNS_ERROR_SECONDARY_REQUIRES_MASTER_IP	Secondary DNS zone requires master IP address.
9613	0x0000258D	DNS_ERROR_ZONE_NOT_SECONDARY	DNS zone not secondary.
9614	0x0000258E	DNS_ERROR_NEED_SECONDARY_ADDRESSES	Need secondary IP address.
9615	0x0000258F	DNS_ERROR_WINS_INIT_FAILED	WINS initialization failed.
9616	0x00002590	DNS_ERROR_NEED_WINS_SERVERS	Need WINS servers.
9617	0x00002591	DNS_ERROR_NBSTAT_INIT_FAILED	NBTSTAT initialization call failed.
9618	0x00002592	DNS_ERROR_SOA_DELETE_INVALID	Invalid delete of start of authority (SOA)
9619	0x00002593	DNS_ERROR_FORWARDER_ALREADY_EXISTS	A conditional forwarding zone already exists for that name.
9651	0x000025B3	DNS_ERROR_PRIMARY_REQUIRES_DATAFILE	Primary DNS zone requires datafile.
9652	0x000025B4	DNS_ERROR_INVALID_DATAFILE_NAME	Invalid datafile name for DNS zone.
9653	0x000025B5	DNS_ERROR_DATAFILE_OPEN_FAILURE	Failed to open datafile for DNS zone.
9654	0x000025B6	DNS_ERROR_FILE_WRITEBACK_FAILED	Failed to write datafile for DNS zone.
9655	0x000025B7	DNS_ERROR_DATAFILE_PARSING	Failure while reading datafile for DNS zone.
9701	0x000025E5	DNS_ERROR_RECORD_DOES_NOT_EXIST	DNS record does not exist.
9702	0x000025E6	DNS_ERROR_RECORD_FORMAT	DNS record format error.

Error			Description
decimal	Hexadecimal	Name	
9703	0x000025E7	DNS_ERROR_NODE_CREATION_FAILED	Node creation failure in DNS.
9704	0x000025E8	DNS_ERROR_UNKNOWN_RECORD_TYPE	Unknown DNS record type.
9705	0x000025E9	DNS_ERROR_RECORD_TIMED_OUT	DNS record timed out.
9706	0x000025EA	DNS_ERROR_NAME_NOT_IN_ZONE	Name not in DNS zone.
9707	0x000025EB	DNS_ERROR_CNAME_LOOP	CNAME loop detected.
9708	0x000025EC	DNS_ERROR_NODE_IS_CNAME	Node is a CNAME DNS record.
9709	0x000025ED	DNS_ERROR_CNAME_COLLISION	A CNAME record already exists for given name.
9710	0x000025EE	DNS_ERROR_RECORD_ONLY_AT_ZONE_ROOT	Record only at DNS zone root.
9711	0x000025EF	DNS_ERROR_RECORD_ALREADY_EXISTS	DNS record already exists.
9712	0x000025F0	DNS_ERROR_SECONDARY_DATA	Secondary DNS zone data error.
9713	0x000025F1	DNS_ERROR_NO_CREATE_CACHE_DATA	Could not create DNS cache data.
9714	0x000025F2	DNS_ERROR_NAME_DOES_NOT_EXIST	DNS name does not exist.
9715	0x000025F3	DNS_WARNING_PTR_CREATE_FAILED	Could not create pointer (PTR) record.
9716	0x000025F4	DNS_WARNING_DOMAIN_UNDELETED	DNS domain was undeleted.
9717	0x000025F5	DNS_ERROR_DS_UNAVAILABLE	The directory service is unavailable.
9718	0x000025F6	DNS_ERROR_DS_ZONE_ALREADY_EXISTS	DNS zone already exists in the directory service.
9719	0x000025F7	DNS_ERROR_NO_BOOTFILE_IF_DS_ZONE	DNS server not creating or reading the boot file for the directory service integrated DNS zone.
9751	0x00002617	DNS_INFO_AXFR_COMPLETE	DNS AXFR (zone transfer) complete.
9752	0x00002618	DNS_ERROR_AXFR	DNS zone transfer failed.
9753	0x00002619	DNS_INFO_ADDED_LOCAL_WINS	Added local WINS server.
9801	0x00002649	DNS_STATUS_CONTINUE_NEEDED	Secure update call needs to continue update request.
9851	0x0000267B	DNS_ERROR_NO_TCPIP	TCP/IP network protocol not installed.
9852	0x0000267C	DNS_ERROR_NO_DNS_SERVERS	No DNS servers configured for local system.
9901	0x000026AD	DNS_ERROR_DP_DOES_NOT_EXIST	The specified directory partition does not exist.
9902	0x000026AE	DNS_ERROR_DP_ALREADY_EXISTS	The specified directory partition already exists.
9903	0x000026AF	DNS_ERROR_DP_NOT_ENLISTED	The DS is not enlisted in the specified directory partition.
9904	0x000026B0	DNS_ERROR_DP_ALREADY_ENLISTED	The DS is already enlisted in the specified directory partition.

Error			Description
decimal	Hexadecimal	Name	
10004	0x00002714	WSAEINTR	A blocking operation was interrupted by a call to WSACancelBlockingCall.
10009	0x00002719	WSAEBADF	The file handle supplied is not valid.
10013	0x0000271D	WSAEACCES	An attempt was made to access a socket in a way forbidden by its access permissions.
10014	0x0000271E	WSAEFAULT	The system detected an invalid pointer address in attempting to use a pointer argument in a call.
10022	0x00002726	WSAEINVAL	An invalid argument was supplied.
10024	0x00002728	WSAEMFILE	Too many open sockets.
10035	0x00002733	WSAEWOULDBLOCK	A non-blocking socket operation could not be completed immediately.
10036	0x00002734	WSAEINPROGRESS	A blocking operation is currently executing.
10037	0x00002735	WSAEALREADY	An operation was attempted on a non-blocking socket that already had an operation in progress.
10038	0x00002736	WSAENOTSOCK	An operation was attempted on something that is not a socket.
10039	0x00002737	WSAEDESTADDRREQ	A required address was omitted from an operation on a socket.
10040	0x00002738	WSAEMSGSIZE	A message sent on a datagram socket was larger than the internal message buffer or some other network limit, or the buffer used to receive a datagram into was smaller than the datagram itself.
10041	0x00002739	WSAEPROTOTYPE	A protocol was specified in the socket function call that does not support the semantics of the socket type requested.
10042	0x0000273A	WSAENOPROTOOPT	An unknown, invalid, or unsupported option or level was specified in a getsockopt or setsockopt call.
10043	0x0000273B	WSAEPROTONOSUPPORT	The requested protocol has not been configured into the system, or no implementation for it exists.
10044	0x0000273C	WSAESOCKTNOSUPPORT	The support for the specified socket type does not exist in this address family.
10045	0x0000273D	WSAEOPNOTSUPP	The attempted operation is not supported for the type of object referenced.
10046	0x0000273E	WSAEPFNOSUPPORT	The protocol family has not been configured into the system or no implementation for it exists.
10047	0x0000273F	WSAEAFNOSUPPORT	An address incompatible with the requested protocol was used.
10048	0x00002740	WSAEADDRINUSE	Only one usage of each socket address (protocol/ network address/port) is normally permitted.
10049	0x00002741	WSAEADDRNOTAVAIL	The requested address is not valid in its context.
10050	0x00002742	WSAENETDOWN	A socket operation encountered a dead network.
10051	0x00002743	WSAENETUNREACH	A socket operation was attempted to an unreachable network.
10052	0x00002744	WSAENETRESET	The connection has been broken due to keep-alive activity detecting a failure while the operation was in progress.
10053	0x00002745	WSAECONNABORTED	An established connection was aborted by the software in your host machine.
10054	0x00002746	WSAECONNRESET	An existing connection was forcibly closed by the remote host.
10055	0x00002747	WSAENOBUFS	An operation on a socket could not be performed because the system lacked sufficient buffer space or because a queue was full.
10056	0x00002748	WSAEISCONN	A connect request was made on an already connected socket.
10057	0x00002749	WSAENOTCONN	A request to send or receive data was disallowed because the socket is not connected and (when sending on a datagram socket using a sendto call) no address was supplied.
10058	0x0000274A	WSAESHUTDOWN	A request to send or receive data was disallowed because the socket had already been shut down in that direction with a previous shutdown call.
10059	0x0000274B	WSAETOOMANYREFS	Too many references to some kernel object.

Error			Description
decimal	Hexadecimal	Name	
10060	0x0000274C	WSAETIMEDOUT	A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond.
10061	0x0000274D	WSAECONNREFUSED	No connection could be made because the target machine actively refused it.
10062	0x0000274E	WSAELOOP	Cannot translate name.
10063	0x0000274F	WSAENAMETOOLONG	Name component or name was too long.
10064	0x00002750	WSAEHOSTDOWN	A socket operation failed because the destination host was down.
10065	0x00002751	WSAEHOSTUNREACH	A socket operation was attempted to an unreachable host.
10066	0x00002752	WSAENOTEMPTY	Cannot remove a directory that is not empty.
10067	0x00002753	WSAEPROCLIM	A Windows Sockets implementation may have a limit on the number of applications that may use it simultaneously.
10068	0x00002754	WSAEUSERS	Ran out of quota.
10069	0x00002755	WSAEDQUOT	Ran out of disk quota.
10070	0x00002756	WSAESTALE	File handle reference is no longer available.
10071	0x00002757	WSAEREMOTE	Item is not available locally.
10091	0x0000276B	WSASYSNOTREADY	WSAStartup cannot function at this time because the underlying system it uses to provide network services is currently unavailable.
10092	0x0000276C	WSAVERNOTSUPPORTED	The Windows Sockets version requested is not supported.
10093	0x0000276D	WSANOTINITIALISED	Either the application has not called WSAStartup, or WSAStartup failed.
10101	0x00002775	WSAEDISCON	Returned by WSARecv or WSARecvFrom to indicate the remote party has initiated a graceful shutdown sequence.
10102	0x00002776	WSAENOMORE	No more results can be returned by WSALookupServiceNext.
10103	0x00002777	WSAECANCELLED	A call to WSALookupServiceEnd was made while this call was still processing. The call has been canceled.
10104	0x00002778	WSAEINVALIDPROCTABLE	The procedure call table is invalid.
10105	0x00002779	WSAEINVALIDPROVIDER	The requested service provider is invalid.
10106	0x0000277A	WSAEPROVIDERFAILEDINIT	The requested service provider could not be loaded or initialized.
10107	0x0000277B	WSASYSCALLFAILURE	A system call that should never fail has failed.
10108	0x0000277C	WSASERVICE_NOT_FOUND	No such service is known. The service cannot be found in the specified name space.
10109	0x0000277D	WSATYPE_NOT_FOUND	The specified class was not found.
10110	0x0000277E	WSA_E_NO_MORE	No more results can be returned by WSALookupServiceNext.
10111	0x0000277F	WSA_E_CANCELLED	A call to WSALookupServiceEnd was made while this call was still processing. The call has been canceled.
10112	0x00002780	WSAEREFUSED	A database query failed because it was actively refused.
11001	0x00002AF9	WSAHOST_NOT_FOUND	No such host is known.
11002	0x00002AFA	WSATRY_AGAIN	This is usually a temporary error during hostname resolution and means that the local server did not receive a response from an authoritative server.
11003	0x00002AFB	WSANO_RECOVERY	A non-recoverable error occurred during a database lookup.
11004	0x00002AFC	WSANO_DATA	The requested name is valid and was found in the database, but it does not have the correct associated data being resolved for.
11005	0x00002AFD	WSA_QOS_RECEIVERS	At least one reserve has arrived.
11006	0x00002AFE	WSA_QOS_SENDERS	At least one path has arrived.
11007	0x00002AFF	WSA_QOS_NO_SENDERS	There are no senders.
11008	0x00002B00	WSA_QOS_NO_RECEIVERS	There are no receivers.
11009	0x00002B01	WSA_QOS_REQUEST_CONFIRMED	Reserve has been confirmed.

Error			Description
decimal	Hexadecimal	Name	
11010	0x00002B02	WSA_QOS_ADMISSION_FAILURE	Error due to lack of resources.
11011	0x00002B03	WSA_QOS_POLICY_FAILURE	Rejected for administrative reasons - bad credentials.
11012	0x00002B04	WSA_QOS_BAD_STYLE	Unknown or conflicting style.
11013	0x00002B05	WSA_QOS_BAD_OBJECT	Problem with some part of the filterspec or providerspecific buffer in general.
11014	0x00002B06	WSA_QOS_TRAFFIC_CTRL_ERROR	Problem with some part of the flowspec.
11015	0x00002B07	WSA_QOS_GENERIC_ERROR	General QOS error.
11016	0x00002B08	WSA_QOS_ESERVICETYPE	An invalid or unrecognized service type was found in the flowspec.
11017	0x00002B09	WSA_QOS_EFLOWSPEC	An invalid or inconsistent flowspec was found in the QOS structure.
11018	0x00002B0A	WSA_QOS_EPROVSPECBUF	Invalid QOS provider-specific buffer.
11019	0x00002B0B	WSA_QOS_EFILTERSTYLE	An invalid QOS filter style was used.
11020	0x00002B0C	WSA_QOS_EFILTERTYPE	An invalid QOS filter type was used.
11021	0x00002B0D	WSA_QOS_EFILTERCOUNT	An incorrect number of QOS FILTERSPECs were specified in the FLOWDESCRIPTOR.
11022	0x00002B0E	WSA_QOS_EOBJLENGTH	An object with an invalid ObjectLength field was specified in the QOS provider-specific buffer.
11023	0x00002B0F	WSA_QOS_EFLOWCOUNT	An incorrect number of flow descriptors was specified in the QOS structure.
11024	0x00002B10	WSA_QOS_EUNKNOWNPSOBJ	An unrecognized object was found in the QOS provider-specific buffer.
11025	0x00002B11	WSA_QOS_EPOLICYOBJ	An invalid policy object was found in the QOS provider-specific buffer.
11026	0x00002B12	WSA_QOS_EFLOWDESC	An invalid QOS flow descriptor was found in the flow descriptor list.
11027	0x00002B13	WSA_QOS_EPSFLOWSPEC	An invalid or inconsistent flowspec was found in the QOS provider-specific buffer.
11028	0x00002B14	WSA_QOS_EPSFILTERSPEC	An invalid FILTERSPEC was found in the QOS provider-specific buffer.
11029	0x00002B15	WSA_QOS_ESDMODEOBJ	An invalid shape discard mode object was found in the QOS provider-specific buffer.
11030	0x00002B16	WSA_QOS_ESHAPERATEOBJ	An invalid shaping rate object was found in the QOS provider-specific buffer.
11031	0x00002B17	WSA_QOS_RESERVED_PETYPE	A reserved policy element was found in the QOS provider-specific buffer.

Error			Description
decimal	Hexadecimal	Name	
12000	0x00002EE0	ERROR_SXS_SECTION_NOT_FOUND	The requested section was not present in the activation context.
12001	0x00002EE1	ERROR_SXS_CANT_GEN_ACTCTX	This application has failed to start because the application configuration is incorrect. Reinstalling the application may fix this problem.
12002	0x00002EE2	ERROR_SXS_INVALID_ACTCTXDATA_FORMAT	The application binding data format is invalid.
12003	0x00002EE3	ERROR_SXS_ASSEMBLY_NOT_FOUND	The referenced assembly is not installed on your system.
12004	0x00002EE4	ERROR_SXS_MANIFEST_FORMAT_ERROR	The manifest file does not begin with the required tag and format information.
12005	0x00002EE5	ERROR_SXS_MANIFEST_PARSE_ERROR	The manifest file contains one or more syntax errors.
12006	0x00002EE6	ERROR_SXS_ACTIVATION_CONTEXT_DISABLED	The application attempted to activate a disabled activation context.
12007	0x00002EE7	ERROR_SXS_KEY_NOT_FOUND	The requested lookup key was not found in any active activation context.
12008	0x00002EE8	ERROR_SXS_VERSION_CONFLICT	A component version required by the application conflicts with another component version already active.
12009	0x00002EE9	ERROR_SXS_WRONG_SECTION_TYPE	The type requested activation context section does not match the query API used.
12010	0x00002EEA	ERROR_SXS_THREAD_QUERIES_DISABLED	Lack of system resources has required isolated activation to be disabled for the current thread of execution.
12011	0x00002EEB	ERROR_SXS_PROCESS_DEFAULT_ALREADY_SET	An attempt to set the process default activation context failed because the process default activation context was already set.
12012	0x00002EEC	ERROR_SXS_UNKNOWN_ENCODING_GROUP	The encoding group identifier specified is not recognized.
12013	0x00002EED	ERROR_SXS_UNKNOWN_ENCODING	The encoding requested is not recognized.
12014	0x00002EEE	ERROR_SXS_INVALID_XML_NAMESPACE_URI	The manifest contains a reference to an invalid URI.
12015	0x00002EEF	ERROR_SXS_ROOT_MANIFEST_DEPENDENCY_NOT_INSTALLED	The application manifest contains a reference to a dependent assembly which is not installed.
12016	0x00002EF0	ERROR_SXS_LEAF_MANIFEST_DEPENDENCY_NOT_INSTALLED	The manifest for an assembly used by the application has a reference to a dependent assembly which is not installed.
12017	0x00002EF1	ERROR_SXS_INVALID_ASSEMBLY_IDENTITY_ATTRIBUTE	The manifest contains an attribute for the assembly identity which is not valid.
12018	0x00002EF2	ERROR_SXS_MANIFEST_MISSING_REQUIRED_DEFAULT_NAMESPACE	The manifest is missing the required default namespace specification on the assembly element.
12019	0x00002EF3	ERROR_SXS_MANIFEST_INVALID_REQUIRED_DEFAULT_NAMESPACE	The manifest has a default namespace specified on the assembly element but its value is not "urn:schemas-microsoft-com:asm.v1".
12020	0x00002EF4	ERROR_SXS_PRIVATE_MANIFEST_CROSS_PATH_WITH_REPARSE_POINT	The private manifest probe has crossed the reparse-point-associated path.
12021	0x00002EF5	ERROR_SXS_DUPLICATE_DLL_NAME	Two or more components referenced directly or indirectly by the application manifest have files by the same name.
12022	0x00002EF6	ERROR_SXS_DUPLICATE_WINDOWCLASS_NAME	Two or more components referenced directly or indirectly by the application manifest have window classes with the same name.
12023	0x00002EF7	ERROR_SXS_DUPLICATE_CLSID	Two or more components referenced directly or indirectly by the application manifest have the same COM server CLSIDs.
12024	0x00002EF8	ERROR_SXS_DUPLICATE_IID	Two or more components referenced directly or indirectly by the application manifest have proxies for the same COM interface IIDs.
12025	0x00002EF9	ERROR_SXS_DUPLICATE_TLBID	Two or more components referenced directly or indirectly by the application manifest have the same COM type library TLBIDs.
12026	0x00002EFA	ERROR_SXS_DUPLICATE_PROGID	Two or more components referenced directly or indirectly by the application manifest have the same COM ProgIDs.

Error			Description
decimal	Hexadecimal	Name	
12027	0x00002EFB	ERROR_SXS_DUPLICATE_ASSEMBLY_NAME	Two or more components referenced directly or indirectly by the application manifest are different versions of the same component which is not permitted.
12028	0x00002EFC	ERROR_SXS_FILE_HASH_MISMATCH	A component's file does not match the verification information present in the component manifest.
12029	0x00002EFD	ERROR_SXS_POLICY_PARSE_ERROR	The policy manifest contains one or more syntax errors.
12030	0x00002EFE	ERROR_SXS_XML_E_MISSINGQUOTE	Manifest Parse Error : A string literal was expected, but no opening quote character was found.
12031	0x00002EFF	ERROR_SXS_XML_E_COMMENTSYNTAX	Manifest Parse Error : Incorrect syntax was used in a comment.
12032	0x00002F00	ERROR_SXS_XML_E_BADSTARTNAMECHAR	Manifest Parse Error : A name was started with an invalid character.
12033	0x00002F01	ERROR_SXS_XML_E_BADNAMECHAR	Manifest Parse Error : A name contained an invalid character.
12034	0x00002F02	ERROR_SXS_XML_E_BADCHARINSTRING	Manifest Parse Error : A string literal contained an invalid character.
12035	0x00002F03	ERROR_SXS_XML_E_XMLDECLSYNTAX	Manifest Parse Error : Invalid syntax for an XML declaration.
12036	0x00002F04	ERROR_SXS_XML_E_BADCHARDATA	Manifest Parse Error : An invalid character was found in text content.
12037	0x00002F05	ERROR_SXS_XML_E_MISSINGWHITESPACE	Manifest Parse Error : Required white space was missing.
12038	0x00002F06	ERROR_SXS_XML_E_EXPECTINGTAGEND	Manifest Parse Error : The character '>' was expected.
12039	0x00002F07	ERROR_SXS_XML_E_MISSINGSEMICOLON	Manifest Parse Error : A semi colon character was expected.
12040	0x00002F08	ERROR_SXS_XML_E_UNBALANCEDPAREN	Manifest Parse Error : Unbalanced parentheses.
12041	0x00002F09	ERROR_SXS_XML_E_INTERNALERROR	Manifest Parse Error : Internal error.
12042	0x00002F0A	ERROR_SXS_XML_E_UNEXPECTED_WHITESPACE	Manifest Parse Error : White space is not allowed at this location.
12043	0x00002F0B	ERROR_SXS_XML_E_INCOMPLETE_ENCODING	Manifest Parse Error : End of file reached in invalid state for current encoding.
12044	0x00002F0C	ERROR_SXS_XML_E_MISSING_PAREN	Manifest Parse Error : Missing parenthesis.
12045	0x00002F0D	ERROR_SXS_XML_E_EXPECTINGCLOSEQUOTE	Manifest Parse Error : A single or double closing quote character (\' or \") is missing.
12046	0x00002F0E	ERROR_SXS_XML_E_MULTIPLE_COLONS	Manifest Parse Error : Multiple colons are not allowed in a name.
12047	0x00002F0F	ERROR_SXS_XML_E_INVALID_DECIMAL	Manifest Parse Error : Invalid character for decimal digit.
12048	0x00002F10	ERROR_SXS_XML_E_INVALID_HEXIDECIMAL	Manifest Parse Error : Invalid character for hexadecimal digit.
12049	0x00002F11	ERROR_SXS_XML_E_INVALID_UNICODE	Manifest Parse Error : Invalid Unicode character value for this platform.
12050	0x00002F12	ERROR_SXS_XML_E_WHITESPACEORQUESTIONMARK	Manifest Parse Error : Expecting white space or '?'.
12051	0x00002F13	ERROR_SXS_XML_E_UNEXPECTEDENDTAG	Manifest Parse Error : End tag was not expected at this location.
12052	0x00002F14	ERROR_SXS_XML_E_UNCLOSEDTAG	Manifest Parse Error : The following tags were not closed: %1.
12053	0x00002F15	ERROR_SXS_XML_E_DUPLICATEATTRIBUTE	Manifest Parse Error : Duplicate attribute.
12054	0x00002F16	ERROR_SXS_XML_E_MULTIPLEROOTS	Manifest Parse Error : Only one top level element is allowed in an XML document.
12055	0x00002F17	ERROR_SXS_XML_E_INVALIDATROOTLEVEL	Manifest Parse Error : Invalid at the top level of the document.
12056	0x00002F18	ERROR_SXS_XML_E_BADXMLDECL	Manifest Parse Error : Invalid XML declaration.
12057	0x00002F19	ERROR_SXS_XML_E_MISSINGROOT	Manifest Parse Error : XML document must have a top level element.
12058	0x00002F1A	ERROR_SXS_XML_E_UNEXPECTEDEOF	Manifest Parse Error : Unexpected end of file.
12059	0x00002F1B	ERROR_SXS_XML_E_BADPEREFINSUBSET	Manifest Parse Error : Parameter entities cannot be used inside markup declarations in an internal subset.
12060	0x00002F1C	ERROR_SXS_XML_E_UNCLOSEDSTARTTAG	Manifest Parse Error : Element was not closed.

Error			Description
decimal	Hexadecimal	Name	
12061	0x00002F1D	ERROR_SXS_XML_E_UNCLOSEDENDTAG	Manifest Parse Error : End element was missing the character '>'.
12062	0x00002F1E	ERROR_SXS_XML_E_UNCLOSEDSTRING	Manifest Parse Error : A string literal was not closed.
12063	0x00002F1F	ERROR_SXS_XML_E_UNCLOSEDCOMMENT	Manifest Parse Error : A comment was not closed.
12064	0x00002F20	ERROR_SXS_XML_E_UNCLOSEDDECL	Manifest Parse Error : A declaration was not closed.
12065	0x00002F21	ERROR_SXS_XML_E_UNCLOSEDCDATA	Manifest Parse Error : A CDATA section was not closed.
12066	0x00002F22	ERROR_SXS_XML_E_RESERVEDNAMESPACE	Manifest Parse Error : The namespace prefix is not allowed to start with the reserved string "xml".
12067	0x00002F23	ERROR_SXS_XML_E_INVALIDENCODING	Manifest Parse Error : System does not support the specified encoding.
12068	0x00002F24	ERROR_SXS_XML_E_INVALIDSWITCH	Manifest Parse Error : Switch from current encoding to specified encoding not supported.
12069	0x00002F25	ERROR_SXS_XML_E_BADXMLCASE	Manifest Parse Error : The name 'xml' is reserved and must be lower case.
12070	0x00002F26	ERROR_SXS_XML_E_INVALID_STANDALONE	Manifest Parse Error : The standalone attribute must have the value 'yes' or 'no'.
12071	0x00002F27	ERROR_SXS_XML_E_UNEXPECTED_STANDALONE	Manifest Parse Error : The standalone attribute cannot be used in external entities.
12072	0x00002F28	ERROR_SXS_XML_E_INVALID_VERSION	Manifest Parse Error : Invalid version number.
12073	0x00002F29	ERROR_SXS_XML_E_MISSINGEQUALS	Manifest Parse Error : Missing equals sign between attribute and attribute value.
13000	0x000032C8	ERROR_IPSEC_QM_POLICY_EXISTS	The specified quick mode policy already exists.
13001	0x000032C9	ERROR_IPSEC_QM_POLICY_NOT_FOUND	The specified quick mode policy was not found.
13002	0x000032CA	ERROR_IPSEC_QM_POLICY_IN_USE	The specified quick mode policy is being used.
13003	0x000032CB	ERROR_IPSEC_MM_POLICY_EXISTS	The specified main mode policy already exists.
13004	0x000032CC	ERROR_IPSEC_MM_POLICY_NOT_FOUND	The specified main mode policy was not found.
13005	0x000032CD	ERROR_IPSEC_MM_POLICY_IN_USE	The specified main mode policy is being used.
13006	0x000032CE	ERROR_IPSEC_MM_FILTER_EXISTS	The specified main mode filter already exists.
13007	0x000032CF	ERROR_IPSEC_MM_FILTER_NOT_FOUND	The specified main mode filter was not found.
13008	0x000032D0	ERROR_IPSEC_TRANSPORT_FILTER_EXISTS	The specified transport mode filter already exists.
13009	0x000032D1	ERROR_IPSEC_TRANSPORT_FILTER_NOT_FOUND	The specified transport mode filter does not exist.
13010	0x000032D2	ERROR_IPSEC_MM_AUTH_EXISTS	The specified main mode authentication list exists.
13011	0x000032D3	ERROR_IPSEC_MM_AUTH_NOT_FOUND	The specified main mode authentication list was not found.
13012	0x000032D4	ERROR_IPSEC_MM_AUTH_IN_USE	The specified quick mode policy is being used.
13013	0x000032D5	ERROR_IPSEC_DEFAULT_MM_POLICY_NOT_FOUND	The specified main mode policy was not found.
13014	0x000032D6	ERROR_IPSEC_DEFAULT_MM_AUTH_NOT_FOUND	The specified quick mode policy was not found.
13015	0x000032D7	ERROR_IPSEC_DEFAULT_QM_POLICY_NOT_FOUND	The manifest file contains one or more syntax errors.
13016	0x000032D8	ERROR_IPSEC_TUNNEL_FILTER_EXISTS	The application attempted to activate a disabled activation context.
13017	0x000032D9	ERROR_IPSEC_TUNNEL_FILTER_NOT_FOUND	The requested lookup key was not found in any active activation context.
13018	0x000032DA	ERROR_IPSEC_MM_FILTER_PENDING_DELETION	The Main Mode filter is pending deletion.
13019	0x000032DB	ERROR_IPSEC_TRANSPORT_FILTER_PENDING_DELETION	The transport filter is pending deletion.
13020	0x000032DC	ERROR_IPSEC_TUNNEL_FILTER_PENDING_DELETION	The tunnel filter is pending deletion.
13021	0x000032DD	ERROR_IPSEC_MM_POLICY_PENDING_DELETION	The Main Mode policy is pending deletion.
13022	0x000032DE	ERROR_IPSEC_MM_AUTH_PENDING_DELETION	The Main Mode authentication bundle is pending deletion.
13023	0x000032DF	ERROR_IPSEC_QM_POLICY_PENDING_DELETION	The Quick Mode policy is pending deletion.

Error			Description
decimal	Hexadecimal	Name	
13801	0x000035E9	ERROR_IPSEC_IKE_AUTH_FAIL	IKE authentication credentials are unacceptable.
13802	0x000035EA	ERROR_IPSEC_IKE_ATTRIB_FAIL	IKE security attributes are unacceptable.
13803	0x000035EB	ERROR_IPSEC_IKE_NEGOTIATION_PENDING	IKE Negotiation in progress.
13804	0x000035EC	ERROR_IPSEC_IKE_GENERAL_PROCESSING_ERROR	General processing error.
13805	0x000035ED	ERROR_IPSEC_IKE_TIMED_OUT	Negotiation timed out.
13806	0x000035EE	ERROR_IPSEC_IKE_NO_CERT	IKE failed to find valid machine certificate.
13807	0x000035EF	ERROR_IPSEC_IKE_SA_DELETED	IKE SA deleted by peer before establishment completed.
13808	0x000035F0	ERROR_IPSEC_IKE_SA_REAPED	IKE SA deleted before establishment completed.
13809	0x000035F1	ERROR_IPSEC_IKE_MM_ACQUIRE_DROP	Negotiation request sat in Queue too long.
13810	0x000035F2	ERROR_IPSEC_IKE_QM_ACQUIRE_DROP	Negotiation request sat in Queue too long.
13811	0x000035F3	ERROR_IPSEC_IKE_QUEUE_DROP_MM	Negotiation request sat in Queue too long.
13812	0x000035F4	ERROR_IPSEC_IKE_QUEUE_DROP_NO_MM	Negotiation request sat in Queue too long.
13813	0x000035F5	ERROR_IPSEC_IKE_DROP_NO_RESPONSE	No response from peer.
13814	0x000035F6	ERROR_IPSEC_IKE_MM_DELAY_DROP	Negotiation took too long.
13815	0x000035F7	ERROR_IPSEC_IKE_QM_DELAY_DROP	Negotiation took too long.
13816	0x000035F8	ERROR_IPSEC_IKE_ERROR	Unknown error occurred.
13817	0x000035F9	ERROR_IPSEC_IKE_CRL_FAILED	Certificate Revocation Check failed.
13818	0x000035FA	ERROR_IPSEC_IKE_INVALID_KEY_USAGE	Invalid certificate key usage.
13819	0x000035FB	ERROR_IPSEC_IKE_INVALID_CERT_TYPE	Invalid certificate type.
13820	0x000035FC	ERROR_IPSEC_IKE_NO_PRIVATE_KEY	No private key associated with machine certificate.
13822	0x000035FE	ERROR_IPSEC_IKE_DH_FAIL	Failure in Diffie-Helman computation.
13824	0x00003600	ERROR_IPSEC_IKE_INVALID_HEADER	Invalid header.
13825	0x00003601	ERROR_IPSEC_IKE_NO_POLICY	No policy configured.
13826	0x00003602	ERROR_IPSEC_IKE_INVALID_SIGNATURE	Failed to verify signature.
13827	0x00003603	ERROR_IPSEC_IKE_KERBEROS_ERROR	Failed to authenticate using Kerberos.
13828	0x00003604	ERROR_IPSEC_IKE_NO_PUBLIC_KEY	Peer's certificate did not have a public key.
13829	0x00003605	ERROR_IPSEC_IKE_PROCESS_ERR	Error processing error payload.
13830	0x00003606	ERROR_IPSEC_IKE_PROCESS_ERR_SA	Error processing SA payload.
13831	0x00003607	ERROR_IPSEC_IKE_PROCESS_ERR_PROP	Error processing Proposal payload.
13832	0x00003608	ERROR_IPSEC_IKE_PROCESS_ERR_TRANS	Error processing Transform payload.
13833	0x00003609	ERROR_IPSEC_IKE_PROCESS_ERR_KE	Error processing KE payload.
13834	0x0000360A	ERROR_IPSEC_IKE_PROCESS_ERR_ID	Error processing ID payload.
13835	0x0000360B	ERROR_IPSEC_IKE_PROCESS_ERR_CERT	Error processing Cert payload.
13836	0x0000360C	ERROR_IPSEC_IKE_PROCESS_ERR_CERT_REQ	Error processing Certificate Request payload.
13837	0x0000360D	ERROR_IPSEC_IKE_PROCESS_ERR_HASH	Error processing Hash payload.
13838	0x0000360E	ERROR_IPSEC_IKE_PROCESS_ERR_SIG	Error processing Signature payload.
13839	0x0000360F	ERROR_IPSEC_IKE_PROCESS_ERR_NONCE	Error processing Nonce payload.
13840	0x00003610	ERROR_IPSEC_IKE_PROCESS_ERR_NOTIFY	Error processing Notify payload.
13841	0x00003611	ERROR_IPSEC_IKE_PROCESS_ERR_DELETE	Error processing Delete Payload.
13842	0x00003612	ERROR_IPSEC_IKE_PROCESS_ERR_VENDOR	Error processing VendorId payload.
13843	0x00003613	ERROR_IPSEC_IKE_INVALID_PAYLOAD	Invalid payload received.
13844	0x00003614	ERROR_IPSEC_IKE_LOAD_SOFT_SA	Soft SA loaded.
13845	0x00003615	ERROR_IPSEC_IKE_SOFT_SA_TORN_DOWN	Soft SA torn down.
13846	0x00003616	ERROR_IPSEC_IKE_INVALID_COOKIE	Invalid cookie received..
13847	0x00003617	ERROR_IPSEC_IKE_NO_PEER_CERT	Peer failed to send valid machine certificate.
13848	0x00003618	ERROR_IPSEC_IKE_PEER_CRL_FAILED	Certification Revocation check of peer's certificate failed.
13849	0x00003619	ERROR_IPSEC_IKE_POLICY_CHANGE	New policy invalidated SAs formed with old policy.
13850	0x0000361A	ERROR_IPSEC_IKE_NO_MM_POLICY	There is no available Main Mode IKE policy.
13851	0x0000361B	ERROR_IPSEC_IKE_NOTCBPRIV	Failed to enabled TCB privilege.
13852	0x0000361C	ERROR_IPSEC_IKE_SECLOADFAIL	Failed to load SECURITY.DLL.
13853	0x0000361D	ERROR_IPSEC_IKE_FAILSSPINIT	Failed to obtain security function table dispatch address from SSPI.
13854	0x0000361E	ERROR_IPSEC_IKE_FAILQUERYSSP	Failed to query Kerberos package to obtain max token size.

Error			Description
decimal	Hexadecimal	Name	
13855	0x0000361F	ERROR_IPSEC_IKE_SRVACQFAIL	Failed to obtain Kerberos server credentials for ISAKMP/ERROR_IPSEC_IKE service. Kerberos authentication will not function. The most likely reason for this is lack of domain membership. This is normal if your computer is a member of a workgroup.
13856	0x00003620	ERROR_IPSEC_IKE_SRVQUERYCRED	Failed to determine SSPI principal name for ISAKMP/ERROR_IPSEC_IKE service (QueryCredentialsAttributes).
13857	0x00003621	ERROR_IPSEC_IKE_GETSPIFAIL	Failed to obtain new SPI for the inbound SA from Ipsec driver. The most common cause for this is that the driver does not have the correct filter. Check your policy to verify the filters.
13858	0x00003622	ERROR_IPSEC_IKE_INVALID_FILTER	Given filter is invalid.
13859	0x00003623	ERROR_IPSEC_IKE_OUT_OF_MEMORY	Memory allocation failed.
13860	0x00003624	ERROR_IPSEC_IKE_ADD_UPDATE_KEY_FAILED	Failed to add Security Association to IPSec Driver. The most common cause for this is if the IKE negotiation took too long to complete. If the problem persists, reduce the load on the faulting machine.
13861	0x00003625	ERROR_IPSEC_IKE_INVALID_POLICY	Invalid policy.
13862	0x00003626	ERROR_IPSEC_IKE_UNKNOWN_DOI	Invalid DOI.
13863	0x00003627	ERROR_IPSEC_IKE_INVALID_SITUATION	Invalid situation.
13864	0x00003628	ERROR_IPSEC_IKE_DH_FAILURE	Diffie-Hellman failure.
13865	0x00003629	ERROR_IPSEC_IKE_INVALID_GROUP	Invalid Diffie-Hellman group.
13866	0x0000362A	ERROR_IPSEC_IKE_ENCRYPT	Error encrypting payload.
13867	0x0000362B	ERROR_IPSEC_IKE_DECRYPT	Error decrypting payload.
13868	0x0000362C	ERROR_IPSEC_IKE_POLICY_MATCH	Policy match error.
13869	0x0000362D	ERROR_IPSEC_IKE_UNSUPPORTED_ID	Unsupported ID.
13870	0x0000362E	ERROR_IPSEC_IKE_INVALID_HASH	Hash verification failed.
13871	0x0000362F	ERROR_IPSEC_IKE_INVALID_HASH_ALG	Invalid hash algorithm.
13872	0x00003630	ERROR_IPSEC_IKE_INVALID_HASH_SIZE	Invalid hash size.
13873	0x00003631	ERROR_IPSEC_IKE_INVALID_ENCRYPT_ALG	Invalid encryption algorithm.
13874	0x00003632	ERROR_IPSEC_IKE_INVALID_AUTH_ALG	Invalid authentication algorithm.
13875	0x00003633	ERROR_IPSEC_IKE_INVALID_SIG	Invalid certificate signature.
13876	0x00003634	ERROR_IPSEC_IKE_LOAD_FAILED	Load failed.
13877	0x00003635	ERROR_IPSEC_IKE_RPC_DELETE	Deleted via RPC call.
13878	0x00003636	ERROR_IPSEC_IKE_BENIGN_REINIT	Temporary state created to perform reinitialization. This is not a real failure.
13879	0x00003637	ERROR_IPSEC_IKE_INVALID_RESPONDER_LIFETIME_NOTIFY	The lifetime value received in the Responder Lifetime Notify is below the Windows 2000 configured minimum value. Please fix the policy on the peer machine.
13880	0x00003638	ERROR_IPSEC_IKE_QM_LIMIT_REAP	SA reaped because QM limit was reached.
13881	0x00003639	ERROR_IPSEC_IKE_INVALID_CERT_KEYLEN	Key length in certificate is too small for configured security requirements.
13882	0x0000363A	ERROR_IPSEC_IKE_MM_LIMIT	Max number of established MM SAs to peer exceeded.
13883	0x0000363B	ERROR_IPSEC_IKE_NEGOTIATION_DISABLED	IKE received a policy that disables negotiation.
13884	0x0000363C	ERROR_IPSEC_IKE_QM_LIMIT	Reached maximum quick mode limit for the main mode. New main mode will be started.

6.8.1.60 ST_TcUlosiOutgoingConnectEventClass

The table below contains a list of the events of the ULOSI event class for the outgoing connection.

Namespace: TC_EVENTS [▶ 570].TcUlosiOutgoingConnectEventClass

Type: Global Variable

Library: Tc3_Ulosi (Tc3_Ulosi.compiled-library)

Events

Name	ID	Severity	Description
InvalidPrimitiveState	1	Error	Invalid primitive state.
InvalidAssociationObject	2	Error	Invalid association object interface (null).
InvalidSettingsObject	3	Error	Invalid settings object interface (null).
InvalidRfc1006Object	4	Error	Invalid RFC 1006 object interface (null).
InvalidClientObject	5	Error	Invalid client object interface (null).
AARETimeoutError	6	Error	Application association response (AARE) timeout error.

6.8.2 TC_EVENT_CLASSES constants

TC_EVENT_CLASSES is a global TwinCAT system variable list (GVL). Via the variable name and the variable value (GUID), each constant in this list identifies an event class or event class ID available on a TwinCAT system. The TwinCAT system has several predefined TwinCAT System event class IDs. Further event class IDs are brought along by the integrated PLC libraries and installed TwinCAT Functions or defined by the user in the PLC application. The variable values (GUIDs) are unique on a TwinCAT system and are generated and updated automatically by the TwinCAT system. The number of available event classes IDs can vary from system to system as a result.

Namespace: TwinCAT TypeSystem [▶ 101].TC_EVENT_CLASSES
Type: Global Variable List (GVL)
Library: Not required. TwinCAT System Resource from v3.1.4022.14

TwinCAT System event class IDs

Name	Type
TcSystemEventClass	GUID
TcGeneralAdsEventClass	GUID
TcRouterEventClass	GUID
TcRTimeEventClass	GUID
Win32EventClass	GUID

TF6510 event class IDs

Name	Type
TcAcsiEventClass	GUID
TcIEC61850EventClass	GUID
TcScsmEventClass	GUID
TcMmsAbortEventClass	GUID
TcMmsAssociationEventClass	GUID
TcMmsClientEventClass	GUID
TcMmsIncomingConnectEventClass	GUID
TcMmsIncomingDisconnectEventClass	GUID
TcMmsOutgoingConnectEventClass	GUID
TcMmsOutgoingDisconnectEventClass	GUID
TcMmsReceiverEventClass	GUID
TcMmsSenderEventClass	GUID
TcMmsTransactionEventClass	GUID
TcUlosiAbortEventClass	GUID
TcUlosiAssociationEventClass	GUID
TcUlosiClientEventClass	GUID
TcUlosiIncomingConnectEventClass	GUID
TcUlosiIncomingDisconnectEventClass	GUID
TcUlosiOutgoingConnectEventClass	GUID
TcUlosiOutgoingDisconnectEventClass	GUID
TcUlosiReceiverEventClass	GUID
TcUlosiSenderEventClass	GUID
TcUlosiTransactionEventClass	GUID
TcUlosiServerEventClass	GUID
TcUlosiConnectionEventClass	GUID
TcUlosiAcseAbrtDiagnosticEventClass	GUID
TcUlosiAcseAareDiagnosticEventClass	GUID
TcUlosiSpduReasonCodeEventClass	GUID
TcRfc1006AbortEventClass	GUID
TcRfc1006AssociationEventClass	GUID
TcRfc1006ClientEventClass	GUID
TcRfc1006IncomingConnectEventClass	GUID
TcRfc1006IncomingDisconnectEventClass	GUID
TcRfc1006OutgoingConnectEventClass	GUID
TcRfc1006OutgoingDisconnectEventClass	GUID
TcRfc1006ReceiverEventClass	GUID
TcRfc1006SenderEventClass	GUID
TcRfc1006TransactionEventClass	GUID
TcTpktAbortEventClass	GUID
TcTpktAssociationEventClass	GUID
TcTpktPduEventClass	GUID
TcTpktReceiverEventClass	GUID
TcTpktSenderEventClass	GUID
TcTpktTransactionEventClass	GUID

Name	Type
TcSocketsAbortEventClass	GUID
TcSocketsAssociationEventClass	GUID
TcSocketsClientEventClass	GUID
TcSocketsIncomingConnectEventClass	GUID
TcSocketsIncomingDisconnectEventClass	GUID
TcSocketsOutgoingConnectEventClass	GUID
TcSocketsOutgoingDisconnectEventClass	GUID
TcSocketsReceiverEventClass	GUID
TcSocketsSenderEventClass	GUID
TcSocketsTransactionEventClass	GUID
TcSocketsWin32SrvEventClass	GUID

6.8.3 GVL_Acsi constants

Namespace: Tc3_Acsi [▶ 101].GVL_Acsi
Type: Global Variable List (GVL)
Library: Tc3_Acsi (Tc3_Acsi.compiled-library)

Functional groups

Constant	Type	Value	Description
ACSI_FC_MX	DWORD	16#00000001	Measured values (analog)
ACSI_FC_ST	DWORD	16#00000002	Status information
ACSI_FC_CO	DWORD	16#00000004	Switch control
ACSI_FC_CF	DWORD	16#00000008	Configuration
ACSI_FC_DC	DWORD	16#00000010	Description
ACSI_FC_SP	DWORD	16#00000020	Setpoints
ACSI_FC_SG	DWORD	16#00000040	Setting group
ACSI_FC_RP	DWORD	16#00000080	Unbuffered reports
ACSI_FC_LG	DWORD	16#00000100	Logging
ACSI_FC_BR	DWORD	16#00000200	Buffered reports
ACSI_FC_GO	DWORD	16#00000400	Goose control
ACSI_FC_GS	DWORD	16#00000800	Gsse control
ACSI_FC_SV	DWORD	16#00001000	Substitute values
ACSI_FC_SE	DWORD	16#00002000	Settings group, editable
ACSI_FC_MS	DWORD	16#00004000	Multicast sampling values
ACSI_FC_SC	DWORD	16#00008000	SCL
ACSI_FC_US	DWORD	16#00010000	Unicast sampling values
ACSI_FC_EX	DWORD	16#00020000	Extended definition
ACSI_FC_XX	DWORD	16#0003FFFF	Represent all groups mentioned above

Trigger options for reports

Constant	Type	Value	Description
ACSI_TrgOp_dchg	DWORD	16#02	Report for attribute value change
ACSI_TrgOp_qchg	DWORD	16#04	Report for attribute quality change
ACSI_TrgOp_dupd	DWORD	16#08	Report for attribute value update
ACSI_TrgOp_intg	DWORD	16#10	Integrity report
ACSI_TrgOp_gi	DWORD	16#20	Report triggered by general query

Access authorization

Constant	Type	Value	Description
ACSI_AP_RO	BYTE	1	Read access
ACSI_AP_WO	BYTE	2	Write access
ACSI_AP_RW	BYTE	3	Read and write access (default)

Further constants

Constant	Type	Value	Description
NULL_EntryID	T_OCTET8	[8(0)]	All octets are 0.
NULL_Owner	T_OCTET64	[64(0)]	All octets are 0.
NULL_TimeStamp	T_UtcTime	secondSinceEpoch:=DT#1970-01-01-00:00:00, fractionOfSecond:=[0,0,0], quality:=(LeapSecondsKnown:=0, ClockFailure:=0, ClockNotSynchronized:=0, Accuracy0:=0, Accuracy1:=0, Accuracy2:=0, Accuracy3:=0, Accuracy4:=0)	Timestamp with value 0, all "quality" bits are 0 and "accuracy" is unknown.
NULL_EntryTime	T_BinaryTime	timeOfDay:=TOD#0:0:0, day:=0	Acquisition time with the value 0.
NULL_Originator	ST_AcsiOriginator	orCat:=E_AcsiOrCategory.NotSupported, orIdent:=[64(0)]	Value, "orCat" is 0 and all octets in "orIdent" are 0.
NULL_ReasonCode	ST_AcsiReasonCode	reserved:=0, DataChange:=0, QualityChange:=0, DataUpdate:=0, Integrity:=0, GeneralInterrogation:=0, ApplicationTrigger:=0, unused:=0	All bits are 0.
NULL_TrgOps	ST_AcsiTriggerConditions	reserved:=0, DataChange:=0, QualityChange:=0, DataUpdate:=0, Integrity:=0, GeneralInterrogation:=0, reserved2:=0	All bits are 0.
NULL_OptFlds	ST_AcsiOptionalFields	reserved:=0, SequenceNumber:=0, ReportTimeStamp:=0, ReasonForInclusion:=0, DataSetName:=0, Datareference:=0, BufferOverflow:=0, EntryID:=0, ConfRevision:=0, Segmentation:=0	All bits are 0.
NULL_LogOptFlds	ST_AcsiLogOptionalFields	ReasonForInclusion:=0	All bits are 0.
NULL_Quality	ST_AcsiQuality	Validity0:=0, Validity1:=0, Overflow:=0, OutOfRange:=0, BadReference:=0, Oscillatory:=0, Failure:=0, OldData:=0, Inconsistent:=0, Inaccurate:=0, Source:=0, Test:=0, OperatorBlocked:=0	All bits are 0.
NULL_Unit	ST_AcsiUnit	SIUnit:=E_AcsiSIUnit.None, multiplier:=E_AcsiMultiplier.None	Value, "SIUnit" is dimensionless and "multiplier" is 0.
NULL_ValWithTrans	ST_AcsiValWithTrans	posVal:=0, transInd:=FALSE	Value, "posVal" is 0 and "transInd" is FALSE.
NULL_AnalogueValue	ST_AcsiAnalogueValue	i:=0, f:=REAL#0.0	Value, "i" is 0 and "f" is 0.
NULL_UINT24	T_UINT24	[0,0,0]	Unsigned 24-bit number with the value 0.
NULL_INT24	T_INT24	[0,0,0]	Signed 24-bit number with the value 0.
NULL_UINT128	T_UINT128	Hi:=0, Lo:=0	Unsigned 128-bit number with the value 0.

Constant	Type	Value	Description
NULL_INT128	T_INT128	Hi:=0, Lo:=0	Signed 128-bit number with the value 0.

7 Examples

Some samples of the use of the IEC 61850 implementation in TwinCAT are explained in this chapter. As all samples are based on a general structure, this is explained first and is a prerequisite in the subsequent samples. Following that, each sample has its own chapter in which the special functions of the project are listed.

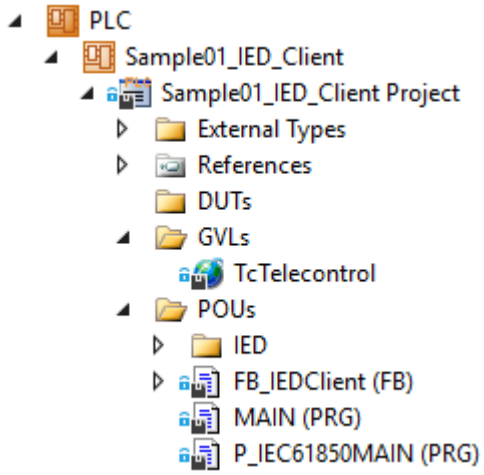
The sample code is included in the documentation as zip files. After unpacking the samples, it is advisable to retain the project folder structure. Each sample is accompanied by an ICD file. If the ICD file is imported from the original folder in the TwinCAT Telecontrol Configurator and a new, modified TwinCAT project is generated, then a TwinCAT project comparison can be made with the original project. TwinCAT Telecontrol Configurator will output a corresponding message during project code generation and suggest project comparison. The prerequisite for this is that a project with the same name has been found in a subdirectory (below the ICD file).
The sample projects use the default solution/project names of the TwinCAT Telecontrol Configurator. In order to generate a project that can be compared with TwinCAT Project Compare, the project type: server must be selected for the server samples and the project type: client for the client samples during code generation.

If the ICD file has been copied to a different location or a different project or solution name has been assigned, then no automatic TwinCAT project comparison can be performed after code generation. TwinCAT Telecontrol Configurator generates a new TwinCAT project at the new location or under a different name. If this is the case, you can start TwinCAT Project Compare as a standalone tool (**Start > Beckhoff > TwinCAT Project Compare**), navigate to the two projects there and perform a comparison.

7.1 General Client - Project structure

Each of the client sample projects follows a general structure, which this document aims to explain. This structure includes the nesting of the function blocks in the Solution Explorer and the structure of the state machine in the client function block, which establishes the connection to the server and implements the data exchange. The basic structure of the TwinCAT IEC 61850 sample projects is based on the PLC project structure automatically generated by the TwinCAT Telecontrol Configurator. The TwinCAT Telecontrol Configurator in turn uses the "TwinCAT XAE Project (XML format)" as a template when generating the solution. The data models communicated in the samples reflect different IEC 61850 servers. They differ from sample to sample and are enclosed with the source code as icd files. The ICD files can also be used by third-party software for the simulation of a server.

First of all, a brief examination of the structure of a TwinCAT IEC 61850 PLC client project:



The generated solution name (unless specified otherwise) corresponds to the TwinCAT Telecontrol Configurator project name. The automatically generated TwinCAT PLC project name (unless specified otherwise) on the other hand has the following structure: "[Project name]_[IEDName]_Client".

As standard, every sample project has a "DUTs", a "GVLs" and a "POUs" folder. A Global Variable List (GVL) with the name "TcTelecontrol" is stored in the "GVLs" folder. The following function blocks are instantiated and initialized in this Global Variable List (see source code below):

- A Client function block instance of the type: FB_[IEDName]Client (connection management and data exchange).
- An IED data model function block instance of the type: FB_IED_[IEDName].
- Optional (depending on the GOOSE Subscriber configuration): one or more function block instances of the type: "FB_[IEDName]Gse" for GOOSE communication and GSE management.

In addition, the code version used during code generation and the version of the TwinCAT Telecontrol Configurator used are also stored there.

Namespace: TcTelecontrol

Type: Global Variable List (GVL)

```
VAR_GLOBAL
    ipCreator         : I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=2).SetGui
Ver(major:=1, minor:=0, build:=93, revision:=10);
    fb[IEDName]       : FB_IED_[IEDName];
    fb[IEDName]Client : FB_[IEDName]Client := (fbConnection:=(ipIED:=fb[IEDName],settings:=(sRemoteH
ost:='127.0.0.1')));
    fb[IEDName]Gse    : FB_[IEDName]Gse := (fbAdapter:=(ipIED:=fb[IEDName], settings:=(sMulticastAdd
r:='01-0C-CD-01-00-00', eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
END_VAR
```

In the "POUs" folder there is a further folder: "[IEDName]", which contains the entire hierarchical structure of the IED data model as function blocks. This folder also contains the IED function block that is instanced in the global variable list and has already been mentioned. The Client function block: "FB_[IEDName]Client", which implements the establishment of the connection and the data exchange with an IEC 61850 Server, is located on the same level. The TwinCAT PLC project contains a "MAIN" program as standard. This is called cyclically by a TwinCAT task and in turn calls the program "P_IEC61850MAIN". The program "P_IEC61850MAIN" encapsulates the call of the client function block and of the optional GSE function block, separates the IEC 61850 communication from the remainder of the PLC machine program and helps, for example, with the implementation of further clients.

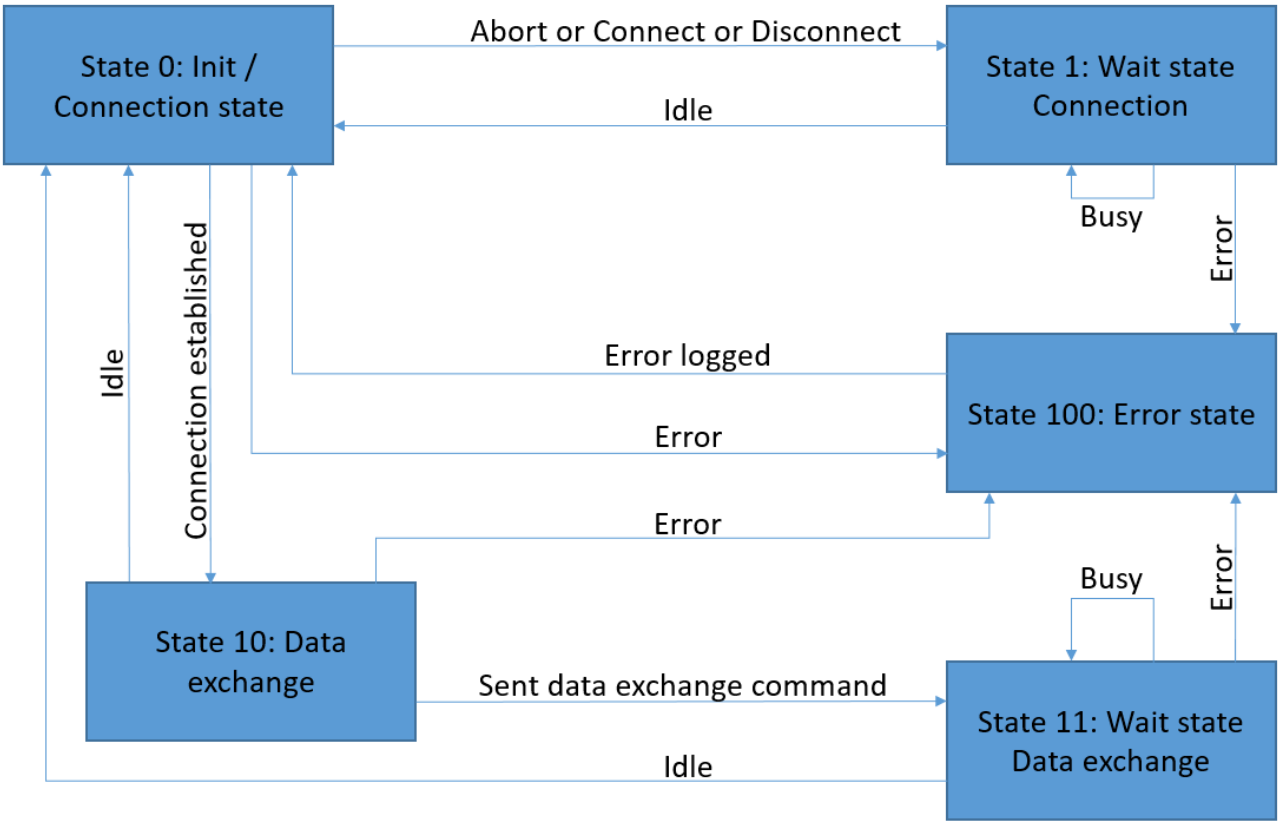
```
PROGRAM MAIN
VAR
END_VAR
```

```
P_IEC61850MAIN();
```

```
PROGRAM P_IEC61850MAIN
VAR
END_VAR
```

```
fb[IEDName]Client();
fb[IEDName]Gse();
```

In the FB_[IEDName]Client function block, there is a state machine, the basic states of which are used in every client sample. These states are graphically illustrated in the following diagram:



State 0 (Init state): the state machine is in this state as soon as the PLC program has been started. Commands for the management of the client/server connection are handled here (and in State 1). This is mainly controlled via four Boolean variables. When set, these variables then activate the corresponding commands (in this case these are once-only method calls at the client function block).

- _bAbort: calls the method "AbortReq", which activates the command to abort the client connection to the server.
- _bConnect: calls the method "AssociateReq", which activates the command to establish a new client connection to the server.
- _bDisconnect: calls the method "ReleaseReq", which activates the command for the controlled release of an existing client connection to the server.
- _bReconnect: also calls the method "AssociateReq" if the Client connection to the Server has been aborted/disconnected, but is to be restored automatically.

The methods listed above, which are called once only in this state, require longer than one PLC cycle for their execution. For this reason the state machine switches to a wait state (State 1), in which the termination of the activated command is awaited.

If the client connection to the server has already been established beforehand, the state machine will switch to the data transmission state (State 10).

State 1 (Wait State): in this state, the client waits until the command processing for the management of the client/server connection is no longer busy. As long as the connection is established, terminated or aborted, the state machine is in state 1. As soon as the command has been successfully processed, the state machine is returned to state 0 (Init state).

State 10 (Data exchange): if the state machine is in this state, then the client connection to the server has already been successfully established. The client is ready for the data transmission to the server. During the data transmission, commands are activated for the transmission or reception of the data.

In this state the sample projects differ from one another. Different methods or auxiliary function blocks are called here, depending on the desired functionality or logic in the application. In addition, the client function block can be extended by further states.

All method calls that activate commands for data transmission require several PLC cycles for successful execution, therefore the state machine must be set to State 11 (Wait state) after calling such a method.

In the case of an active connection and no data transmission, the state machine switches between States 0 and 10. The state machine is reset to State 0 in order to react to changes in the client-server connection status and to handle them in State 0.

State 11 (Wait state): this state is a further wait state. As soon as a data transmission command (activated in State 10) has been executed, the state machine is set to State 11 and waits until the command execution is no longer busy. The state machine then switches to State 0.

State 100 (Error state): as soon as an error occurs during the activation or processing of a command, the state machine is set to State 100. The error is logged here and the state machine reset to State 0.

```
FUNCTION_BLOCK FB_[IEDName]Client
VAR_INPUT
    fbConnection              : FB_iec61850ClientClass;
END_VAR
VAR
    _bAbort                   : BOOL;
    _bDisconnect              : BOOL;
    _bConnect                 : BOOL;
    _bReconnect               : BOOL := TRUE;
    _bReadAllData             : BOOL := TRUE;
    state                     : BYTE;
    eState                    : E_AsyncEnvironmentState;
    bBusy                     : BOOL;
    bSuccess                  : BOOL;
    ipResult                  : I_AsyncServiceResultClass;
    sLastErrorResult          : T_MaxString;
    fbAbortReason             : FB_ServiceErrorClass := (stError:=SUCCESS_EVENT);
    sLastAbortReason          : T_MaxString;
    nInvokeID                 : UDINT;
    eServiceError             : E_AcsiServiceError;
    nServiceError             : UDINT;
    nCmdError                 : UDINT;
    sObjReference             : T_AcsiObjectReference;
    sCtrlReference            : T_AcsiObjectReference;

    bGetServerDirectory       : BOOL := TRUE;
    bGetLogicalDeviceDirectory: BOOL := TRUE;
    bGetLogicalNodeDirectory  : BOOL := TRUE;
    bGetAllServerValues       : BOOL := TRUE;
    bGetAllServerValues       : BOOL := TRUE;
    bGetAllDataValues_LLN0_ST : BOOL := TRUE;
    bGetDataValues_LLN0_ST_Beh: BOOL := TRUE;
END_VAR
```

```
fbConnection.Execute();
eState:= fbConnection.eState;

CASE state OF
    0:
        IF _bAbort THEN
            _bAbort:= FALSE;
            bSuccess:= fbConnection.AbortReq(ipReason:=fbAbortReason, ipSink:=0, ipResult=>ipResult)
;
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Idle AND (_bConnect OR _bReconnect) THEN
            _bConnect:= FALSE;
            bGetAllServerValues:= SEL(_bReadAllData, bGetAllServerValues, TRUE);
            bSuccess:= fbConnection.AssociateReq(ipSink:=0, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Established AND _bDisconnect THEN
            _bDisconnect:= FALSE;
            bSuccess:= fbConnection.ReleaseReq(ipSink:=0, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Established THEN
            state:= 10;
        END_IF
        _bConnect:= FALSE;
        _bDisconnect:= FALSE;
    1:
        IF ipResult <> 0 THEN
            ipResult.Execute();
            IF NOT (bBusy:=ipResult.IsBusy()) THEN
                state:= SEL(ipResult.IsCompleted(), 100, 0);
            END_IF
        END_IF
    10:
        IF bGetServerDirectory THEN
            bGetServerDirectory:= FALSE;
```

```
            bSuccess:= fbConnection.GetServerDirectoryReq(ipServer:=fb[IEDName], eClass:=E_AcsiServe
rDirectoryClass.LogicalDevice, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetLogicalDeviceDirectory THEN
            bGetLogicalDeviceDirectory:= FALSE;
            bSuccess:= fbConnection.GetLogicalDeviceDirectoryReq(ipLogicalDevice:=fb[IEDName].IEDLD1
, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetLogicalNodeDirectory THEN
            bGetLogicalNodeDirectory:= FALSE;
            bSuccess:= fbConnection.GetLogicalNodeDirectoryReq(ipLogicalNode:=fb[IEDName].IEDLD1.LLN
0, eClass:=E_AcsiLogicalNodeClass.DataSet, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipRe
sult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetAllServerValues THEN
            bGetAllServerValues:= FALSE;
            bSuccess:= fbConnection.GetAllServerValuesReq(ipServer:=fb[IEDName], hUser:=0, ipSink:=0
, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetAllDataValues_LLN0_ST THEN
            bGetAllDataValues_LLN0_ST:= FALSE;
            bSuccess:= fbConnection.GetAllDataValuesReq(ipLogicalNode:=fb[IEDName].IEDLD1.LLN0, eFc:
=E_AcsiFc.ST_, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSIF bGetDataValues_LLN0_ST_Beh THEN
            bGetDataValues_LLN0_ST_Beh:= FALSE;
            bSuccess:= fbConnection.GetDataValuesReq(ipData:=fb[IEDName].IEDLD1.LLN0.Beh, eFc:=E_Acs
iFc.ST_, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 11);
        ELSE
            state:= 0;
        END_IF
    11:
        IF ipResult <> 0 THEN
            ipResult.Execute();
            IF NOT (bBusy:=ipResult.IsBusy()) THEN
                state:= SEL(ipResult.IsCompleted(), 100, 0);
            END_IF
        END_IF
    100:
        state:= 0;
        IF ipResult <> 0 THEN
            nCmdError:= nCmdError + 1;
            sLastErrorResult:= ipResult.Dump();
        END_IF
END_CASE
```

GOOSE Subscriber (optional)

TwinCAT Telecontrol Configurator can also generate the PLC code for a GOOSE subscriber in a client project during PLC code generation (see code sample below). This requires that the user has previously created the GOOSE components such as GoCBs (goose control blocks) in the TwinCAT Telecontrol Configurator or imported them from an SCL file (e.g. ICD file).
By default, a function block with the name: "FB_[IEDName]Gse" is instantiated during code generation and added to the Global Variable List "TcTelecontrol". This function block establishes the connection between a network adapter of the TwinCAT control computer, the IED data model and the GOOSE configuration in the GoCBs. The GoCBs are instantiated in the IED data model (usually in LLN0). Each GoCB has a function block subelement with the name: "Subscriber". The subscription can be enabled or disabled from the PLC code via the "Subscriber" method calls. By default, the subscription is enabled for all GoCBs when the PLC program is started. This is controlled by the "bSubscriber" variable initialized with "TRUE". A rising edge at the "bUnsubscribe" variable can be used to disable the subscription for all GoCBs. Subscriber commands issued through these methods are executed immediately, without wait cycles or further states required to complete command processing.
The "Subscriber" described here reads the configuration and updates the status of the GoCB (attribute "GoEna" is set to "TRUE" or "FALSE", for example), but it does not use the client-server services such as "SetGoCBValues" or "GetGoCBValues" to enable or disable the "Publisher" on the server side. This means that the generated code already implements a subscriber that can be enabled or disabled, for example, in the first PLC cycle or from the PLC code at any time. The required GoCB configuration settings (GoCB attribute values) can be made via initialization values. However, the GoCBs can already be configured in the TwinCAT Telecontrol Configurator. The initialization values are automatically generated and assigned during code generation. If the Subscriber has been activated and the configuration of the GoCB and the network

adapter matches the receiving GOOSE frame, then the GOOSE data is copied (mapped) into the TwinCAT IED data model. The "Execute" method must be called cyclically the rest of the time. Among other things, it is responsible for updating the status information in the GoCB.

```
FUNCTION_BLOCK FB_[IEDName]Gse IMPLEMENTS I_GseLinkStatusEventSink
VAR_INPUT
    fbAdapter    : FB_GseAdapterClass := (ipLinkStatus:=THIS^);
END_VAR
VAR
    eLinkStatus  : E_GseLinkStatus;
    bSuccess     : BOOL;
    ipError      : I_ServiceErrorClass;
    bSubscribe   : BOOL := TRUE;
    bUnsubscribe : BOOL;
END_VAR
```

```
bSuccess:= fbAdapter.Execute(ipError=>ipError);
IF bSubscribe THEN
    bSubscribe:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Subscriber.Enable(ipAdapter:=fbAdapter, ipError=>ipErr
or);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Subscriber.Enable(ipAdapter:=fbAdapter, ipError=>ipErr
or);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Subscriber.Enable(ipAdapter:=fbAdapter, ipError=>ipErr
or);
ELSIF bUnsubscribe THEN
    bUnsubscribe:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Subscriber.Disable(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Subscriber.Disable(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Subscriber.Disable(ipError=>ipError);
ELSE
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Subscriber.Execute(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Subscriber.Execute(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Subscriber.Execute(ipError=>ipError);
END_IF
```

In the project tree under the I/O-Device branch you will find a network adapter instance named "GSE (RT Ethernet adapter)". This adapter instance must be configured accordingly, i.e. the I/O configuration must be adapted to the existing hardware and to the target platform on which the project is to run.
A new I/O configuration is also necessary if you change the target platform. This configuration must be done manually in TwinCAT XAE. In addition to the I/O configuration of the network adapter, a link must be established between the network adapter and the PLC function blocks for Goose communication. The link can be used to forward the data received from the network adapter to the instance of the function block: "FB_[IEDName]Gse". In the opposite direction the instance of the function block "FB_[IEDName]Gse" can forward the data to be sent to the network adapter.

Here you can find more information: RT Ethernet adapter Configuration [▶ 35].

7.2 Client - Base Sample Project

This example shows the fundamental TwinCAT implementation of the communication of a TwinCAT IEC 61850 Client and the use of some methods to read data from a Server. The GOOSE components and reporting are not included in this example in the interests of simplicity. These and other functionalities are covered in separate examples. The Client implemented here attempts to establish a connection to the Server with the host address: "127.0.0.1" and port number: 102 after the PLC program start. These values can be adapted in the PLC sample code. The PLC development environment will output a warning with a note about this when translating. Double-clicking on this message will take you to the corresponding location in the PLC code.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/5494779659.zip

The example described here uses the state machine that is described in the "General Client project structure [▶ 664]" chapter. The States 0, 1, 11 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

In State 10, several IF instructions are listed that activate commands (method calls) of the Client block. The command is activated and the corresponding Client method called via a rising edge at one of the Boolean variables in the IF statement. For test purposes, the user can set the values of the Boolean variables to "TRUE" in the online view and in this way initiate the command processing.

Following the successful processing of a command, the state machine is set to State 0. If several of the Boolean variables have been set to "TRUE", the uppermost IF statement will be processed first with the respective command. The more frequent and complex the commands sent to the Server, the longer the Client will take to process them.

Some of the Boolean variables are listed in the following table. In addition, they contain the method names and a description of the function.

Commands in Sample01

Variable name	Method name	Description
bGetServerDirectory	GetServerDirectoryReq	Returns a list of all the Logical Devices of an IED (Intelligent Electronic Device).
bGetLogicalDeviceDirectory	GetLogicalDeviceDirectoryReq	Returns a list of all Logical Nodes within a Logical Device.
bGetLogicalNodeDirectory	GetLogicalNodeDirectoryReq	Returns a list of all instances of a certain Logical Node Class. In our example, it is the class: "E_AcsiLogicalNodeClass.DataSet ".
bGetAllServerValues	GetAllServerValuesReq	Reads all data values of the specified IED.
bGetAllDataValues_LLN0_ST	GetAllDataValuesReq	Reads all data values of a specified Logical Node and of the specified Functional Constraint.
bGetDataValues_LLN0_ST_Beh	GetDataValuesReq	Reads all data values of a specified Data Object and of the specified Functional Constraint.

The naming of the Boolean variables is only an example and can be adapted as desired. It is only intended to show which values can be read by connecting the variables.

7.3 Client - Read Data (GetAllDataValues, GetDataValues)

This sample shows how to use the "GetAllDataValuesReq" and "GetDataValuesReq" methods of the client function block.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/5494782347.zip

The example described here uses the state machine that is described in the "General Client project structure [▶ 664]" chapter. The States 0, 1, 11 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

In State 10, several IF instructions are listed that activate commands (method calls) of the Client block. The command is activated and the corresponding Client method called via a rising edge at one of the Boolean variables in the IF statement. For test purposes, the user can set the values of the Boolean variables to "TRUE" in the online view and in this way initiate the command processing.

Following the successful processing of a command, the state machine is set to State 0. If several of the Boolean variables have been set to "TRUE", the uppermost IF statement will be processed first with the respective command. The more frequent and complex the commands sent to the Server, the longer the Client will take to process them.

Some of the Boolean variables are listed in the following table. In addition, they contain the method names and a description of the function.

Commands in Sample02

Variable name	Method name	Description
bGetAllDataValues_LLN0 similar variables (commands): bGetAllDataValues_LPHD1 bGetAllDataValues_MMXU1 bGetAllDataValues_XCBR1	GetAllDataValuesReq	As soon as this Boolean variable is set to "TRUE", all data values of all sub-elements of the Logical Node: „LLN0“ and every functional group are read. To enable this, the method call uses an interface pointer to the Logical Node to be read: "LLN0" as a parameter and the specification of the functional group: "E_AcsiFc.XX."
bGetAllDataValues_LLN0_ST similar variables (commands): bGetAllDataValues_LLN0_CF bGetAllDataValues_LLN0_DC bGetAllDataValues_LLN0_EX bGetAllDataValues_LPHD1_ST bGetAllDataValues_LPHD1_DC bGetAllDataValues_MMXU1_MX bGetAllDataValues_MMXU1_ST bGetAllDataValues_XCBR1_ST bGetAllDataValues_XCBR1_CO bGetAllDataValues_XCBR1_CF	GetAllDataValuesReq	As soon as this Boolean variable is set to "TRUE", all data values of all sub-elements of the Logical Node: „LLN0“ and of the "ST" functional group are read. To enable this, the method call uses an interface pointer to the Logical Node to be read: "LLN0" as a parameter and the specification of the functional group: "E_AcsiFc.ST_".
bGetDataValues_LLN0_ST_Beh similar variables (commands): bGetDataValues_LLN0_ST_Health bGetDataValues_LLN0_ST_Mod bGetDataValues_LLN0_CF_Mod bGetDataValues_LLN0_DC_NamPlt bGetDataValues_LLN0_EX_NamPlt bGetDataValues_LPHD1_ST_PhyHealth bGetDataValues_LPHD1_ST_Proxy bGetDataValues_LPHD1_DC_PhyNam bGetDataValues_MMXU1_MX_PhV bGetDataValues_MMXU1_ST_Beh bGetDataValues_XCBR1_ST_Beh	GetDataValuesReq	As soon as this Boolean variable is set to "TRUE", all data values of all sub-elements of the data object: "Beh" of the Logical Node: „LLN0“ and of the "ST" functional group are read. To enable this, the method call uses an interface pointer to the data object to be read: "LLN0.Beh" as a parameter and the specification of the functional group: "E_AcsiFc.ST_".
bGetDataValues_LLN0_ST_Beh_stVal similar variables (commands): bGetDataValues_LLN0_ST_Beh_q bGetDataValues_LLN0_ST_Beh_t bGetDataValues_LLN0_ST_Health_q bGetDataValues_LLN0_ST_Health_t bGetDataValues_LLN0_ST_Mod_q bGetDataValues_LLN0_ST_Mod_t bGetDataValues_LLN0_CF_Mod_ctlModel bGetDataValues_LLN0_DC_NamPlt_vendor bGetDataValues_LLN0_DC_NamPlt_swRev	GetDataValuesReq	As soon as this Boolean variable is set to "TRUE", the value of the data attribute is read: "stVal" of the data object: "Beh", of the Logical Node: "LLN0", and of the functional group: "ST". To enable this, the method call uses an interface pointer to the data attribute to be read: "LLN0.Beh.stVal" as a parameter and the specification of the functional group: "E_AcsiFc.ST_".

The naming of the Boolean variables is only an example and can be adapted as desired. It is only intended to show which values can be read by connecting the variables.

7.4 Client - Write Data (SetAllDataValues, SetDataValues)

This sample shows the use of the "SetAllDataValuesReq" and "SetDataValuesReq" methods of the client function block.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/8922555275.zip

The example described here uses the state machine that is described in the "General Client project structure [▶ 664]" chapter. The States 0, 1, 11 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

Sample project

In the FB_IEDClient function block and State 10, several IF instructions are listed that activate commands (method calls) of the client function block. Each command in the IF statement is activated by a rising edge at one of the Boolean variables. The Boolean variables are defined in the declaration part of the "FB_IEDClient" function block. For test purposes, the user can set the values of the Boolean variables to "TRUE" in the online view and in this way activate the command execution. After that, the system switches to state 11, where the method "ipResult.Execute()" is called in further PLC cycles until the command processing has been completed. This is the case when "ipResult.IsBusy()" returns the value "FALSE".

Following the successful execution of a command, the state machine is set to State 0. If several of the Boolean variables are set to "TRUE", then the topmost IF statement is processed first with the respective command. Some of the Boolean variables are listed in the tables below. In addition, they contain the method names and a description of the function.

Variable name	Method name	Description
bSetAllDataValues_LPHD1_DC similar variables (commands): bSetAllDataValues_LLN0_DC bSetAllDataValues_MMXU1_DC bSetAllDataValues_XCBR1_DC bSetAllDataValues_CSWI1_DC bSetAllDataValues_LEDGGIO1_DC bSetAllDataValues_LEDGGIO2_DC bSetAllDataValues_LEDGGIO3_DC bSetAllDataValues_LEDGGIO4_DC bSetAllDataValues_LEDGGIO5_DC bSetAllDataValues_LEDGGIO6_DC bSetAllDataValues_LEDGGIO7_DC bSetAllDataValues_LEDGGIO8_DC	SetAllDataValuesReq	As soon as this Boolean variable is set to "TRUE", all data values of all subelements of the logical node: "LPHD1" and the functional group "DC" are written. To enable this, the method call uses as parameter an interface pointer to the logical node to be written: "LPHD1" and the specification of the functional group: "E_AcsiFc.DC".
bSetDataValues_MMXU1_CF_TotW similar variables (commands): bSetDataValues_LEDGGIO1_CF_AnOut1_sboClass bSetDataValues_LEDGGIO1_CF_AnOut1_ctlModel bSetDataValues_LEDGGIO1_CF_AnOut1_units bSetDataValues_LEDGGIO1_CF_AnOut1_units_SIUnit bSetDataValues_LEDGGIO1_CF_AnOut1_units_multiplier	SetDataValuesReq	As soon as this Boolean variable is set to "TRUE", all data values of all subelements of the data object: "TotW" of the logical node "MMXU1" and the functional group: "CF" are written. To enable this, the method call uses as parameter an interface pointer to the data object to be written: "MMXU1.TotW" and the specification of the functional group: "E_AcsiFc.CF".

The naming of the Boolean variables is only an example and can be adapted as desired. It is only intended to show which values can be written by connecting the variables.

7.5 Client - Report Control Blocks (Unbuffered, Buffered)

This sample shows how to use the report control block instances for buffered and unbuffered reports (Unbuffered/Buffered Report Control Blocks: URCB, BRCB). This includes activating and deactivating reports or activating the general interrogation (GI).

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/5494785035.zip

The example described here uses the state machine that is described in the "General Client project structure [▶ 664]" chapter. The States 0, 1, 11 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

The Report Control Blocks (RCBs) configure and control the sending of buffered or unbuffered report messages. Reports are configured and controlled via read/write accesses to the corresponding attribute values of a Report Control Block. For example, some attribute values configure the content of the information contained in the report, while other attribute values enable or disable the sending of reports. So it is theoretically possible to configure and control a RCB only with the help of the client methods: "GetDataValuesReq" and "SetDataValuesReq". This requires a more detailed knowledge of how the RCBs work since not every attribute may be accessed at all times. Some attribute values only allow read access, while others also allow write access. Some attribute values allow write access only when the RCB is in a certain state. The general interrogation (GI) commands may only be executed when the RCB is in the enabled state. To simplify handling, the TwinCAT IEC 61850 implementation offers even more options for configuring and controlling the RCBs. All available options (methods) are briefly described below.

Basically the RCB attributes are initialized with initial values that have been preconfigured in TwinCAT Telecontrol Configurator for TrgOps, OptFlds, IntgPd, RptID, DatSet etc. after downloading the PLC program (PLC not yet running or no boot project active). If the configuration originates from an ICD file, then TwinCAT Telecontrol Configurator takes the initial values from the ICD file. The client projects are designed in such a way that after starting the PLC they first establish a connection to the server and then read all server data into the client. After that, the client side has the data values that are also present on the server. If the server data differs from the initial values on the client side, then it will be overwritten with the server data. This includes the attributes of the RCBs. The reading of all values is controlled in the client projects via two Boolean variables: "bGetAllServerValues" and "_bReadAllData". If "_bReadAllData" is TRUE then all server data are always read once after the connection is established or restored. Via a rising edge at "bGetAllServerValues" the values can also be read in again at any time. If both Boolean variables are set to FALSE, the data will not be matched. This depends on the application.
Basically, it makes sense to compare the client data with the server after the connection is established. However, the initial values configured in the TwinCAT Telecontrol Configurator are not lost. The application can still access these values through the Config properties of the data attributes. Each data attribute has a property: eConfig or iConfig or nConfig, bConfig (depending on the base type of the data attribute) etc. with the original initial value. In this sample project, among other things, initial values configured in the TwinCAT Telecontrol Configurator are also used when configuring and enabling the RCBs.

The RCB instances used in this sample attach to the Logical Node instance: LLN0. The configuration values exported by the TwinCAT Telecontrol Configurator for report trigger options (TrgOps) or optional report data fields (OptFlds) are also located there, for example.

```
FUNCTION_BLOCK FB_LN_IED_LD1_LLN0 EXTENDS FB_AcsiCommonLogicalNodeClass
VAR_INPUT
    NamPlt: FB_DO_IED_LD1_LLN0_NamPlt := (sClass:='LPL', bLinkResult:=THIS^.AddDataToContainer(ipDat
a:=NamPlt));
    Beh: FB_DO_IED_LD1_LLN0_Beh := (sClass:='ENS', bLinkResult:=THIS^.AddDataToContainer(ipData:=Beh
));
    Health: FB_DO_IED_LD1_LLN0_Health := (sClass:='ENS', bLinkResult:=THIS^.AddDataToContainer(ipDat
a:=Health));
    Mod_: FB_DO_IED_LD1_LLN0_Mod := (sObjectName:='Mod', sClass:='ENC', bLinkResult:=THIS^.AddDataTo
Container(ipData:=Mod_));

    DS1: FB_DS_IED_LD1_LLN0_DS1 := (bLinkResult:=THIS^.AddDataSetToContainer(ipDataSet:=DS1));
    DS2: FB_DS_IED_LD1_LLN0_DS2 := (bLinkResult:=THIS^.AddDataSetToContainer(ipDataSet:=DS2));
    DS3: FB_DS_IED_LD1_LLN0_DS3 := (bLinkResult:=THIS^.AddDataSetToContainer(ipDataSet:=DS3));

    urcb101: FB_ScsmUrCBImplClass := (RptID:=(sValue:='IEDLD1/LLN0.urcb101'),
                                      DatSet:=(sValue:='IEDLD1/LLN0.DS1'),
                                      ConfRev:=(nValue:=1),
                                      OptFlds:=(SequenceNumber:=TRUE, ReportTimeStamp:=TRUE, ReasonFor
Inclusion:=TRUE, DataSetName:=TRUE, DataReference:=TRUE, BufferOverflow:=FALSE, EntryID:=FALSE, Conf
Revision:=TRUE),
                                      TrgOps:=(DataChange:=TRUE, QualityChange:=TRUE, DataUpdate:=FALS
E, Integrity:=TRUE, GeneralInterrogation:=TRUE),
                                      IntgPd:=(nValue:=2000),
                                      bLinkResult:=THIS^.AddUnbufferedReportControlBlockToContainer(ip
UnbufferedReportControlBlock:=urcb101));
    urcb201: FB_ScsmUrCBImplClass := (RptID:=(sValue:='IEDLD1/LLN0.urcb201'),
                                      DatSet:=(sValue:='IEDLD1/LLN0.DS2'),
```

```
                                    ConfRev:=(nValue:=1),
                                    OptFlds:=(SequenceNumber:=TRUE, ReportTimeStamp:=TRUE, ReasonFor
Inclusion:=TRUE, DataSetName:=TRUE, DataReference:=FALSE, BufferOverflow:=FALSE, EntryID:=FALSE, Con
fRevision:=TRUE),
                                    TrgOps:=(DataChange:=TRUE, QualityChange:=TRUE, DataUpdate:=TRUE
, Integrity:=TRUE, GeneralInterrogation:=TRUE),
                                    IntgPd:=(nValue:=5000),
                                    bLinkResult:=THIS^.AddUnbufferedReportControlBlockToContainer(ip
UnbufferedReportControlBlock:=urcb201));
    urcb301: FB_ScsmUrCBImplClass := (RptID:=(sValue:='IEDLD1/LLN0.urcb301'),
                                    DatSet:=(sValue:='IEDLD1/LLN0.DS3'),
                                    ConfRev:=(nValue:=1), OptFlds:=(SequenceNumber:=TRUE, ReportTime
Stamp:=TRUE, ReasonForInclusion:=TRUE, DataSetName:=TRUE, DataReference:=FALSE, BufferOverflow:=FALS
E, EntryID:=FALSE, ConfRevision:=TRUE),
                                    TrgOps:=(DataChange:=TRUE, QualityChange:=TRUE, DataUpdate:=TRUE
, Integrity:=TRUE, GeneralInterrogation:=TRUE),
                                    IntgPd:=(nValue:=5000),
                                    bLinkResult:=THIS^.AddUnbufferedReportControlBlockToContainer(ip
UnbufferedReportControlBlock:=urcb301));
    brcb101: FB_ScsmBrCBImplClass := (RptID:=(sValue:='IEDLD1/LLN0.brcb101'),
                                    DatSet:=(sValue:='IEDLD1/LLN0.DS1'),
                                    ConfRev:=(nValue:=1),
                                    OptFlds:=(SequenceNumber:=TRUE, ReportTimeStamp:=TRUE, ReasonFor
Inclusion:=TRUE, DataSetName:=TRUE, DataReference:=TRUE, BufferOverflow:=TRUE, EntryID:=TRUE, ConfRe
vision:=TRUE),
                                    TrgOps:=(DataChange:=TRUE, QualityChange:=TRUE, DataUpdate:=FALS
E, Integrity:=TRUE, GeneralInterrogation:=TRUE),
                                    IntgPd:=(nValue:=2000),
                                    bLinkResult:=THIS^.AddBufferedReportControlBlockToContainer(ipBu
fferedReportControlBlock:=brcb101));
    brcb201: FB_ScsmBrCBImplClass := (RptID:=(sValue:='IEDLD1/LLN0.brcb201'),
                                    DatSet:=(sValue:='IEDLD1/LLN0.DS2'),
                                    ConfRev:=(nValue:=1), OptFlds:=(SequenceNumber:=TRUE, ReportTime
Stamp:=TRUE, ReasonForInclusion:=TRUE, DataSetName:=TRUE, DataReference:=FALSE, BufferOverflow:=TRUE
, EntryID:=TRUE, ConfRevision:=TRUE),
                                    TrgOps:=(DataChange:=TRUE, QualityChange:=TRUE, DataUpdate:=TRUE
, Integrity:=TRUE, GeneralInterrogation:=TRUE),
                                    IntgPd:=(nValue:=5000),
                                    bLinkResult:=THIS^.AddBufferedReportControlBlockToContainer(ipBu
fferedReportControlBlock:=brcb201));
    brcb301: FB_ScsmBrCBImplClass := (RptID:=(sValue:='IEDLD1/LLN0.brcb301'),
                                    DatSet:=(sValue:='IEDLD1/LLN0.DS3'),
                                    ConfRev:=(nValue:=1),
                                    OptFlds:=(SequenceNumber:=TRUE, ReportTimeStamp:=TRUE, ReasonFor
Inclusion:=TRUE, DataSetName:=TRUE, DataReference:=FALSE, BufferOverflow:=TRUE, EntryID:=TRUE, ConfR
evision:=TRUE),
                                    TrgOps:=(DataChange:=TRUE, QualityChange:=TRUE, DataUpdate:=TRUE
, Integrity:=TRUE, GeneralInterrogation:=TRUE),
                                    IntgPd:=(nValue:=5000),
                                    bLinkResult:=THIS^.AddBufferedReportControlBlockToContainer(ipBu
fferedReportControlBlock:=brcb301));
END_VAR
VAR
END_VAR
```

Report configuration/control with the methods: GetDataValuesReq, SetDataValuesReq

These are the basic methods to configure and control the RCBs. They provide full control over the behavior of the RCB, but require more detailed knowledge of how RCBs work and more programming effort. You may need to implement a more complex state machine yourself in which you perform multiple read/write accesses to various RCB attribute values using these methods. Use these methods, for example, when you need a behavior that does not conform to the default. In this sample, the use of these methods to configure and control the RCBs will not be discussed further. However, the following steps are necessary, for example, to activate the sending of reports of an unbuffered RCB:

- Read current configuration and status information of the RCB. The values of all attributes: "RptEna", "Resv", "OptFlds", "DatSet", "TrgOps", ..., etc. are read with the method: "GetDataValuesReq".
- Write new configuration values if necessary. New values are written into the attributes "OptFlds", "DatSet" or "TrgOps", ... etc. with the method: "SetDataValuesReq".
- Write the value: "TRUE" into the attribute: "Resv". The RCB is reserved for use with the client.
- Write the value: "TRUE" into the attribute: "RptEna". The sending of the reports is then enabled.

The following steps are necessary, for example, to disable the sending of reports of an unbuffered RCB:

- Write the value "FALSE" into the attribute "RptEna". The sending of reports is disabled.

- Write the value "FALSE" into the attribute "Resv". The reservation for using the RCB with the client is removed.

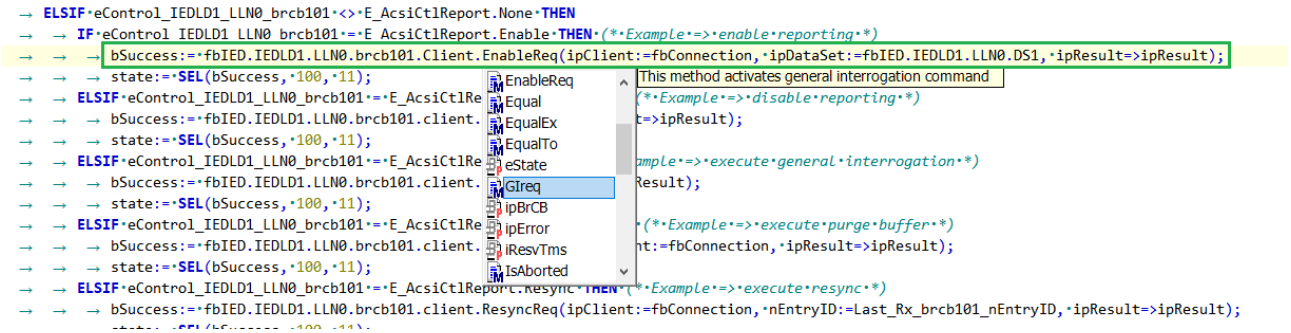
Report configuration/control with the methods: GetUrCBValuesReq, SetUrCBValuesReq, GetBrCBValuesReq, SetBrCBValuesReq

These methods allow reading all or writing one or more attribute values with a single method call. The methods: "GetUrCBValuesReq" or "GetBrCBValuesReq" read all attribute values of a RCB. In this case, if successful, all attribute values of an RCB are transferred from the server to the client and copied to the TwinCAT IEC 61850 data model.

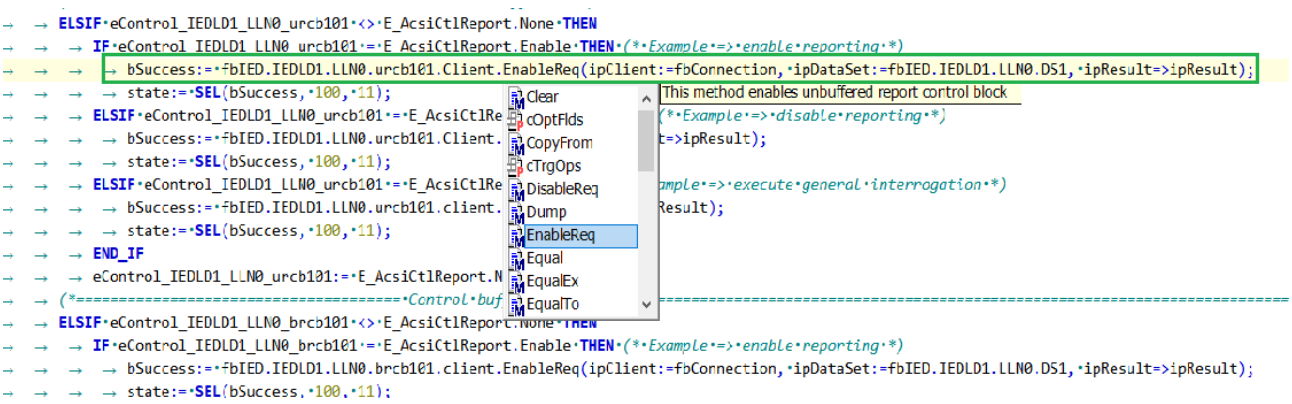
The methods: "SetUrCBValuesReq" and "SetBrCBValuesReq" allow writing one or more attribute values of a RCB. When enabling/disabling the reports, the order in which the attribute values are written plays an important role. Most RCB configuration values can only be written when reports are not active. For unbuffered reports, for example, the attribute value "Resv" must first be set to "TRUE" before the attribute value "RptEna" can be set to "TRUE" (to enable the reports). When disabling the reports, these attribute values must be set to "FALSE" in the reverse order. I.e. first "RptEna" and then "Resv". The Set method parameter "stSet" determines which attribute values are to be written. The new values to be written are first written into the data model, then the corresponding attributes are selected via the "stSet" parameter and the Set method is called. On success, the data is transferred from the IEC 61850 client data model to the server.

Report configuration/control using the methods of the RCB client object

The easiest way to configure and control the buffered or unbuffered reports is to use the client function block below the report control block instance. The client function block has properties to configure the RCB and methods to enable or disable the transmission of reports. For example, in the image below, the IntelliSense shows the available methods and properties with a buffered RCB:



Some methods and properties in unbuffered RCB:



The following table lists the main methods and properties of the client function block at the report control block instance. The client function block works with predefined default values for trigger options, optional data fields and max. time between Integrity Reports unless configured otherwise. The default values can be changed individually for each client instance by writing to the appropriate properties. The new values are first transferred to the server before enabling the report, if they are different from those of the server side, and only then the report is enabled.

Method/property	Description	Initial value
EnableReq	Enables the sending of the reports	-
DisableReq	Disables (stops) the sending of the reports	-
GIReq	Enabless the general interrogation command	-
PurgeBufReq (only for buffered RCBs)	Enables the command to reset the buffer	-
ResyncReq (only for buffered RCBs)	Enables the command for resynchronization of the buffered data	-
Clear	Resets the function block	-
cOptFlds	Configures the data fields contained in the report	Unbuffered RCBs: all data fields enabled except the "DataReference", "BufferOverflow" and "EntryID". Buffered RCBs: all data fields enabled except the "DataReference"
cTrgOps	Configures the trigger options for a report	All enabled ("DataUpdate", "DataChange", "QualityChange", "GeneralInterrogation", "Integrity")
iResvTms	OPTIONAL: configures the time for reserving the RCB for a specific client.	60 if present
nBufTm	Max. report buffer time in milliseconds	0
nIntgPd	Max. time between integrity reports in milliseconds	5000
sRptID	Report ID	Empty string. A new value, if set, will be transmitted to the server only if it is not an empty string and if it is different from the current server value.

When a method is enabled, the current status of the RCB is first queried internally by the server. Next, depending on the selected method, some properties are then transferred to the server. The set values of the properties are transferred to the server only if the value in the property differs from the respective value of the corresponding attribute in the data model. Finally, the status information is queried again internally from the server. Some methods require additional parameters. For example, the "EnableReq" method requires the interface pointer to the connection instance and the DataSet.

The client function block has a structured output variable with the name: "stInfo". This variable contains RCB status information and information about the last report received. However, this status information is only updated if the client function block was also used to configure and enable the reports. The client function block does not update its status information when the other methods described above (SetUrCBValuesReq, SetBrCBValuesReq or SetDataValuesReq, etc.) are used to configure or enable the reports.

Client	FB_ScsmUrCBClientJobClass	
_eState	E_ASYNCSERVICESTATE	Completed
fbError	FB_ServiceErrorClass	
stInfo	ST_ScsmUrCBJobInfo	
eState	E_SCSMRCBSTATE	Enabled
sRptID	T_AcsiVisString129	'IEDLD1/LLN0.urcb101'
sDatSet	T_AcsiObjectReference	'IEDLD1/LLN0.DS1'
nSqNum	BYTE	33
tTimeOfEntry	T_BinaryTime	
timeOfDay	TIME_OF_DAY	TOD#16:12:33.256
day	WORD	13864
stReasonCode	ST_AcsiReasonCode	
DataChange	BIT	FALSE
QualityChange	BIT	FALSE
DataUpdate	BIT	FALSE
Integrity	BIT	TRUE
GeneralInterrogation	BIT	FALSE
ApplicationTrigger	BIT	FALSE
ipAA	I_ScsmAssociationClass	16#FFFFB08F37741F00
_sRptID	T_AcsiVisString129	''
_cTrgOps	ST_AcsiTriggerConditions	
_cOptFlds	ST_AcsiOptionalFields	
_nIntgPd	DWORD	5000
_nBufTm	DWORD	0
_bOwnedResv	BOOL	TRUE

In online mode, the status information of the client function block can be used for simple diagnostics.

Parameter list for configuring the default values of the RCB client object

The default values of all client instances can be changed system-wide, if necessary, by changing the parameter list: Param_Scsm (in the Tc3_iec61850_8_1 PLC library).

```
VAR_GLOBAL CONSTANT
   cBrCB_OptFlds    : ST_AcsiOptionalFields:=(SequenceNumber:=1, ReportTimeStamp:=1, ReasonForInclusi
on:=1, DataSetName:=1, DataReference:=0, BufferOverflow:=1, EntryID:=1, ConfRevision:=1, Segmentatio
n:=0);
   cBrCB_TrgOps     : ST_AcsiTriggerConditions:=(DataChange:=1, QualityChange:=1, DataUpdate:=1, Inte
grity:=1, GeneralInterrogation:=1);
   cBrCB_IntgPd     : DWORD:=5000;

   cUrCB_OptFlds    : ST_AcsiOptionalFields:=(SequenceNumber:=1, ReportTimeStamp:=1, ReasonForInclusi
on:=1, DataSetName:=1, DataReference:=0, BufferOverflow:=0, EntryID:=0, ConfRevision:=1, Segmentatio
n:=0);
   cUrCB_TrgOps     : ST_AcsiTriggerConditions:=(DataChange:=1, QualityChange:=1, DataUpdate:=1, Inte
grity:=1, GeneralInterrogation:=1);
   cUrCB_IntgPd     : DWORD:=5000;
   cBrCB_iResvTms   : INT(-1..3600):=60;
END_VAR
```

Sample project

The unbuffered "urcb101" and the buffered RCB "brcb101" can be controlled in two ways in the sample project. Once with the help of the methods: "GetUrCBValuesReq", "GetBrCBValuesReq", "SetUrCBValuesReq" and "SetBrCBValuesReq". The other option involves the methods at the client function block below the report control block instance. The Boolean and Enum variables described further below can be found in the implementation of the function block: "FB_IEDClient". There is also an ICD file in the project archive. This file describes the IEC 61850 data model used in the sample project. For example, you can use this file in a third-party software and simulate a server.

Test with Get/Set methods

A rising edge at the Boolean variable: "bGetUrCBValues_urcb101" or "bGetBrCBValues_brcb101" enables the reading of all attribute values of the RCB: "urcb101" or "brcb101". By changing the value at the enumeration variable "eSetUrCBValues_urcb101_Resv" and "eSetUrCBValues_urcb101_RptEna" or "eSetBrCBValues_brcb101_RptEna" the Boolean value of the data attributes "Resv" or "RptEna" can be set to "True" or "False". Ideally, if all other RCB attribute values are already preconfigured and these values are already on the server side, the RCB: "urcb101" can be enabled in the following way:
the value "Enable" written to the variable: "eSetUrCBValues_urcb101_Resv" describes the attribute: "Resv" with the value "TRUE". The value "Enable" written to the variable: "eSetUrCBValues_urcb101_RptEna" describes the attribute: "RptEna" with the value "TRUE". After that the transmission of the unbuffered reports of "urcb101" is enabled.

The RCB: "brcb101" can be enabled in the following way:
the value "Enable" written in the variable: "eSetBrCBValues_brcb101_RptEna" describes the attribute: "RptEna" with the value "TRUE". After that the transmission of the buffered reports of "brcb101" is enabled.

```
…
     (*=========================================================================================================
=====================================*)
     (* Write "IEDLD1/LLN0.urcb101.Resv" data attribute value: True|False *)
     ELSIF eSetUrCBValues_urcb101_Resv <> E_CtlBool.None THEN (* Example => execute SetUrCBValuesReq
() command *)
         memset(ADR(stSet_urcb), 0, SIZEOF(stSet_urcb)); (* Clear all URCB attribute selection flags
 *)
         stSet_urcb.Resv:= TRUE; (* Select URCB->Resv attribute to be written *)
         fbIED.IEDLD1.LLN0.urcb101.Resv.bValue:= SEL(eSetUrCBValues_urcb101_Resv = E_CtlBool.Enable,
 FALSE, TRUE); (* Set new data attribute value to be written *)
         bSuccess:= fbConnection.SetUrCBValuesReq(ipUrCB:=fbIED.IEDLD1.LLN0.urcb101, stSet:=stSet_ur
cb, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult); (* Activation of command executi
on *)
         state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command co
mpletion else report an error *)
         eSetUrCBValues_urcb101_Resv:= E_CtlBool.None; (* Reset enum value to recognize next write r
equest *)

     (*=========================================================================================================
=====================================*)
     (* Write "IEDLD1/LLN0.urcb101.RptEna" data attribute value: True|False *)
     ELSIF eSetUrCBValues_urcb101_RptEna <> E_CtlBool.None THEN (* Example => execute SetUrCBValuesR
eq() command *)
         memset(ADR(stSet_urcb), 0, SIZEOF(stSet_urcb)); (* Clear all URCB attribute selection flags
 *)
         stSet_urcb.RptEna:= TRUE; (* Select URCB->RptEna attribute to be written *)
         fbIED.IEDLD1.LLN0.urcb101.RptEna.bValue:= SEL(eSetUrCBValues_urcb101_RptEna = E_CtlBool.Ena
ble, FALSE, TRUE); (* Set new data attribute value to be written *)
         bSuccess:= fbConnection.SetUrCBValuesReq(ipUrCB:=fbIED.IEDLD1.LLN0.urcb101, stSet:=stSet_ur
cb, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult); (* Activation of command executi
on *)
         state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command co
mpletion else report an error *)
         eSetUrCBValues_urcb101_RptEna:= E_CtlBool.None; (* Reset enum value to recognize next write
 request *)

…

     (*=========================================================================================================
=====================================*)
     (* Write "IEDLD1/LLN0.brcb101.RptEna" data attribute value *)
     ELSIF eSetBrCBValues_brcb101_RptEna <> E_CtlBool.None THEN (* Example => execute SetBrCBValuesR
eq() command *)
         memset(ADR(stSet_brcb), 0, SIZEOF(stSet_brcb)); (* Clear all BRCB attribute selection flags
 *)
         stSet_brcb.RptEna:= TRUE; (* Select BRCB->RptEna attribute to be written *)
         fbIED.IEDLD1.LLN0.brcb101.RptEna.bValue:= SEL(eSetBrCBValues_brcb101_RptEna = E_CtlBool.Ena
ble, FALSE, TRUE); (* Set new data attribute value to be written *)
         bSuccess:= fbConnection.SetBrCBValuesReq(ipBrCB:=fbIED.IEDLD1.LLN0.brcb101, stSet:=stSet_br
cb, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult); (* Activation of command executi
on *)
         state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command co
mpletion else report an error *)
         eSetBrCBValues_brcb101_RptEna:= E_CtlBool.None; (* Reset enum value to recognize next write
 request *)

…
```

To stop the transmission of the unbuffered reports, first the value "Disable" must be written into the variable: "eSetUrCBValues_urcb101_RptEna" and then into the variable: "eSetUrCBValues_urcb101_Resv". This sets the values of the data attributes "RptEna" and "Resv" to FALSE.

To stop the transmission of the buffered reports, the value "Disable" must be written to the variable "eSetBrCBValues_brcb101_RptEna". This sets the value of the data attribute "RptEna" to FALSE.

Test with client methods on report control block instance

To consider all states of a RCB, an enum is used here to enable "urcb101" or "brcb101" control commands. As soon as the enum "eControl_IEDLD1_LLN0_urcb101" or "eControl_IEDLD1_LLN0_brcb101" is not "E_AcsiCtlReport.None", the corresponding command is enabled at the RCB.

The "ipResult" output of the function block can be used to query the status of the processing of the last command. Since the execution of the methods takes longer than one cycle, the state machine is set to State 11. There the method "Execute" of the interface pointer "ipResult" must be called until "ipResult.IsBusy()" returns the value "FALSE". If this is currently processing the function, "IsBusy()" is returned as "TRUE" and the state machine remains in State 11. The state machine is reset to State 0 as soon as the command has been successfully executed. The instances "urcb101" and "brcb101" are enabled in this sample with the values for "TrgOps", "OptFlds" and "IntgPd". These values were configured in the TwinCAT Telecontrol Configurator.

```
…
(*=============================================================================================================
====================================*)
(* Control unbuffered reports using "IEDLD1/LLN0.urcb101.Client" function block instance *)
ELSIF eControl_IEDLD1_LLN0_urcb101 <> E_AcsiCtlReport.None THEN
    IF eControl_IEDLD1_LLN0_urcb101 = E_AcsiCtlReport.Enable THEN (* Example => enable reporting *)

        (* Changing default properties of "Client" function block configures reporting behaviour and
 content of report message data.
           The "Client" function block writes the new property values to the server and enables RCB.
           In this example we use configuration values that were defined in the TwinCAT Telecontrol
Configurator to control "urcb101".*)

        (* Configures "RptID" (report ID) to be used in report message *)
        fbIED.IEDLD1.LLN0.urcb101.Client.sRptID:= fbIED.IEDLD1.LLN0.urcb101.RptID.sConfig;
(* => "IEDLD1/.LLN0.urcb101" *)

        (* Configures "TrgOps" (trigger options) to be used to trigger new report messages *)
        fbIED.IEDLD1.LLN0.urcb101.Client.cTrgOps:= fbIED.IEDLD1.LLN0.urcb101.TrgOps.cConfig;
(* => All enabled except "DataUpdate" *)

        (* Configures "OptFlds" (optional fields) to be used in report message *)
        fbIED.IEDLD1.LLN0.urcb101.Client.cOptFlds:= fbIED.IEDLD1.LLN0.urcb101.OptFlds.cConfig;
(* => All enabled except "BufferOverflow" and "EntryID" *)

        (* Configures "IntgPd" (period in milliseconds used to generate integrity report) *)
        fbIED.IEDLD1.LLN0.urcb101.Client.nIntgPd:= fbIED.IEDLD1.LLN0.urcb101.IntgPd.nConfig;
(* => 2000ms *)

        (* Configures "BufTm" (max. report message buffer time in milliseconds) *)
        fbIED.IEDLD1.LLN0.urcb101.Client.nBufTm:= fbIED.IEDLD1.LLN0.urcb101.BufTm.nConfig;
(* => 0ms *)

        (* Write properties and enable RCB *)
        bSuccess:= fbIED.IEDLD1.LLN0.urcb101.Client.EnableReq(ipClient:=fbConnection, ipDataSet:=fbI
ED.IEDLD1.LLN0.DS1, ipResult=>ipResult); (* Activation of command execution *)
        state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command com
pletion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_urcb101 = E_AcsiCtlReport.Disable THEN (* Example => disable reportin
g *)
        bSuccess:= fbIED.IEDLD1.LLN0.urcb101.Client.DisableReq(ipResult=>ipResult); (* Activation of
 command execution *)
        state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command com
pletion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_urcb101 = E_AcsiCtlReport.GI THEN (* Example => execute general inter
rogation *)
        bSuccess:= fbIED.IEDLD1.LLN0.urcb101.Client.GIReq(ipResult=>ipResult); (* Activation of comm
and execution *)
        state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command com
pletion else report an error *)
    END_IF
    eControl_IEDLD1_LLN0_urcb101:= E_AcsiCtlReport.None; (* Reset enum value to recognize next write
 request *)
```

```
…
(*=========================================================================================================
===================================*)
(* Control buffered reports using "IEDLD1/LLN0.brcb101.Client" function block instance *)
ELSIF eControl_IEDLD1_LLN0_brcb101 <> E_AcsiCtlReport.None THEN
    IF eControl_IEDLD1_LLN0_brcb101 = E_AcsiCtlReport.Enable THEN (* Example => enable reporting usi
ng *)
        (*
          Changing default properties of "Client" function block configures reporting behaviour and
content of report message data.
          The "Client" function block writes the new property values to the server and enables RCB.
          In this example we use configuration values that were defined in the TwinCAT Telecontrol C
onfigurator to control "brcb101".
        *)

        (* Configures "RptID" to be used in report message *)
        fbIED.IEDLD1.LLN0.brcb101.Client.sRptID:= fbIED.IEDLD1.LLN0.brcb101.RptID.sConfig;
(* => "IEDLD1/LLN0.brcb101" *)

        (* Configures "TrgOps" (trigger options) to be used to trigger new report messages *)
        fbIED.IEDLD1.LLN0.brcb101.Client.cTrgOps:= fbIED.IEDLD1.LLN0.brcb101.TrgOps.cConfig;
(* => All enabled except "DataUpdate" *)

        (* Configures "OptFlds" (option fields) to be used in report message *)
        fbIED.IEDLD1.LLN0.brcb101.Client.cOptFlds:= fbIED.IEDLD1.LLN0.brcb101.OptFlds.cConfig;
(* => All enabled *)

        (* Configures "IntgPd" (period in milliseconds used to generate integrity report) *)
        fbIED.IEDLD1.LLN0.brcb101.Client.nIntgPd:= fbIED.IEDLD1.LLN0.brcb101.IntgPd.nConfig;
(* => 2000ms *)

        (* Configures "BufTm" (max. report message buffer time in milliseconds) *)
        fbIED.IEDLD1.LLN0.brcb101.Client.nBufTm:= fbIED.IEDLD1.LLN0.brcb101.BufTm.nConfig;
(* => 0ms *)

        (* Write properties and enable RCB *)
        bSuccess:= fbIED.IEDLD1.LLN0.brcb101.Client.EnableReq(ipClient:=fbConnection, ipDataSet:=fbI
ED.IEDLD1.LLN0.DS1, ipResult=>ipResult); (* Activation of command execution *)
        state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command com
pletion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_brcb101 = E_AcsiCtlReport.Disable THEN (* Example => disable reportin
g *)
        bSuccess:= fbIED.IEDLD1.LLN0.brcb101.Client.DisableReq(ipResult=>ipResult); (* Activation of
 command execution *)
        state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command com
pletion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_brcb101 = E_AcsiCtlReport.GI THEN (* Example => execute general inter
rogation *)
        bSuccess:= fbIED.IEDLD1.LLN0.brcb101.Client.GIreq(ipResult=>ipResult); (* Activation of comm
and execution *)
        state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command com
pletion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_brcb101 = E_AcsiCtlReport.PurgeBuf THEN (* Example => execute purge b
uffer *)
        bSuccess:= fbIED.IEDLD1.LLN0.brcb101.Client.PurgeBufReq(ipClient:=fbConnection, ipResult=>ip
Result); (* Activation of command execution *)
        state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command com
pletion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_brcb101 = E_AcsiCtlReport.Resync THEN (* Example => execute resync *)
        bSuccess:= fbIED.IEDLD1.LLN0.brcb101.Client.ResyncReq(ipClient:=fbConnection, nEntryID:=Last
_Rx_brcb101_nEntryID, ipResult=>ipResult); (* Activation of command execution *)
        state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command com
pletion else report an error *)
    END_IF
    eControl_IEDLD1_LLN0_brcb101:= E_AcsiCtlReport.None;
(* Reset enum value to recognize next write request *)
…
```

For demonstration purposes, however, the instances "urcb201" and "brcb201" are not enabled with the values for "TrgOps", "OptFlds", "IntgPd" etc. configured and exported in the TwinCAT Telecontrol Configurator. Instead, the default values of the "client" instance are used when the report is enabled.

```
…
(*=========================================================================================================
===================================*)
(* Control unbuffered reports using "IEDLD1/LLN0.urcb201.Client" function block instance *)
ELSIF eControl_IEDLD1_LLN0_urcb201 <> E_AcsiCtlReport.None THEN
    IF eControl_IEDLD1_LLN0_urcb201 = E_AcsiCtlReport.Enable THEN (* Example => enable reporting *)
       (*
```

```
         Changing default properties of "Client" function block configures reporting behaviour and c
ontent of report message data (optional).
         The "Client" function block writes the new property values to the server and enables RCB.
         In this example we use the standard configuration values
of the function block "Client" to control the "urcb201":
         "fbIED.IEDLD1.LLN0.urcb201.Client.sRptID":= Empty string. Property "sRptID" is written if <
> '' and differs from server value.
         "fbIED.IEDLD1.LLN0.urcb201.Client.cTrgOps":= All trigger options are enabled
         "fbIED.IEDLD1.LLN0.urcb201.Client.cOptFlds":= All options are enabled except "DataReference
", "BufferOverflow" and "EntryID"
         "fbIED.IEDLD1.LLN0.urcb201.Client.nIntgPd":= Set to 5000ms
         "fbIED.IEDLD1.LLN0.urcb201.Client.nBufTm":= Set to 0ms
        *)
       (* Write properties and enable RCB *)
        bSuccess:= fbIED.IEDLD1.LLN0.urcb201.Client.EnableReq(ipClient:=fbConnection, ipDataSet:=fbI
ED.IEDLD1.LLN0.DS2, ipResult=>ipResult); (* Activation of command execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_urcb201 = E_AcsiCtlReport.Disable THEN (* Example => disable reportin
g *)
       bSuccess:= fbIED.IEDLD1.LLN0.urcb201.Client.DisableReq(ipResult=>ipResult); (* Activation of
command execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_urcb201 = E_AcsiCtlReport.GI THEN (* Example => execute general inter
rogation *)
       bSuccess:= fbIED.IEDLD1.LLN0.urcb201.Client.GIReq(ipResult=>ipResult); (* Activation of comma
nd execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    END_IF
    eControl_IEDLD1_LLN0_urcb201:= E_AcsiCtlReport.None; (* Reset enum value to recognize next write
 request *)

…
(*=================================================================================================
====================================*)
(* Control buffered reports using "IEDLD1/LLN0.brcb201.Client" function block instance *)
ELSIF eControl_IEDLD1_LLN0_brcb201 <> E_AcsiCtlReport.None THEN
    IF eControl_IEDLD1_LLN0_brcb201 = E_AcsiCtlReport.Enable THEN (* Example => enable reporting *)

       (*
         Changing default properties of "Client" function block configures reporting behaviour and c
ontent of report message data (optional).
         The "Client" function block writes the new property values to the server and enables RCB.
         In this example we use the standard configuration values
of the function block "Client" to control the "brcb201":
         "fbIED.IEDLD1.LLN0.brcb201.Client.sRptID":= Empty string. Property "sRptID" is only written
 if <> '' and differs from server value
         "fbIED.IEDLD1.LLN0.brcb201.Client.cTrgOps":= All trigger options are enabled
         "fbIED.IEDLD1.LLN0.brcb201.Client.cOptFlds":= All options are enabled except "DataReference
"
         "fbIED.IEDLD1.LLN0.brcb201.Client.nIntgPd":= Set to 5000ms
         "fbIED.IEDLD1.LLN0.brcb201.Client.nBufTm":= Set to 0ms
       *)

       (* Write properties and enable RCB *)
       bSuccess:= fbIED.IEDLD1.LLN0.brcb201.Client.EnableReq(ipClient:=fbConnection, ipDataSet:=fbIE
D.IEDLD1.LLN0.DS2, ipResult=>ipResult); (* Activation of command execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_brcb201 = E_AcsiCtlReport.Disable THEN (* Example => disable reportin
g *)
       bSuccess:= fbIED.IEDLD1.LLN0.brcb201.Client.DisableReq(ipResult=>ipResult); (* Activation of
command execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_brcb201 = E_AcsiCtlReport.GI THEN (* Example => execute general inter
rogation *)
       bSuccess:= fbIED.IEDLD1.LLN0.brcb201.Client.GIreq(ipResult=>ipResult); (* Activation of comma
nd execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_brcb201 = E_AcsiCtlReport.PurgeBuf THEN (* Example => execute purge b
uffer *)
       bSuccess:= fbIED.IEDLD1.LLN0.brcb201.Client.PurgeBufReq(ipClient:=fbConnection, ipResult=>ipR
esult); (* Activation of command execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_brcb201 = E_AcsiCtlReport.Resync THEN (* Example => execute resync *)
```

```
       bSuccess:= fbIED.IEDLD1.LLN0.brcb201.Client.ResyncReq(ipClient:=fbConnection, nEntryID:=Last_
Rx_brcb201_nEntryID, ipResult=>ipResult); (* Activation of command execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    END_IF
    eControl_IEDLD1_LLN0_brcb201:= E_AcsiCtlReport.None; (* Reset enum value to recognize next write
 request *)

…
```

The "urcb301" and "brcb301" instances are in turn enabled with user-defined values for "TrgOps", "OptFlds" and "IntgPd".

```
…
(*=============================================================================================
====================================*)
(* Control unbuffered reports using "IEDLD1/LLN0.urcb301.Client" function block instance *)
ELSIF eControl_IEDLD1_LLN0_urcb301 <> E_AcsiCtlReport.None THEN
    IF eControl_IEDLD1_LLN0_urcb301 = E_AcsiCtlReport.Enable THEN (* Example => enable reporting *)

       (*
         Changing default properties of "Client" function block configures reporting behaviour and c
ontent of report message data.
         The "Client" function block writes the new property values to the server and enables RCB.
         In this example we use configuration values that were defined by application/
user to control the "urcb301".
       *)

       (* Configures "RptID" (report ID) to be used in report message *)
        fbIED.IEDLD1.LLN0.urcb301.Client.sRptID:= 'IEDLD1/LLN0.RP.urcb301';

       (* Configures "TrgOps" (trigger options) to be used to trigger new report messages *)
       usrTrgOps.DataChange:= TRUE;
       usrTrgOps.DataUpdate:= FALSE;
       usrTrgOps.QualityChange:= FALSE;
       usrTrgOps.Integrity:= TRUE;
       usrTrgOps.GeneralInterrogation:= TRUE;
       fbIED.IEDLD1.LLN0.urcb301.Client.cTrgOps:= usrTrgOps;

       (* Configures "OptFlds" (option fields) to be used in report message *)
       usrOptFlds.ConfRevision:= TRUE;
       usrOptFlds.DataReference:= FALSE;
       usrOptFlds.DataSetName:= FALSE;
       usrOptFlds.ReasonForInclusion:= TRUE;
       usrOptFlds.ReportTimeStamp:= FALSE;
       usrOptFlds.SequenceNumber:= TRUE;
       usrOptFlds.BufferOverflow:= FALSE;
       usrOptFlds.EntryID:= FALSE;
       fbIED.IEDLD1.LLN0.urcb301.Client.cOptFlds:= usrOptFlds;

       (* Configures "IntgPd" (period in milliseconds used to generate integrity report) *)
       fbIED.IEDLD1.LLN0.urcb301.Client.nIntgPd:= 1000;

       (* Configures "BufTm" (max. report message buffer time in milliseconds) *)
       fbIED.IEDLD1.LLN0.urcb301.Client.nBufTm:= 500;

       (* Write properties and enable RCB *)
       bSuccess:= fbIED.IEDLD1.LLN0.urcb301.Client.EnableReq(ipClient:=fbConnection, ipDataSet:=fbIE
D.IEDLD1.LLN0.DS3, ipResult=>ipResult); (* Activation of command execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_urcb301 = E_AcsiCtlReport.Disable THEN (* Example => disable reportin
g *)
       bSuccess:= fbIED.IEDLD1.LLN0.urcb301.Client.DisableReq(ipResult=>ipResult); (* Activation of
command execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_urcb301 = E_AcsiCtlReport.GI THEN (* Example => execute general inter
rogation *)
       bSuccess:= fbIED.IEDLD1.LLN0.urcb301.Client.GIReq(ipResult=>ipResult); (* Activation of comma
nd execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    END_IF
    eControl_IEDLD1_LLN0_urcb301:= E_AcsiCtlReport.None; (* Reset enum value to recognize next write
 request *)

…
(*=============================================================================================
====================================*)
```

```
(* Control buffered reports using "IEDLD1/LLN0.brcb301.Client" function block instance *)
ELSIF eControl_IEDLD1_LLN0_brcb301 <> E_AcsiCtlReport.None THEN
    IF eControl_IEDLD1_LLN0_brcb301 = E_AcsiCtlReport.Enable THEN (* Example => enable reporting usi
ng *)

       (*
         Changing default properties of "Client" function block configures reporting behaviour and c
ontent of report message data.
         The "Client" function block writes the new property values to the server and enables RCB.
         In this example we use configuration values that were defined by application/
user to control the "brcb301".
        *)

       (* Configures "RptID" (report ID) to be used in report message *)
       fbIED.IEDLD1.LLN0.brcb301.Client.sRptID:= 'IEDLD1/LLN0.BR.brcb301';

       (* Configures "TrgOps" (trigger options) to be used to trigger new report messages *)
       usrTrgOps.DataChange:= TRUE;
       usrTrgOps.DataUpdate:= FALSE;
       usrTrgOps.QualityChange:= FALSE;
       usrTrgOps.Integrity:= TRUE;
       usrTrgOps.GeneralInterrogation:= TRUE;
       fbIED.IEDLD1.LLN0.brcb301.Client.cTrgOps:= usrTrgOps;

       (* Configures "OptFlds" (option fields) to be used in report message *)
       usrOptFlds.ConfRevision:= TRUE;
       usrOptFlds.DataReference:= FALSE;
       usrOptFlds.DataSetName:= FALSE;
       usrOptFlds.ReasonForInclusion:= TRUE;
       usrOptFlds.ReportTimeStamp:= FALSE;
       usrOptFlds.SequenceNumber:= TRUE;
       usrOptFlds.BufferOverflow:= TRUE;
       usrOptFlds.EntryID:= TRUE;
       fbIED.IEDLD1.LLN0.brcb301.Client.cOptFlds:= usrOptFlds;

       (* Configures "IntgPd" (period in milliseconds used to generate integrity report) *)
       fbIED.IEDLD1.LLN0.brcb301.Client.nIntgPd:= 10000;

       (* Configures "BufTm" (max. report message buffer time in milliseconds) *)
       fbIED.IEDLD1.LLN0.brcb301.Client.nBufTm:= 500;

       (* Write properties and enable RCB *)
       bSuccess:= fbIED.IEDLD1.LLN0.brcb301.Client.EnableReq(ipClient:=fbConnection, ipDataSet:=fbIE
D.IEDLD1.LLN0.DS3, ipResult=>ipResult); (* Activation of command execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_brcb301 = E_AcsiCtlReport.Disable THEN (* Example => disable reportin
g *)
       bSuccess:= fbIED.IEDLD1.LLN0.brcb301.Client.DisableReq(ipResult=>ipResult); (* Activation of
command execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_brcb301 = E_AcsiCtlReport.GI THEN (* Example => execute general inter
rogation *)
       bSuccess:= fbIED.IEDLD1.LLN0.brcb301.Client.GIreq(ipResult=>ipResult); (* Activation of comma
nd execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_brcb301 = E_AcsiCtlReport.PurgeBuf THEN (* Example => execute purge b
uffer *)
       bSuccess:= fbIED.IEDLD1.LLN0.brcb301.Client.PurgeBufReq(ipClient:=fbConnection, ipResult=>ipR
esult); (* Activation of command execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    ELSIF eControl_IEDLD1_LLN0_brcb301 = E_AcsiCtlReport.Resync THEN (* Example => execute resync *)
       bSuccess:= fbIED.IEDLD1.LLN0.brcb301.Client.ResyncReq(ipClient:=fbConnection, nEntryID:=Last_
Rx_brcb301_nEntryID, ipResult=>ipResult); (* Activation of command execution *)
       state:= SEL(bSuccess, 100, 11); (* If command activation succeeded then wait for command comp
letion else report an error *)
    END_IF
    eControl_IEDLD1_LLN0_brcb301:= E_AcsiCtlReport.None; (* Reset enum value to recognize next write
 request *)

…
```



This is just one example of the implementation of the Report Control Block function in the TwinCAT PLC.

7.6 Client - Direct Control with normal security

This sample shows the exemplary implementation of the direct commands (ctlModel := 1, direct control with normal security) in a TwinCAT IEC 61850 client project for the following switch control data classes (Common Data Class, CDC):

- SPC (controllable single point) object
- DPC (controllable double point) object
- APC (controllable analogue process value) object
- BAC (binary controlled analogue process value) object
- BSC (binary controlled step position information) object
- ENC (controllable enumerated status) object: Mod
- INC (controllable integer status) object
- ISC (integer controlled step position information) object

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/5494787339.zip

The example described here uses the state machine that is described in the "General Client project structure [▶ 664]" chapter. The States 0, 1, 11 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

An additional, self-implemented function block is required to implement the direct commands. In the sample a new function block: "FB_DirectControl" has been added for this purpose. The function block: "FB_DirectControl" handles the actual command execution for the different CDC types. In the modified State 10 of the "FB_IEDClient" several IF statements are listed, which enable the command execution of the client via the instance of the "FB_DirectControl" function block. Several Boolean variables are defined in the declaration part of "FB_IEDClient". A rising edge at the corresponding Boolean variable enables command execution in the IF statement. The user can initiate this via the online values in the application.

Methods of the "FB_DirectControl" function block

Method name	Description
APC BAC BSC DPC ENC_Mod INC ISC SPC	A method is added for each controllable CDC type, as most CDCs use different control value types, for example, "APC" -> "REAL" and "SPC" -> "BOOL". As the CDCs use different types, each of these methods also has a different input parameter for the control value (e.g. "ctlVal"). Inside the method the value is written to a union type that contains a member for each CDC type. The union variable is passed to the "Request" method. The "Request" method is a private method and it activates the command execution.
Execute	As soon as the "Request" method of the function block has been successfully called, the "Execute" method must be called in every cycle. Within this method there is a separate state machine that works independently of the state machine in the client function block. As a result of this, the code is divided into smaller sections and is easier to extend in future.
Clear	Aborts the command and resets the function block.

The command, depending on the CDC type, modifies the control value of the CDC. For demonstration purposes, a numerical value is incremented or a Boolean value is toggled, for example.

The state machine of the "FB_IEDClient" function block is set to State 11 as soon as a command has been initiated and successfully executed. The state machine waits here until the command has been successfully executed. For this purpose, the "Execute" method is called in every cycle in State 11 and it is possible, on

the basis of the "ipResult" interface pointer, to determine whether the command is still being processed or whether it has already been completed. This is queried via the "IsBusy()" method. The state machine switches to State 0 as soon as the command has been successfully executed.



This is only an exemplary implementation of the direct commands in the TwinCAT PLC.

7.7 Client - SBO Control with normal security

This sample shows the exemplary implementation of the command: select-before-operate (ctlModel := 2, select before operate control with normal security) in a TwinCAT IEC 61850 client project for the following switch control data classes (Common Data Class, CDC):

- SPC (controllable single point) object;
- DPC (controllable double point) object
- APC (controllable analogue process value) object
- BAC (binary controlled analogue process value) object
- BSC (binary controlled step position information) object
- ENC (controllable enumerated status) object: Mod
- INC (controllable integer status) object
- ISC (integer controlled step position information) object

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/5494803595.zip

The example described here uses the state machine that is described in the "General Client project structure [▶ 664]" chapter. The States 0, 1, 11 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

For the implementation of the commands: select-before-operate an additional, self-implemented function block is required. In the sample a new function block: "FB_SBOControl" has been added for this purpose. The function block: "FB_SBOControl" handles the actual command execution for the different CDC types.

In the modified State 10 of the "FB_IEDClient" several IF statements are listed, which enable the command execution of the client via the instance of the "FB_SBOControl" function block. An enum is used here for activation in order to consider all states of the SBO command (SBO = select before operate). The command is activated as soon as one of the enums, for example, "eControl_LLN0_Mod" is not "E_AcsiCtlService.None". The user can initiate this via the online values in the application.

Methods of the "FB_DirectControl" function block

Method name	Description
APC BAC BSC DPC ENC_Mod INC ISC SPC	A method is added for each controllable CDC type, as most CDCs use different types, for example, "APC" -> "REAL" and "SPC" -> "BOOL". As the CDCs use different types, each of these methods also has a different input parameter for the control value (e.g. "ctlVal"). Inside the method the value is written to a union type that contains a member for each CDC type. The union variable is passed to the "Request" method. The "Request" method is a private method and it activates the command execution.
Execute	As soon as the "Request" method of the function block has been successfully called, the "Execute" method must be called in every cycle. Within this method there is a separate state machine that works independently of the state machine in the client function block. As a result of this, the code is divided into smaller sections and is easier to extend in future.
Clear	Aborts the command and resets the function block.

The command, depending on the CDC type, modifies the control value of the CDC. For demonstration purposes, a numerical value is incremented or a Boolean value is toggled, for example.

The execution changes depending on the enum value for initiating a command. The enum can take the following values: None, Select, Operate, Select&Operate, Cancel. Only one command is executed in the case of Select, Operate and Cancel. In the case of Select & Operate, a Select command is executed first and, as soon as this has been successfully completed, the Operate command begins. The user can choose one of these options when switching over the enum in the online values.

As soon as a change takes place with one of the enum values, the respective method to suit the CDC type is called. If this has been successfully completed, the state machine changes to State 11. This is necessary because the processing of the commands takes longer than one cycle in the PLC. The "Execute" method is called cyclically within this state. On the basis of the "ipResult" interface pointer, it is possible to determine whether the command is still being processed or whether it has already been completed. This is queried via the "IsBusy()" method. The state machine switches to State 0 as soon as the command has been successfully executed.



This is only an exemplary implementation of the commands: select-before-operate in the TwinCAT PLC.

7.8 Client - Direct Control with enhanced security

This sample shows the exemplary implementation of direct control with enhanced security (ctlModel := 3, direct control with enhanced security) in a TwinCAT IEC 61850 client project for the following switch control data classes (Common Data Class, CDC):

- SPC (controllable single point) object
- DPC (controllable double point) object
- APC (controllable analogue process value) object
- BAC (binary controlled analogue process value) object
- BSC (binary controlled step position information) object
- ENC (controllable enumerated status) object: Mod
- INC (controllable integer status) object
- ISC (integer controlled step position information) object

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/5494805899.zip

The example described here uses the state machine that is described in the "General Client project structure [▶ 664]" chapter. The States 0, 1, 11 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

An additional, self-implemented function block is required to implement the direct commands. In the sample a new function block: "FB_DIRwControl" has been added for this purpose. The function block: "FB_DIRwControl" handles the actual command execution for the different CDC types. In the modified State 10 several IF statements are listed, which enable the command execution of the client via the instance of the "FB_DIRwControl" function block.

Several Boolean variables are defined in the declaration part of "FB_IEDClient". A rising edge at the corresponding Boolean variable enables command execution in the IF statement. The user can initiate this via the online values in the application.

Methods of the "FB_DIRwControl" function block

Method name	Description
APC BAC BSC DPC ENC_Mod INC ISC SPC	A method is added for each controllable CDC, as most CDCs use different control value types, for example, "APC" -> "REAL" and "SPC" -> "BOOL". As the CDCs use different types, each of these methods also has a different input parameter for the control value (e.g. "ctlVal"). Inside the method the value is written to a union type that contains a member for each CDC. The union variable is passed to the "Request" method. The "Request" method is a private method and it activates the command execution.
Execute	As soon as the "Request" method of the function block has been successfully called, the "Execute" method must be called in every cycle. Within this method there is a separate state machine that works independently of the state machine in the client function block. As a result of this, the code is divided into smaller sections and is easier to extend in future.
Clear	Aborts the command and resets the function block.

The command, depending on the CDC type, modifies the control value of the CDC. For demonstration purposes, a numerical value is incremented or a Boolean value is toggled, for example.

The state machine of the "FB_IEDClient" function block is set to State 11 as soon as a command has been initiated and successfully executed. The state machine waits here until the command has been successfully executed. For this purpose, the "Execute" method is called in every cycle in State 11 and it is possible, on the basis of the "ipResult" feedback interface pointer, to determine whether the command is still being processed or whether it has already been completed. This is queried via the "IsBusy()" method. The state machine switches to State 0 as soon as the command has been successfully executed.

As opposed to Sample04 with the implementation example for "Direct control with normal security", the user receives the event "OnCommandTerminationInd" at the "FB_IEDClient" function block when the "Direct control with enhanced security" is successfully executed. If an error occurs when executing the command, the user receives the event "OnLastApplErrorInd".



This is only an exemplary implementation of the direct control with enhanced security in the TwinCAT PLC.

7.9 Client - SBO Control with enhanced security

This sample shows the exemplary implementation of the command: select-before-operate-control-with-enhanced-security (ctlModel := 4, select before operate control with enhanced security) in a TwinCAT IEC 61850 client project for the following switch control data classes (Common Data Class, CDC):

- SPC (controllable single point) object

- DPC (controllable double point) object
- APC (controllable analogue process value) object
- BAC (binary controlled analogue process value) object
- BSC (binary controlled step position information) object
- ENC (controllable enumerated status) object: Mod
- INC (controllable integer status) object
- ISC (integer controlled step position information) object

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/5494808203.zip

The example described here uses the state machine that is described in the "General Client project structure [▶ 664]" chapter. The States 0, 1, 11 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

For the implementation of the commands: select before operate control with enhanced security, an additional, self-implemented function block is required. In the sample a new function block: "FB_SBOwControl" has been added for this purpose. The function block: "FB_SBOwControl" handles the actual command execution for the different CDC types.

In the modified State 10 of the "FB_IEDClient" several IF statements are listed, which enable the command execution of the client via the instance of the "FB_SBOwControl" function block. An enum is used here for activation in order to consider all states of the SBO command (SBO = select before operate). The command is activated as soon as one of the enums, for example, "eControl_LLN0_Mod" is not "E_AcsiCtlService.None". The user can initiate this via the online values in the application.

Methods of the "FB_DirectControl" function block

Method name	Description
APC BAC BSC DPC ENC_Mod INC ISC SPC	A method is added for each controllable CDC type, as most CDCs use different types, for example, "APC" -> "REAL" and "SPC" -> "BOOL". As the CDCs use different types, each of these methods also has a different input parameter for the control value (e.g. "ctlVal"). Inside the method the value is written to a union type that contains a member for each CDC. The union variable is passed to the "Request" method. The "Request" method is a private method and it activates the command execution.
Execute	As soon as the "Request" method of the function block has been successfully called, the "Execute" method must be called in every cycle. Within this method there is a separate state machine that works independently of the state machine in the client function block. As a result of this, the code is divided into smaller sections and is easier to extend in future.
Clear	Aborts the command and resets the function block.

The command, depending on the CDC type, modifies the control value of the CDC. For demonstration purposes, a numerical value is incremented or a Boolean value is toggled, for example.

The execution changes depending on the enum value for initiating a command. The enum can take the following values: None, Select, Operate, Select&Operate, Cancel. Only one command is executed in the case of Select, Operate and Cancel. In the case of Select & Operate, a Select command is executed first and, as soon as this has been successfully completed, the Operate command begins. The user can choose one of these options when switching over the enum in the online values.

As soon as a change takes place with one of the enum values, the respective method to suit the CDC type is called. If this has been successfully completed, the state machine changes to State 11. This is necessary because the processing of the commands takes longer than one cycle in the PLC. The "Execute" method is called cyclically within this state. On the basis of the "ipResult" interface pointer, it is possible to determine

whether the command is still being processed or whether it has already been completed. This is queried via the "IsBusy()" method. The state machine switches to State 0 as soon as the command has been successfully executed.

As opposed to Sample05 with the implementation example for "Select before operate control with normal security", the user receives the event "OnCommandTerminationInd" at the "FB_IEDClient" function block when the "Select before operate with enhanced security" is successfully executed. If an error occurs when executing the command, the user receives the event "OnLastApplErrorInd".



This is only an exemplary implementation of the commands: select before operate with enhanced security in the TwinCAT PLC.

7.10 Client - Read/Write DataSet Values (GetDataSetValues, SetDataSetValues)

This sample shows the use of the "GetDataSetValuesReq" and "SetDataSetValuesReq" methods of the client function block.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/8922514571.zip

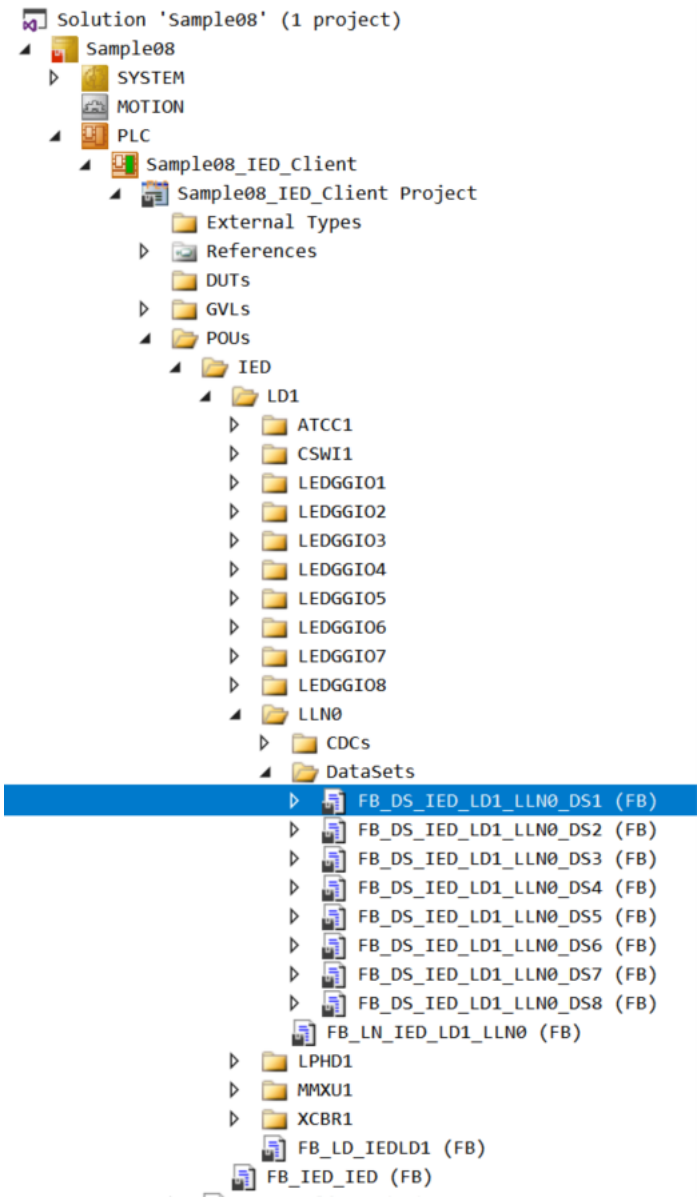
The example described here uses the state machine that is described in the "General Client project structure [▶ 664]" chapter. The States 0, 1, 11 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

General information about IEC 61850 DataSets

A DataSet is a list of data attributes or data that can be transmitted together (usually in a report or GOOSE message). The individual list entries of a DataSet are called DataSet members. The DataSet members are configured in the TwinCAT Telecontrol Configurator. The PLC code of the DataSets and DataSet members is automatically generated by the configurator during TwinCAT project generation. With a DataSet several data or data attributes can be read or written at the same time. The client method "GetDataSetValuesReq" can read all DataSet members defined in the DataSet together (the values are transferred from the server to the client). The client method "SetDataSetValuesReq" can be used to write to all DataSet members defined in the DataSet (the values are transferred from the client to the server).

Preferably the DataSets are found in the IEC 61850 data model below the logical node LLN0. However, a DataSet can theoretically be assigned to (linked to) any other logical node.

The following figure shows the DataSet function blocks created during project generation in the TwinCAT project tree:

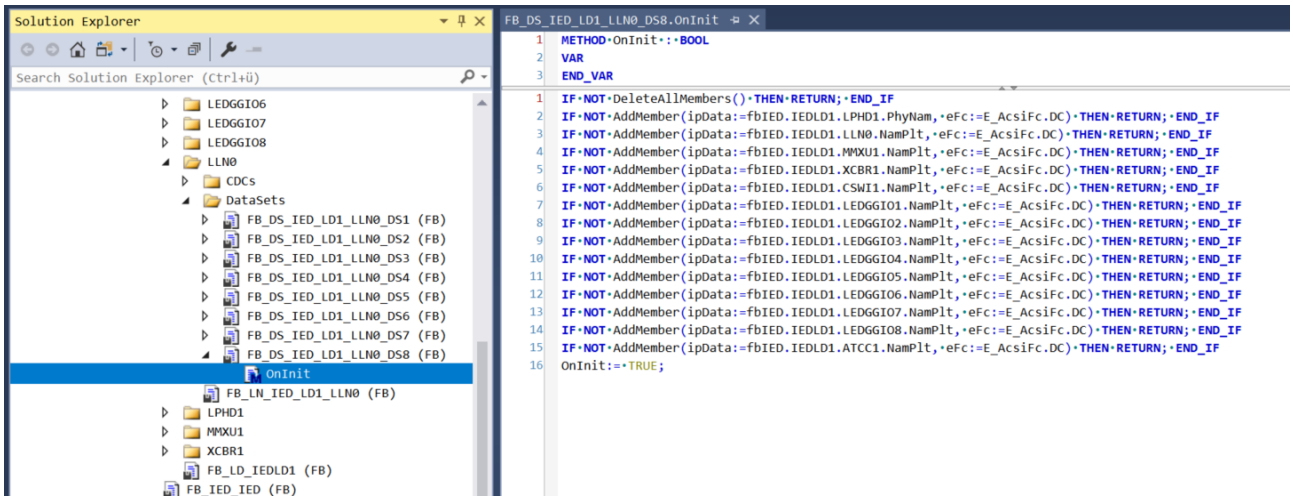


The DataSet function blocks shown belong to LLN0 and for this reason they are also instantiated in the declaration part of LLN0. The instances of the DataSets: "DS1" to "DS8" and the link to the logical node were also generated automatically.

FB_LN_IED_LD1_LLN0

```
1  FUNCTION_BLOCK FB_LN_IED_LD1_LLN0 EXTENDS FB_AcsiCommonLogicalNodeClass
2  VAR_INPUT
3    → NamPlt: FB_DO_IED_LD1_LLN0_NamPlt := (sClass:='LPL', bLinkResult:=THIS^.AddDataToContainer(ipData:=NamPlt));
4    → Beh: FB_DO_IED_LD1_LLN0_Beh := (sClass:='ENS', bLinkResult:=THIS^.AddDataToContainer(ipData:=Beh));
5    → Health: FB_DO_IED_LD1_LLN0_Health := (sClass:='ENS', bLinkResult:=THIS^.AddDataToContainer(ipData:=Health));
6    → Mod_: FB_DO_IED_LD1_LLN0_Mod := (sObjectName:='Mod', sClass:='ENC', bLinkResult:=THIS^.AddDataToContainer(ipData:=Mod_))
7    →
8    → DS1: FB_DS_IED_LD1_LLN0_DS1 := (bLinkResult:=THIS^.AddDataSetToContainer(ipDataSet:=DS1));
9    → DS2: FB_DS_IED_LD1_LLN0_DS2 := (bLinkResult:=THIS^.AddDataSetToContainer(ipDataSet:=DS2));
10   → DS3: FB_DS_IED_LD1_LLN0_DS3 := (bLinkResult:=THIS^.AddDataSetToContainer(ipDataSet:=DS3));
11   → DS4: FB_DS_IED_LD1_LLN0_DS4 := (bLinkResult:=THIS^.AddDataSetToContainer(ipDataSet:=DS4));
12   → DS5: FB_DS_IED_LD1_LLN0_DS5 := (bLinkResult:=THIS^.AddDataSetToContainer(ipDataSet:=DS5));
13   → DS6: FB_DS_IED_LD1_LLN0_DS6 := (bLinkResult:=THIS^.AddDataSetToContainer(ipDataSet:=DS6));
14   → DS7: FB_DS_IED_LD1_LLN0_DS7 := (bLinkResult:=THIS^.AddDataSetToContainer(ipDataSet:=DS7));
15   → DS8: FB_DS_IED_LD1_LLN0_DS8 := (bLinkResult:=THIS^.AddDataSetToContainer(ipDataSet:=DS8));
16   →
17   → urcb101: FB_ScsmUrCBImplClass := (RptID:=(sValue:='IEDLD1/LLN0.urcb101'), DatSet:=(sValue:='IEDLD1/LLN0.DS1'), ConfRev:=
18   → brcb101: FB_ScsmBrCBImplClass := (RptID:=(sValue:='IEDLD1/LLN0.brcb101'), DatSet:=(sValue:='IEDLD1/LLN0.DS1'), ConfRev:=
19 END_VAR
20 VAR
21 END_VAR
```

The PLC code for the configuration of the DataSet members is also automatically generated like the DataSets themselves. This code is located in the OnInit method of the respective DataSet function block (see figure below).



Static DataSets vs. dynamic DataSets

The TwinCAT project generation creates static, persistent DataSets. I.e. these DataSets are always available in the data model and cannot and should not be deleted. However, there are also dynamic DataSets that can be created during communication. The dynamically created DataSets can be deleted again. They are deleted either automatically, when the connection is terminated (dynamically created non-persistent data-sets) or at a later time, also by another client (dynamically created persistent data-sets).

Sample project

In the zip archive there is an ICD configuration file. This file can be used, for example, to simulate a server using third-party software or to create a new or modified TwinCAT project with TwinCAT Telecontrol Configurator.

If you plan to use a DataSet to write multiple data or data attributes at the same time, then you still need to consider the following: all configured DataSet members must also allow write access, otherwise the write access will fail. This depends mainly on the functional group (FC) of the DataSet member. In this sample, for the purpose of demonstrating write access, the DataSet "DS8" has been configured so that all DataSet members can be written to (functional group: "DC"). In our client sample project, of course, write access to all DataSet members must be possible on the server side.

Several Boolean variables are defined in the FB_IEDClient function block. A rising edge at one of these variables enables the method: "GetDataSetValuesReq" or "SetDataSetValuesReq" with the respective DataSet as input parameter. As an example: a rising edge at the variable "bGetDataSetValues_LLN0_DS1" enables the command for reading the DataSet: "DS1" and a rising edge at "bSetDataSetValues_LLN0_DS8" enables the command for writing the DataSet: "DS8".

```
…
bGetDataSetValues_LLN0_DS1 : BOOL := TRUE;
bGetDataSetValues_LLN0_DS2 : BOOL := TRUE;
bGetDataSetValues_LLN0_DS3 : BOOL := TRUE;
bGetDataSetValues_LLN0_DS4 : BOOL := TRUE;
bGetDataSetValues_LLN0_DS5 : BOOL := TRUE;
bGetDataSetValues_LLN0_DS6 : BOOL := TRUE;
bGetDataSetValues_LLN0_DS7 : BOOL := TRUE;
bGetDataSetValues_LLN0_DS8 : BOOL := TRUE;
bSetDataSetValues_LLN0_DS8 : BOOL := TRUE;
…
```

The following is an excerpt from the sample code for reading the DataSet: "DS8". First, the method call "GetDataSetValuesReq" enables the transfer of data from the server to the client. After that, in further PLC cycles the method "ipResult.Execute()" must be called until the transfer is completed. "ipResult.IsBusy()" returns FALSE in this case. The received data values or attribute values of the DataSet members are automatically copied (mapped) to the client data model upon success.

```
…
ELSIF bGetDataSetValues_LLN0_DS8 THEN
    bGetDataSetValues_LLN0_DS8:= FALSE;
```

```
    bSuccess:= fbConnection.GetDataSetValuesReq(ipDataSet:=fbIED.IEDLD1.LLN0.DS8, hUser:=0, ipSink:=
0, nInvokeID=>nInvokeID, ipResult=>ipResult);
    state:= SEL(bSuccess, 100, 11);
…
```

In the sample code for writing the DataSet: "DS8", the data to be written to the DataSet: "DS8" is modified each time for demonstration purposes. First, new data values or attribute values are assigned to DataSet members in the client data model. After that the method call "SetDataSetValuesReq" activates the transfer of the data to the server. In further PLC cycles, the method "ipResult.Execute()" must be called analogously as already described above until the transfer has been completed. The new data values or attribute values are copied (mapped) into the server data model on the server side.

```
…
ELSIF bSetDataSetValues_LLN0_DS8 THEN
    bSetDataSetValues_LLN0_DS8:= FALSE;
    (* As example we modify some description data values *)
    sConfigRev:= TO_STRING(nRev:=nRev+1);
    sHwRev:= TO_STRING(TO_REAL(nRev));
    sSwRev:= TO_STRING(TO_REAL(nRev));

    fbIED.IEDLD1.LPHD1.PhyNam.Vendor.sValue:= sVendor;
    fbIED.IEDLD1.LPHD1.PhyNam.hwRev.sValue:= sHwRev;
    fbIED.IEDLD1.LPHD1.PhyNam.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.LPHD1.PhyNam.serNum.sValue:= sSerNum;
    fbIED.IEDLD1.LPHD1.PhyNam.model.sValue:= sModel;
    fbIED.IEDLD1.LPHD1.PhyNam.location.sValue:= sLocation;

    fbIED.IEDLD1.LLN0.NamPlt.vendor.sValue:= sVendor;
    fbIED.IEDLD1.LLN0.NamPlt.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.LLN0.NamPlt.d.sValue:= 'LLN0 demo node';
    fbIED.IEDLD1.LLN0.NamPlt.configRev.sValue:= sConfigRev;

    fbIED.IEDLD1.MMXU1.NamPlt.vendor.sValue:= sVendor;
    fbIED.IEDLD1.MMXU1.NamPlt.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.MMXU1.NamPlt.d.sValue:= 'MMXU1 demo node';
    fbIED.IEDLD1.MMXU1.NamPlt.configRev.sValue:= sConfigRev;

    fbIED.IEDLD1.XCBR1.NamPlt.vendor.sValue:= sVendor;
    fbIED.IEDLD1.XCBR1.NamPlt.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.XCBR1.NamPlt.d.sValue:= 'XCBR1 demo node';
    fbIED.IEDLD1.XCBR1.NamPlt.configRev.sValue:= sConfigRev;

    fbIED.IEDLD1.CSWI1.NamPlt.vendor.sValue:= sVendor;
    fbIED.IEDLD1.CSWI1.NamPlt.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.CSWI1.NamPlt.d.sValue:= 'CSWI1 demo node';
    fbIED.IEDLD1.CSWI1.NamPlt.configRev.sValue:= sConfigRev;

    fbIED.IEDLD1.LEDGGIO1.NamPlt.vendor.sValue:= sVendor;
    fbIED.IEDLD1.LEDGGIO1.NamPlt.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.LEDGGIO1.NamPlt.d.sValue:= 'LEDGGIO1 demo node';

    fbIED.IEDLD1.LEDGGIO2.NamPlt.vendor.sValue:= sVendor;
    fbIED.IEDLD1.LEDGGIO2.NamPlt.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.LEDGGIO2.NamPlt.d.sValue:= 'LEDGGIO2 demo node';

    fbIED.IEDLD1.LEDGGIO3.NamPlt.vendor.sValue:= sVendor;
    fbIED.IEDLD1.LEDGGIO3.NamPlt.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.LEDGGIO3.NamPlt.d.sValue:= 'LEDGGIO3 demo node';

    fbIED.IEDLD1.LEDGGIO4.NamPlt.vendor.sValue:= sVendor;
    fbIED.IEDLD1.LEDGGIO4.NamPlt.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.LEDGGIO4.NamPlt.d.sValue:= 'LEDGGIO4 demo node';

    fbIED.IEDLD1.LEDGGIO5.NamPlt.vendor.sValue:= sVendor;
    fbIED.IEDLD1.LEDGGIO5.NamPlt.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.LEDGGIO5.NamPlt.d.sValue:= 'LEDGGIO5 demo node';

    fbIED.IEDLD1.LEDGGIO6.NamPlt.vendor.sValue:= sVendor;
    fbIED.IEDLD1.LEDGGIO6.NamPlt.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.LEDGGIO6.NamPlt.d.sValue:= 'LEDGGIO6 demo node';

    fbIED.IEDLD1.LEDGGIO7.NamPlt.vendor.sValue:= sVendor;
    fbIED.IEDLD1.LEDGGIO7.NamPlt.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.LEDGGIO7.NamPlt.d.sValue:= 'LEDGGIO7 demo node';

    fbIED.IEDLD1.LEDGGIO8.NamPlt.vendor.sValue:= sVendor;
    fbIED.IEDLD1.LEDGGIO8.NamPlt.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.LEDGGIO8.NamPlt.d.sValue:= 'LEDGGIO8 demo node';
```

```
    fbIED.IEDLD1.ATCC1.NamPlt.vendor.sValue:= sVendor;
    fbIED.IEDLD1.ATCC1.NamPlt.swRev.sValue:= sSwRev;
    fbIED.IEDLD1.ATCC1.NamPlt.d.sValue:= 'ATCC1 demo node';
    fbIED.IEDLD1.ATCC1.NamPlt.configRev.sValue:= sConfigRev;

    bSuccess:= fbConnection.SetDataSetValuesReq(ipDataSet:=fbIED.IEDLD1.LLN0.DS8, hUser:=0, ipSink:=
0, nInvokeID=>nInvokeID, ipResult=>ipResult);
    state:= SEL(bSuccess, 100, 11);
…
```

```
…
CASE state OF
…
    11:
        IF ipResult <> 0 THEN
            ipResult.Execute();
            IF NOT (bBusy:=ipResult.IsBusy()) THEN
                state:= SEL(ipResult.IsCompleted(), 100(* failed or aborted *), 0(* succeeded *));
            END_IF
        END_IF
…
END_CASE
…
```

7.11 Client - GOOSE Subscriber (GetGoCBDataValues, SetGoCBDataValues)

This example shows how a GOOSE control block on the Server side is controlled with the help of the services GetGoCBDataValues and SetGoCBDataValues. The services enable the reading and/or writing of the attribute values of the GOOSE control block. The sending (publishing) of the GOOSE messages on the Server side is activated when the attribute "GoEna" is set to "TRUE". The sending of the GOOSE messages is deactivated when the attribute "GoEna" is set to "FALSE".

A Subscriber that receives the GOOSE messages on the Client side is also included in the example.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/7280990219.zip

The example described here uses the state machine that is described in the "General Client project structure [▶ 664]" chapter. The States 0, 1, 11 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

In the project tree under the I/O-Device branch you will find a network adapter instance named "GSE (RT Ethernet adapter)". This adapter instance must be configured accordingly, i.e. the I/O configuration must be adapted to the existing hardware and to the target platform on which the project is to run.
A new I/O configuration is also necessary if you change the target platform. This configuration must be done manually in TwinCAT XAE. In addition to the I/O configuration of the network adapter, a link must be established between the network adapter and the PLC function blocks for Goose communication. The link can be used to forward the data received from the network adapter to the instance of the function block: "FB_[IEDName]Gse". In the opposite direction the instance of the function block "FB_[IEDName]Gse" can forward the data to be sent to the network adapter.

Here you can find more information: RT Ethernet adapter Configuration [▶ 35].

7.12 Client - GOOSE Subscriber (without Client-Server communication)

In a Client project, the TwinCAT Telecontrol Configurator also generates a Client block by default (see: "General Client project structure" [▶ 664]). The Client block can then be used, for example, to activate/deactivate the GOOSE Publisher on the Server side via Client-Server services such as "GetGoCBDataValues" or "SetGoCBDataValues" or to configure the GoCBs. In some cases, however, Client-Server communication with the device should be dispensed with and only a "pure" GOOSE Subscriber implemented. Such a Subscriber can automatically start receiving GOOSE messages after the PLC program start. This example shows the implementation of a Subscriber, but without the Client-Server communication part. The parts of the Client project that are not needed have been deliberately removed in this example.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/7282249995.zip

In the Global Variable List, an IED data model function block instance ("fbIED") and one or more GSE function block instances for GOOSE communication and GSE management ("fbIEDGse") are required. This sample uses only one network adapter for GOOSE communication and three GoCBs in the IED data model.

Namespace: TcTelecontrol

Type: Global Variable List (GVL)

```
VAR_GLOBAL
    ipCreator  : I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=2).SetGuiVer(maj
or:=1, minor:=0, build:=93, revision:=10);
    fbIED      : FB_IED_IED;
    fbIEDGse   : FB_IEDGse := (fbAdapter:=(ipIED:=fbIED, settings:=(sMulticastAddr:='01-0C-
CD-01-00-01', eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
END_VAR
```

The "MAIN" program is called cyclically by a TwinCAT task and only needs to call the "P_IEC61850MAIN" program. "P_IEC61850MAIN" in turn calls the GSE function block that is responsible for receiving the GOOSE messages and mapping the received GOOSE data into the IED data model.

```
PROGRAM MAIN
VAR
END_VAR
```

```
P_IEC61850MAIN();
```

```
PROGRAM P_IEC61850MAIN
VAR
END_VAR
```

```
fbIEDGse();
```

In the sample implementation of the GSE function block, the Subscriber process is automatically activated for all three GoCBs after PLC startup. A rising edge at the "bUnsubscribe" variable disables the Subscriber process of the GoCBs.

```
FUNCTION_BLOCK FB_IEDGse IMPLEMENTS I_GseLinkStatusEventSink
VAR_INPUT
    fbAdapter    : FB_GseAdapterClass := (ipLinkStatus:=THIS^);
END_VAR
VAR
    eLinkStatus  : E_GseLinkStatus;
    bSuccess     : BOOL;
    ipError      : I_ServiceErrorClass;
    bSubscribe   : BOOL := TRUE;
    bUnsubscribe : BOOL;
END_VAR
```

```
bSuccess:= fbAdapter.Execute(ipError=>ipError);

IF bSubscribe THEN
    bSubscribe:= FALSE;
    bSuccess:= fbIED.IEDLD1.LLN0.gocb01.Subscriber.Enable(ipAdapter:=fbAdapter, ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb02.Subscriber.Enable(ipAdapter:=fbAdapter, ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb03.Subscriber.Enable(ipAdapter:=fbAdapter, ipError=>ipError);
ELSIF bUnsubscribe THEN
    bUnsubscribe:= FALSE;
    bSuccess:= fbIED.IEDLD1.LLN0.gocb01.Subscriber.Disable(ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb02.Subscriber.Disable(ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb03.Subscriber.Disable(ipError=>ipError);
ELSE
    bSuccess:= fbIED.IEDLD1.LLN0.gocb01.Subscriber.Execute(ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb02.Subscriber.Execute(ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb03.Subscriber.Execute(ipError=>ipError);
END_IF
```

The GSE block implements the "I_GseLinkStatusEventSink" interface. The method: "OnLinkStatusChange" belongs to this interface implementation and is called whenever the status of the network connection (at the network adapter) changes. The PLC application can, for example, query or check the network connection status via "eLinkStatus" variable.

```
METHOD OnLinkStatusChange
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    eStatus   : E_GseLinkStatus;
```

```
END_VAR
VAR
END_VAR
```

```
eLinkStatus:= eStatus;
```

In the project tree under the I/O-Device branch you will find a network adapter instance named "GSE (RT Ethernet adapter)". This adapter instance must be configured accordingly, i.e. the I/O configuration must be adapted to the existing hardware and to the target platform on which the project is to run.
A new I/O configuration is also necessary if you change the target platform. This configuration must be done manually in TwinCAT XAE. In addition to the I/O configuration of the network adapter, a link must be established between the network adapter and the PLC function blocks for Goose communication. The link can be used to forward the data received from the network adapter to the instance of the function block: "FB_[IEDName]Gse". In the opposite direction the instance of the function block "FB_[IEDName]Gse" can forward the data to be sent to the network adapter.

Here you can find more information: RT Ethernet adapter Configuration [▶ 35].

7.13 Client - User defined timestamp clock source

In some applications, a time stamp that originates from an external source (e.g. a GPS clock) should also be used in addition to the setpoint, for example. This example shows the implementation of a user-defined clock/time for timestamping tasks.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/10698182027.zip

The example described here uses the state machine that is described in the "General Client project structure [▶ 664]" chapter. The States 0, 1, 11 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

The TwinCAT IEC 61850 Client implementation uses the interface: "I_ScsmSystemClockEventSink" to query the time for time stamping tasks. It is possible to use a custom function block that implements this special interface and configures the client function block to use the new time source. In the Global Variable List "TcTelecontrol", the new time source is assigned to the client function block by setting the property "ipSystemClock:=fbMyClock". The client function block will then call the method: "OnGetSystemTime" each time it needs a new time stamp from it. The IEC 61850 client-server communication uses time stamps in at least two different formats: UTC time and binary time. For this reason, this method returns the time stamp as output variables in these two formats.

```
VAR_GLOBAL
    ipCreator   : I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=2).SetGuiVer(ma
jor:=1, minor:=0, build:=93, revision:=10);
    fbIED       : FB_IED_IED;
    (* User defined clock (time stamp source) *)
    fbMyClock   : FB_MyClock;
    fbIEDClient : FB_IEDClient := (fbConnection:=(ipIED:=fbIED, ipSystemClock:=fbMyClock, settings:=
(sRemoteHost:='127.0.0.1')));
END_VAR
```

The sample demonstrates a very simple software clock, which was implemented with the help of the RTC_EX function block.

```
FUNCTION_BLOCK FB_MyClock IMPLEMENTS I_ScsmSystemClockEventSink
VAR
    _tT : T_UtcTime:=(secondSinceEpoch:=DT#2021-04-01-00:00:00, quality:=(ClockNotSynchronized:=TRUE
, ClockFailure:=FALSE, LeapSecondsKnown:=FALSE), fractionOfSecond:=[0,0,0]);(* Actual UTC time. *)
    _tB : T_BinaryTime;(* Actual binary-time (EntryTime) *)
    clock : RTC_EX:=(EN:=TRUE, PDT:=DT#2021-04-01-00:00:00, PMSEK:=0);
    refreshTimer : TON:=(IN:=TRUE, PT:=T#1S);
END_VAR
```

```
METHOD FINAL Execute : BOOL
VAR_INPUT
END_VAR
```

```
refreshTimer();
IF Execute:=refreshTimer.Q THEN
    refreshTimer(IN:=FALSE); refreshTimer(IN:=TRUE);
    Update();
END_IF
```

```
METHOD FINAL OnGetSystemTime : BOOL
VAR_INPUT
    ipAA: I_ScsmAssociationClass;(* Application association. If = 0 => optional or unknown. *)
END_VAR
VAR_OUTPUT
    tT: T_UtcTime;(* UTC-time. *)
    tB: T_BinaryTime;(* Binary-time *)
END_VAR
```

```
tT:=_tT;
tB:=_tB;
OnGetSystemTime:=TRUE;
```

```
METHOD FINAL SetClock
VAR_INPUT
    tSet : DT;(* New time to set *)
END_VAR
```

```
clock(EN:=FALSE);
clock(EN:=TRUE, PDT:=tSet);
_tT.quality.ClockNotSynchronized:=FALSE;
Update();
```

```
METHOD FINAL Update
VAR_INPUT
END_VAR
```

```
clock();(* update clock time *)

(* convert to utc-time format *)
_tT.secondSinceEpoch:=clock.CDT;
_tT.fractionOfSecond:=LTIME_TO_UtcTimeFractionofSecond(in:=TIME_TO_LTIME(DWORD_TO_TIME(clock.CMSEK))
);
Accuracy_To_UtcTimeQualityAccuracy(in:=E_UtcTimeAccuracy._03, bAccuracy0=>_tT.quality.Accuracy0, bAc
curacy1=>_tT.quality.Accuracy1, bAccuracy2=>_tT.quality.Accuracy2, bAccuracy3=>_tT.quality.Accuracy3
, bAccuracy4=>_tT.quality.Accuracy4);

(* convert to binary-time format *)
_tB.day:=Date_To_BinaryTime6Day(in:=DT_TO_DATE(clock.CDT));
_tB.timeOfDay:=DT_TO_TOD(clock.CDT) + DWORD_TO_TIME(clock.CMSEK);
```

7.14 Client - Dynamic created DataSets (CreateDataSet, DeleteDataSet)

This sample shows the use of the "CreateDataSetReq" and "DeleteDataSet" methods of the client function block.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/11311587339.zip

The example described here uses the state machine that is described in the "General Client project structure [▶ 664]" chapter. The States 0, 1, 11 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

General information about the dynamic DataSets

The instances of the static DataSets, which are always available during runtime, are either configured by the user in TwinCAT Telecontrol Configurator or imported from an ICD file. The PLC project generation later creates the necessary PLC code for the instantiation of these DataSets. This code also includes the configuration of the DataSet members and the assignment of the DataSet to a logical node in the IEC 61850 data model. The configuration of the static DataSets must not be changed at runtime of the PLC program and the DataSets can also not be deleted.

Sometimes, however, you want to create a DataSet dynamically and use it only for a short time (e.g. as long as the communication connection is up). During this time you may want to monitor some measured values with the help of a report and when the connection is no longer present then remove/delete this DataSet from the data model again.

With the help of the client methods: "CreateDataSetReq" and "DeleteDataSetReq" DataSets can be created, configured and deleted dynamically (at runtime). The dynamically created DataSets can be configured as either "persistent" or "non-persistent" DataSets. The "non-persistent" DataSets are permanently assigned to a client-server connection and only exist as long as the connection is not interrupted. The "non-persistent"

DataSets of a client-server connection are not visible for another client-server connection. I.e. a client cannot see or delete the "non-persistent" DataSets of another client. A non-persistent DataSet can only be deleted by the client that created it or is automatically deleted when that client is disconnected.

The "persistent" DataSets, on the other hand, are visible to all clients and can also be deleted by another client at a later time. However, this is not recommended as the creator client may not notice and may try to continue accessing this DataSet. The "persistent" DataSets are not automatically deleted when the connection is terminated and remain in the data model until they are explicitly deleted.

A dynamically created "persistent" or "non-persistent" DataSet, which is used, for example, by a report control block, cannot be deleted. I.e. as long as the control block Attribute: "DatSet" references this DataSet, this DataSet cannot be deleted. If you want to delete such a DataSet, then the attribute "DatSet" must reference another DataSet or the "DatSet" reference must be deleted (e.g. an empty string deletes the "DatSet" reference).

In a client project and after a TwinCAT restart or a reset of the TwinCAT PLC, all "persistent" and "non-persistent" DataSets on the client side are automatically deleted. On the server side, however, the "persistent" DataSets may remain because they could not be explicitly deleted.

The static DataSets are basically also "persistent" DataSets. They differ from the dynamically generated DataSets in that they cannot be deleted.

For each DataSet that you want to create dynamically at a later time (at runtime), you need an instance of the function block: "FB_AcsiCommonDataSetClass". These instances are manually added to the PLC project beforehand during PLC programming. However, they are not yet linked to the client data model. The PLC application can then later use these instances at runtime to create the dynamic DataSets on the server side. Only then are these instances also linked to the data model on the client side. The DataSet members can also be preconfigured. The method "CreateDataSetReq" needs as first parameter: "ipDataSet" an interface pointer to such a preconfigured DataSet object. The second parameter: "ipLogicalNode" determines whether a "persistent" or "non-persistent" DataSet should be created. Only the "persistent" DataSets are linked to a logical node. In this case, this parameter must be valid. For a "non-persistent" DataSet this parameter is zero. On success, a dynamic DataSet is created on the server side and the preconfigured DataSet instance is linked to the data model on the client side. The method "DeleteDataSetReq" needs as first parameter: "ipDataSet" an interface pointer to a DataSet object (which was already added to the data model before). If successful, the DataSet is then deleted from the data model on the server side and the link between the DataSet and the data model is also deleted on the client side.

Sample project

In the zip archive there is an ICD configuration file. This file can be used, for example, to simulate a server using third-party software or to create a new or modified TwinCAT project with TwinCAT Telecontrol Configurator.

In the FB_IEDClient function block and State 10, several IF instructions are listed that activate commands (method calls) of the client function block. Each command in the IF statement is activated by a rising edge at one of the Boolean variables. The Boolean variables are defined in the declaration part of the FB_IEDClient function block. For test purposes, the user can set the values of the Boolean variables to "TRUE" in the online view and in this way activate the command execution. After that, the system switches to state 11, where the method "ipResult.Execute()" is called in further PLC cycles until the command processing has been completed. This is the case if "ipResult.IsBusy()" returns FALSE.

Following the successful execution of a command, the state machine is set to State 0. If several of the Boolean variables are set to "TRUE", then the topmost IF statement is processed first with the respective command. Some of the Boolean variables are listed in the tables below. In addition, they contain the method names and a description of the function.

The sample uses two preconfigured instances of the "FB_AcsiCommonDataSetClass" function block. The instance: "fbNonPersistent" to create a "non-persistent" DataSet and the instance: "fbPersistent" to create a "persistent" DataSet. The names of the DataSets have already been preconfigured to "NonPersistentDataSet" and "PersistentDataSet" in the declaration part using the "sObjectName" property.

```
…
fbNonPersistent: FB_AcsiCommonDataSetClass:=(sObjectName:='NonPersistentDataSet');
fbPersistent: FB_AcsiCommonDataSetClass:=(sObjectName:='PersistentDataSet');
…
```

In the IF statement, just before enabling the "CreateDataSetReq" method and if not already done, the DataSet members are added to the DataSets. However, this can also be outsourced to a separate routine which is called, for example, once at program start.

```
…
ELSIF bCreateDataSet_NonPersistent THEN
    bCreateDataSet_NonPersistent:= FALSE;
    IF fbNonPersistent.nMembers = 0 THEN
        bSuccess:= fbNonPersistent.AddMember(ipData:=fbIED.IEDLD1.XCBR1.Pos.stVal, eFc:=E_AcsiFc.ST_
);
        bSuccess:= fbNonPersistent.AddMember(ipData:=fbIED.IEDLD1.LLN0.Mod_.stVal, eFc:=E_AcsiFc.ST_
);
    END_IF
    bSuccess:= fbConnection.CreateDataSetReq(ipDataSet:=fbNonPersistent, ipLogicalNode:=0, hUser:=0,
 ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
    state:= SEL(bSuccess, 100, 11);
ELSIF bCreateDataSet_Persistent THEN
    bCreateDataSet_Persistent:= FALSE;
    IF fbPersistent.nMembers = 0 THEN
        bSuccess:= fbPersistent.AddMember(ipData:=fbIED.IEDLD1.MMXU1.TotW.mag.f, eFc:=E_AcsiFc.MX);
        bSuccess:= fbPersistent.AddMember(ipData:=fbIED.IEDLD1.LLN0.Beh.stVal, eFc:=E_AcsiFc.ST_);
    END_IF
    bSuccess:= fbConnection.CreateDataSetReq(ipDataSet:=fbPersistent, ipLogicalNode:=fbIED.IEDLD1.LL
N0, hUser:=0, ipSink:=0, nInvokeID=>nInvokeID, ipResult=>ipResult);
    state:= SEL(bSuccess, 100, 11);
…
```

Variable name	Method name	Description
bCreateDataSet_NonPersistent	CreateDataSetReq	A rising edge at this variable enables the command to create a dynamic "non-persistent" DataSet with the name: "NonPersistentDataSet" (object reference: '@NonPersistent').
bCreateDataSet_Persistent	CreateDataSetReq	A rising edge at this variable enables the command to create a dynamic "persistent" DataSet with the name: "PersistentDataSet" (object reference: 'IEDLD1/LLN0.PersistentDataSet'). On success, the new DataSet is associated with the logical node "LLN0".
bDeleteDataSet_NonPersistent	DeleteDataSetReq	A rising edge at this variable enables the command to delete a dynamic "non-persistent" DataSet with the name: "NonPersistentDataSet".
bDeleteDataSet_Persistent	DeleteDataSetReq	A rising edge at this variable enables the command to delete a dynamic "persistent" DataSet with the name: "PersistentDataSet".
bEnable_urcb101_NonPersistent	urcb101.Client.EnableReq	A rising edge at this variable enables the report control block instance for unbuffered reports: "urcb101". The reports should transfer the DataSet members of the previously dynamically created "non-persistent" DataSet: "fbNonPersistent". Internally, the attribute "DatSet" of the report control block is set accordingly (the string references the dynamically created DataSet).
bEnable_urcb101_Persistent	urcb101.Client.EnableReq	A rising edge at this variable enables the report control block instance for unbuffered reports "urcb101". The reports should transfer the DataSet members of the previously dynamically created "persistent" DataSet: "fbPersistent". Internally, the attribute "DatSet" of the report control block is set accordingly (the string references the dynamically created DataSet).
bEnable_urcb101_DS1	urcb101.Client.EnableReq	A rising edge at this variable enables the report control block instance for unbuffered reports "urcb101". The reports should transfer the DataSet members of the static DataSet: "DS1". Internally, the attribute "DatSet" of the report control block is set accordingly (the string references the static DataSet).
bDisable_urcb101	urcb101.Client.DisableReq	A rising edge at this variable disables the report control block instance for unbuffered reports "urcb101". The transmission of the reports is then stopped.
bSetDataValues_urcb101_DatSet	SetDataValuesReq	As long as "DatSet" references a dynamically created DataSet, this DataSet cannot be deleted again. To be able to delete a dynamically created DataSet, it must not be used by any control block and must not be referenced by the "DatSet". A rising edge at this variable describes the attribute value: "IEDLD1/LLN0.RP.urcb101.DatSet" with an empty string.
bGetDataSetDirectory_NonPersistent	GetDataSetDirectoryReq	A rising edge at this variable enables the command to read the DataSet member configuration of the previously dynamically created "non-persistent" DataSet: "fbNonPersistent". The DataSet members are enumerated.
bGetDataSetDirectory_Persistent	GetDataSetDirectoryReq	A rising edge at this variable enables the command to read the DataSet member configuration of the previously dynamically created "persistent" DataSet: "fbPersistent". The DataSet members are enumerated.
bGetDataSetDirectory_DS1	GetDataSetDirectoryReq	A rising edge at this variable enables the command to read the DataSet member configuration of the static DataSet: "DS1". The DataSet members are enumerated.

Small test

After the program start, when the connection to the server is established, first write the value TRUE into the Boolean variable: "bCreateDataSet_NonPersistent". On success, a new dynamic "non-persistent" DataSet is created. This DataSet can then be used by the report control block instance for unbuffered reports. If you write the value TRUE in the variable "bEnable_urcb101_NonPersistent" then you enable the sending of the reports with the "non-persistent" DataSet member data. Stop sending the reports by writing the value TRUE in the variable: "bDisable_urcb101". However, the dynamic "non-persistent" DataSet is still used by the

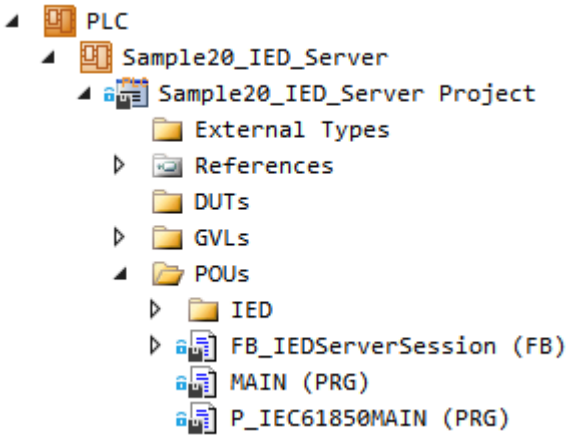
attribute: "DatSet" of the report control block instance "urcb101" and cannot be deleted yet. Write the value TRUE into the Boolean variable "bSetDataValues_urcb101_DatSet". Now the dynamic "non-persistent" DataSet can be deleted. Set the value TRUE in the variable: "bDeleteDataSet_NonPersistent".

7.15 General Server - Project structure

All sample Server projects also follow a general structure, similar to the sample Client projects. In contrast to the client, the connection at the server is not actively established on the server side. After the program start, the server waits for a connection request from a client and accepts or rejects it. In addition, it must be possible for the server to establish connections to multiple clients.

This results in a different server project structure than for the client. The basic structure of the TwinCAT IEC 61850 Server sample projects is based on the PLC project structure automatically generated by the TwinCAT Telecontrol Configurator. The TwinCAT Telecontrol Configurator in turn uses the "TwinCAT XAE Project (XML format)" as a template when generating the solution. The data models communicated in the samples reflect different IEC 61850 servers. They differ from sample to sample and are enclosed with the source code as ICD files. The ICD files can also be used by third-party software for simulation purposes.

Structure of a TwinCAT IEC 61850 PLC Server project:



The generated solution name (unless specified otherwise) corresponds to the TwinCAT Telecontrol Configurator project name. The automatically generated TwinCAT PLC project name (unless specified otherwise) on the other hand has the following structure: "[Project name]_[IEDName]_Server".

As standard, every sample project has a "DUTs", a "GVLs" and a "POUs" folder. A Global Variable List (GVL) with the name "TcTelecontrol" is stored in the "GVLs" folder. The following function blocks are instantiated and initialized in this Global Variable List (see source code below):

- A server function block instance of the type: "FB_iec61850ServerClass" (server configuration settings and connection management).
- At least one instance of the server session function block of the type: "FB_[IEDName]ServerSession[1..n]" (connection management of a single connection and data exchange with a client). At this point, manually add additional instances of the Server Session function block if you want the server to communicate simultaneously with more than one client.
- An IED data model function block instance of the type: "FB_IED_[IEDName]".
- Optional (depending on the GOOSE publisher configuration): one or more function block instances of the type: "FB_[IEDName]Gse" for GOOSE communication and GSE management.

In addition, the code version used during code generation and the version of the TwinCAT Telecontrol Configurator used are also stored there.

Namespace: TcTelecontrol

Type: Global Variable List (GVL)

```
VAR_GLOBAL
    ipCreator                : I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=2
).SetGuiVer(major:=1, minor:=1, build:=94, revision:=1);
    fb[IEDName]              : FB_IED_[IEDName];
    fb[IEDName]Server        : FB_iec61850ServerClass := (ipIED:=fb[IEDName], settings:=(bEnable:=T
```

```
RUE, sLocalHost:='127.0.0.1'));
    fb[IEDName]ServerSession1 : FB_[IEDName]ServerSession := (fbConnection:=(ipServer:=fb[IEDName]Se
rver, settings:=(bEnable:=TRUE)));
    fb[IEDName]Gse            : FB_[IEDName]Gse := (fbAdapter:=(ipIED:=fb[IEDName], settings:=(sMult
icastAddr:='01-0C-CD-01-00-00', eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
END_VAR
```

In the "POUs" folder there is a further folder: "[IEDName]", which contains the entire hierarchical structure of the IED data model as function blocks. This folder also contains the IED data model function block that is instantiated in the Global Variable List and that has already been mentioned. The server session function block: "FB_[IEDName]ServerSession", which implements the establishment of the connection and the data exchange with an IEC 61850 Client, is located on the same level.

The TwinCAT PLC project contains a "MAIN" program as standard. This is called cyclically by a TwinCAT task and in turn calls the program "P_IEC61850MAIN". The program "P_IEC61850MAIN" encapsulates the call of the server, server session and the optional Gse function block, separates the IEC 61850 communication from the rest of the PLC machine program.

```
PROGRAM MAIN
VAR
END_VAR
```

```
P_IEC61850MAIN();
```

```
PROGRAM P_IEC61850MAIN
VAR
END_VAR
```

```
fb[IEDName]Server.Execute();
fb[IEDName]ServerSession1();
fb[IEDName]Gse();
```

In the FB_[IEDName]ServerSession function block, there is a state machine, the basic states of which are used in every server sample (see source code below). The server session function block is responsible for establishing connections and exchanging data with a single client. If there are multiple simultaneous client connections, multiple instances of this function block are required and instantiated.

State 0 (Init state): the state machine is in this state as soon as the PLC program has been started. Commands for the management of a client-server connection are handled here (and in State 1). Normally, the server session function block remains in this state and waits for a connection request from a client. The server then establishes a connection as soon as the connection request is detected. This happens automatically and does not require any further commands from the PLC application. If the server has established a connection to the client, the state machine switches to the data transmission state (State 10).

It is also possible to close an already established connection. This is controlled via two Boolean variables. When set, these variables enable the corresponding commands (in this case these are once-only method calls on the server session function block).

- _bAbort: calls the method "AbortReq", which activates the command to abort the server connection to the client.
- _bDisconnect: calls the method "ReleaseReq", which activates the command for the controlled release of an existing server connection to the client. As a rule, however, it is the client that should terminate its active connection to the server.

The methods listed above require longer than one PLC cycle for their execution. For this reason the state machine switches to a wait state (State 1), in which the termination of the activated command is awaited.

State 1 (Wait State): in this state, the server session function block waits until the command processing for the management of the server client connection is no longer busy. As long as the connection is being released or interrupted, the state machine is in State 1. The state machine is reset to State 0 (Init State) as soon as the command has been successfully executed.

State 10 (Data exchange): if the state machine is in this state, then the server connection to the client has already been successfully established. The server session function block is ready for data transmission to the server. The server data transmission to the client takes place automatically in the background and does not have to be initiated by the PLC application. It is the client that triggers a data transmission in the server by a request.

The state machine switches between states 0 and 10 when there is an active connection. The state machine is reset to State 0 in order to react to changes in the client-server connection status and to handle them in State 0.

State 100 (Error state): as soon as an error occurs during the activation or processing of a command, the state machine is set to State 100. The error is logged here and the state machine reset to State 0.

```
FUNCTION_BLOCK FB_[IEDName]ServerSession
VAR_INPUT
    fbConnection     : FB_iec61850ConnectionClass := (ipAbortInd:=THIS^, ipAssociateInd:=THIS^, ipRe
leaseInd:=THIS^);
END_VAR
VAR
    _bAbort          : BOOL;
    _bDisconnect     : BOOL;
    state            : BYTE;
    eState           : E_AsyncEnvironmentState;
    bBusy            : BOOL;
    bSuccess         : BOOL;
    ipResult         : I_AsyncServiceResultClass;
    sLastErrorResult : T_MaxString;
    fbAbortReason    : FB_ServiceErrorClass := (stError:=SUCCESS_EVENT);
    sLastAbortReason : T_MaxString;
END_VAR
```

```
fbConnection.Execute();
eState:= fbConnection.eState;

CASE state OF
    0:
        IF _bAbort THEN
            _bAbort:= FALSE;
            bSuccess:= fbConnection.AbortReq(ipReason:=fbAbortReason, ipSink:=0, ipResult=>ipResult)
;
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Established AND _bDisconnect THEN
            _bDisconnect:= FALSE;
            bSuccess:= fbConnection.ReleaseReq(ipSink:=0, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Established THEN
            state:= 10;
        END_IF
        _bDisconnect:= FALSE;
    1:
        IF ipResult <> 0 THEN
            ipResult.Execute();
            IF NOT (bBusy:=ipResult.IsBusy()) THEN
                state:= SEL(ipResult.IsCompleted(), 100, 0);
            END_IF
        END_IF
    10:
        state:= 0;
    100:
        state:= 0;
        IF ipResult <> 0 THEN
            sLastErrorResult:= ipResult.Dump();
        END_IF
END_CASE
```

GOOSE Publisher (optional)

TwinCAT Telecontrol Configurator can also generate the PLC code for a GOOSE publisher in a server project during PLC code generation (see code sample below). However, this is only possible if the user has previously created the GOOSE components such as GoCBs (goose control blocks) in the TwinCAT Telecontrol Configurator or imported them from an SCL file (e.g. ICD file). By default, a function block with the name: "FB_[IEDName]Gse" is instantiated during code generation and added to the Global Variable List "TcTelecontrol". This function block establishes the connection between a network adapter of the TwinCAT control computer, the IED data model and the GOOSE configuration in the GoCBs. The GoCBs are instantiated in the IED data model (usually in LLN0). Each GoCB has a function block subelement with the name: "Publisher". The "Publisher" method calls can be used to start or stop publishing from the PLC code. By default, publishing is started at PLC program start for all GoCBs. This is controlled by the "bStart" variable initialized with "TRUE". Publishing can be stopped for all GoCBs via a rising edge at the "bStop" variable. Publisher commands issued through these methods are executed immediately, without wait cycles or further states required to complete command processing. The "Publisher" described here reads the configuration and does update the status of the GoCB (attribute "GoEna" is set to "TRUE" or "FALSE", for example), but it does not use the client-server services such as "SetGoCBValues" or "GetGoCBValues" to start or stop the Publisher. This means that the generated code already implements a publisher that can be started or stopped, for example, in the first PLC cycle or from the PLC code at any time. The required GoCB configuration settings (GoCB attribute values) can be made via initialization values. However, the GoCBs

can already be configured in the TwinCAT Telecontrol Configurator. The initialization values are then automatically generated and assigned during code generation. If the Publisher has been started and the configuration of the GoCB and the network adapter shows a match, then the Publisher immediately starts sending the first GOOSE frames (including frame repetitions). The "Update" method has a special significance. Every time it is called, the "Publisher" immediately sends a new GOOSE frame (and automatically new frame repetitions). This means that the application can set all relevant GOOSE dataset data in the IED data model first and then initiate the sending of the changed dataset data with the "Update" method call. The "Execute" method must be called cyclically the rest of the time. It is responsible for sending the frame repetitions and updating the status information in the GoCB.

```
FUNCTION_BLOCK FB_[IEDName]Gse IMPLEMENTS I_GseLinkStatusEventSink
VAR_INPUT
    fbAdapter   : FB_GseAdapterClass := (ipLinkStatus:=THIS^);
END_VAR
VAR
    eLinkStatus : E_GseLinkStatus;
    bSuccess    : BOOL;
    ipError     : I_ServiceErrorClass;
    bStart      : BOOL := TRUE;
    bStop       : BOOL;
    bUpdate     : BOOL;
END_VAR
```

```
bSuccess:= fbAdapter.Execute(ipError=>ipError);

IF bStart THEN
    bStart:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Publisher.Start(ipAdapter:=fbAdapter, ipError=>ipError
);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Publisher.Start(ipAdapter:=fbAdapter, ipError=>ipError
);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Publisher.Start(ipAdapter:=fbAdapter, ipError=>ipError
);
ELSIF bStop THEN
    bStop:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Publisher.Stop(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Publisher.Stop(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Publisher.Stop(ipError=>ipError);
ELSIF bUpdate THEN
    bUpdate:= FALSE;
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Publisher.Update(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Publisher.Update(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Publisher.Update(ipError=>ipError);
ELSE
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb01.Publisher.Execute(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb02.Publisher.Execute(ipError=>ipError);
    bSuccess:= fb[IEDName].IEDLD1.LLN0.gocb03.Publisher.Execute(ipError=>ipError);
END_IF
```

The GSE block implements the "I_GseLinkStatusEventSink" interface. The method: "OnLinkStatusChange" belongs to this interface implementation and is called whenever the status of the network connection (at the network adapter) changes. The PLC application can, for example, query or check the network connection status via "eLinkStatus" variable.

```
METHOD OnLinkStatusChange
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    eStatus   : E_GseLinkStatus;
END_VAR
VAR
END_VAR
```

```
eLinkStatus:= eStatus;
```

In the project tree under the I/O-Device branch you will find a network adapter instance named "GSE (RT Ethernet adapter)". This adapter instance must be configured accordingly, i.e. the I/O configuration must be adapted to the existing hardware and to the target platform on which the project is to run.
A new I/O configuration is also necessary if you change the target platform. This configuration must be done manually in TwinCAT XAE. In addition to the I/O configuration of the network adapter, a link must be established between the network adapter and the PLC function blocks for Goose communication. The link can be used to forward the data received from the network adapter to the instance of the function block: "FB_[IEDName]Gse". In the opposite direction the instance of the function block "FB_[IEDName]Gse" can forward the data to be sent to the network adapter.

Here you can find more information: RT Ethernet adapter Configuration [▶ 35].

7.16 Server - Base Sample Project

This example shows the basic TwinCAT implementation of the Client-Server communication of a TwinCAT IEC 61850 Server. The GOOSE components and reporting are not included in this example in the interests of simplicity. These and other functionalities are covered in separate examples. The Server implemented here has a default host address: "127.0.0.1" and a default port number: 102. These values can be adapted in the PLC sample code. The PLC development environment will output a warning with a note about this when translating. Double-clicking on this message will take you to the corresponding location in the PLC code.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/10698184715.zip

The example described here uses the state machine that is described in the "General Server project structure [▶ 700]" chapter. The States 0, 1, 10 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

7.17 Server – GOOSE Publisher (GetGoCBDataValues, SetGoCBDataValues)

In this example, the GOOSE control block can be controlled by Clients using the GetGoCBDataValues and SetGoCBDataValues services.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/10698177035.zip

The example described here uses the state machine that is described in the "General Server project structure [▶ 700]" chapter. The States 0, 1, 10 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

In the project tree under the I/O-Device branch you will find a network adapter instance named "GSE (RT Ethernet adapter)". This adapter instance must be configured accordingly, i.e. the I/O configuration must be adapted to the existing hardware and to the target platform on which the project is to run.
A new I/O configuration is also necessary if you change the target platform. This configuration must be done manually in TwinCAT XAE. In addition to the I/O configuration of the network adapter, a link must be established between the network adapter and the PLC function blocks for Goose communication. The link can be used to forward the data received from the network adapter to the instance of the function block: "FB_[IEDName]Gse". In the opposite direction the instance of the function block "FB_[IEDName]Gse" can forward the data to be sent to the network adapter.

Here you can find more information: RT Ethernet adapter Configuration [▶ 35].

7.18 Server - GOOSE Publisher (without Client-Server communication)

In a Server project, the TwinCAT Telecontrol Configurator also generates a Server block and one or more instances of the Server session block by default (see: "General Server project structure" [▶ 700]). The Server and ServerSession block can, for example, ensure that the Publisher can be activated/deactivated from the Client via Client-Server services such as "GetGoCBDataValues" or "SetGoCBDataValues" or that the GoCBs can be configured. In some cases, however, Client-Server communication with the device should be omitted and only a "pure" GOOSE Publisher should be implemented. Such a Publisher can automatically start sending GOOSE messages after the PLC program start. This example shows the implementation of a Publisher, but without the Client-Server communication part. The parts of the Server project that are not needed have been deliberately removed in this example.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/7280992907.zip

Namespace: TcTelecontrol

Type: Global Variable List (GVL)

In the Global Variable List, an IED data model function block instance ("fbIED") and one or more GSE function block instances for GOOSE communication and GSE management ("fbIEDGse") are required. This sample uses only one network adapter for GOOSE communication and three GoCBs in the IED data model.

```
VAR_GLOBAL
    ipCreator : I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=2).SetGuiVer(majo
r:=1, minor:=0, build:=93, revision:=10);
    fbIED     : FB_IED_IED;
    fbIEDGse  : FB_IEDGse := (fbAdapter:=(ipIED:=fbIED, settings:=(sMulticastAddr:='01-0C-
CD-01-00-01', eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
END_VAR
```

The "MAIN" program is called cyclically by a TwinCAT task and only needs to call the "P_IEC61850MAIN" program. "P_IEC61850MAIN" in turn calls the GSE function block responsible for sending the GOOSE messages and mapping the data to be sent from the IED data model in the GOOSE messages.

```
PROGRAM MAIN
VAR
END_VAR
```

```
P_IEC61850MAIN();
```

```
PROGRAM P_IEC61850MAIN
VAR
END_VAR
```

```
fbIEDGse();
```

In the sample implementation of the GSE function block, the Publishing process is automatically started for all three GoCBs after PLC startup. A rising edge at the "bStop" variable stops the Publishing process of the GoCBs. The value changes of some GOOSE data are simulated in the sample. Every 5 s, some values of the GoCB DataSet members are modified and a rising edge is generated at the variable "bUpdate". This rising edge causes the Publishers to generate and send a new GOOSE message. The PLC application does not have to take care of the GOOSE message repetitions. This is done automatically.

```
FUNCTION_BLOCK FB_IEDGse IMPLEMENTS I_GseSystemClockEventSink, I_GseLinkStatusEventSink
VAR_INPUT
    fbAdapter     : FB_GseAdapterClass := (ipSystemClock:=THIS^, ipLinkStatus:=THIS^);
END_VAR
VAR
    eLinkStatus   : E_GseLinkStatus;
    bSuccess      : BOOL;
    ipError       : I_ServiceErrorClass;
    bStart        : BOOL := TRUE;
    bStop         : BOOL;
    bUpdate       : BOOL;

    bSimulation   : BOOL := TRUE;
    tSimulation   : TIME := T#5S;
    fbUpdateTimer : TON;

    bSync         : BOOL := TRUE;
    tSync         : T_UtcTime := String_TO_UtcTime(in:='UT#2019-07-12-12:00:00.000000000|000|3');
    fbClock       : FB_GseSystemClock;
END_VAR
```

```
fbUpdateTimer(IN:=bSimulation, PT:=tSimulation);
IF fbUpdateTimer.Q THEN
    fbUpdateTimer(IN:=FALSE);
    fbUpdateTimer(IN:=bSimulation);

    fbIED.IEDLD1.LEDGGIO1.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO1.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO2.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO2.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO3.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO3.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO4.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO4.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO5.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO5.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO6.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO6.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO7.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO7.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO8.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO8.SPCSO1.stVal.bValue;

    fbIED.IEDLD1.MMXU1.TotW.mag.f.fValue:= fbIED.IEDLD1.MMXU1.TotW.mag.f.fValue + 0.1;

    IF fbIED.IEDLD1.XCBR1.Pos.stVal.eValue = E_AcsiDbpos.On THEN
        fbIED.IEDLD1.XCBR1.Pos.stVal.eValue:= E_AcsiDbpos.Off;
    ELSE
        fbIED.IEDLD1.XCBR1.Pos.stVal.eValue:= E_AcsiDbpos.On;
    END_IF
    fbIED.IEDLD1.XCBR1.Pos.q.OldData:= NOT fbIED.IEDLD1.XCBR1.Pos.q.OldData;
    fbIED.IEDLD1.XCBR1.Pos.t.SecondSinceEpoch:= fbIED.IEDLD1.XCBR1.Pos.t.SecondSinceEpoch + TIME#1S;
```

```
    bUpdate:= TRUE;
END_IF
```

```
IF bSync THEN
    bSync:= FALSE;
    bSuccess:= fbClock.SetToUtcTime(in:=tSync);
ELSE
    fbClock.Execute();
END_IF
```

```
bSuccess:= fbAdapter.Execute(ipError=>ipError);

IF bStart THEN
    bStart:= FALSE;
    bSuccess:= fbIED.IEDLD1.LLN0.gocb01.Publisher.Start(ipAdapter:=fbAdapter, ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb02.Publisher.Start(ipAdapter:=fbAdapter, ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb03.Publisher.Start(ipAdapter:=fbAdapter, ipError=>ipError);
ELSIF bStop THEN
    bStop:= FALSE;
    bSuccess:= fbIED.IEDLD1.LLN0.gocb01.Publisher.Stop(ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb02.Publisher.Stop(ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb03.Publisher.Stop(ipError=>ipError);
ELSIF bUpdate THEN
    bUpdate:= FALSE;
    bSuccess:= fbIED.IEDLD1.LLN0.gocb01.Publisher.Update(ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb02.Publisher.Update(ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb03.Publisher.Update(ipError=>ipError);
ELSE
    bSuccess:= fbIED.IEDLD1.LLN0.gocb01.Publisher.Execute(ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb02.Publisher.Execute(ipError=>ipError);
    bSuccess:= fbIED.IEDLD1.LLN0.gocb03.Publisher.Execute(ipError=>ipError);
END_IF
```

The GSE function block implements the "I_GseSystemClockEventSink" interface. The method: "OnGetSystemTime" belongs to this interface implementation and is called whenever a Publisher needs a new timestamp for a GOOSE message. For example, the PLC application could transfer its own timestamp, perhaps from a GPS clock, to the GOOSE message. In the sample, a simple software demo clock is used for time stamping.

```
METHOD OnGetSystemTime : BOOL
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
END_VAR
VAR_OUTPUT
    tT        : T_UtcTime;
END_VAR
VAR
END_VAR
```

```
OnGetSystemTime:= fbClock.OnGetSystemTime(ipAdapter:=ipAdapter, tT=>tT);
```

The GSE block implements the "I_GseLinkStatusEventSink" interface. The method: "OnLinkStatusChange" belongs to this interface implementation and is called whenever the status of the network connection (at the network adapter) changes. The PLC application can, for example, query or check the network connection status via "eLinkStatus" variable.

```
METHOD OnLinkStatusChange
VAR_INPUT
    ipAdapter : I_GseAdapterClass;
    eStatus   : E_GseLinkStatus;
END_VAR
VAR
END_VAR
```

```
eLinkStatus:= eStatus;
```

In the project tree under the I/O-Device branch you will find a network adapter instance named "GSE (RT Ethernet adapter)". This adapter instance must be configured accordingly, i.e. the I/O configuration must be adapted to the existing hardware and to the target platform on which the project is to run.
A new I/O configuration is also necessary if you change the target platform. This configuration must be done manually in TwinCAT XAE. In addition to the I/O configuration of the network adapter, a link must be established between the network adapter and the PLC function blocks for Goose communication. The link can be used to forward the data received from the network adapter to the instance of the function block: "FB_[IEDName]Gse". In the opposite direction the instance of the function block "FB_[IEDName]Gse" can forward the data to be sent to the network adapter.

Here you can find more information: RT Ethernet adapter Configuration [▶ 35].

7.19 Server - Direct Control with normal security

This sample shows the implementation of ctlModel := 1 (Direct control with normal security, direct-operate) in a server for the following CDCs:

- SPC (controllable single point)
- DPC (controllable double point)
- APC (controllable analogue process value)
- BAC (binary controlled analog process value)
- BSC (binary controlled step position information)
- ENC (controllable enumerated status)
- INC (controllable integer status)
- ISC (integer controlled step position information)

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/10698251403.zip

The example described here uses the state machine that is described in the "General Server project structure [▶ 700]" chapter. The States 0, 1, 10 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

A new "FB_DirectControl" function block has been added. The state machine for the control model is implemented in this. In order for this to be executed for a data object, an "FB_DirectControl" instance must be declared as follows in the "VAR_INPUT" part of the function block of the data object to be controlled:

```
fbControl: FB_DirectControl := (ipCtrl:=THIS^.AddServerJob(ipEvent:=fbControl, ipCompletion:=fbControl));
```

Each of these instances has an "Execute" method that must be called every cycle to monitor the control model. In this sample, this is accomplished in the "FB_IEDServerSession" function block. This method contains the state machine and the monitoring for sending "LastApplError" messages. In addition to the "Execute" method, the "FB_DirectControl" function block has further methods and properties that are required for the control model. These are explained in the following tables.

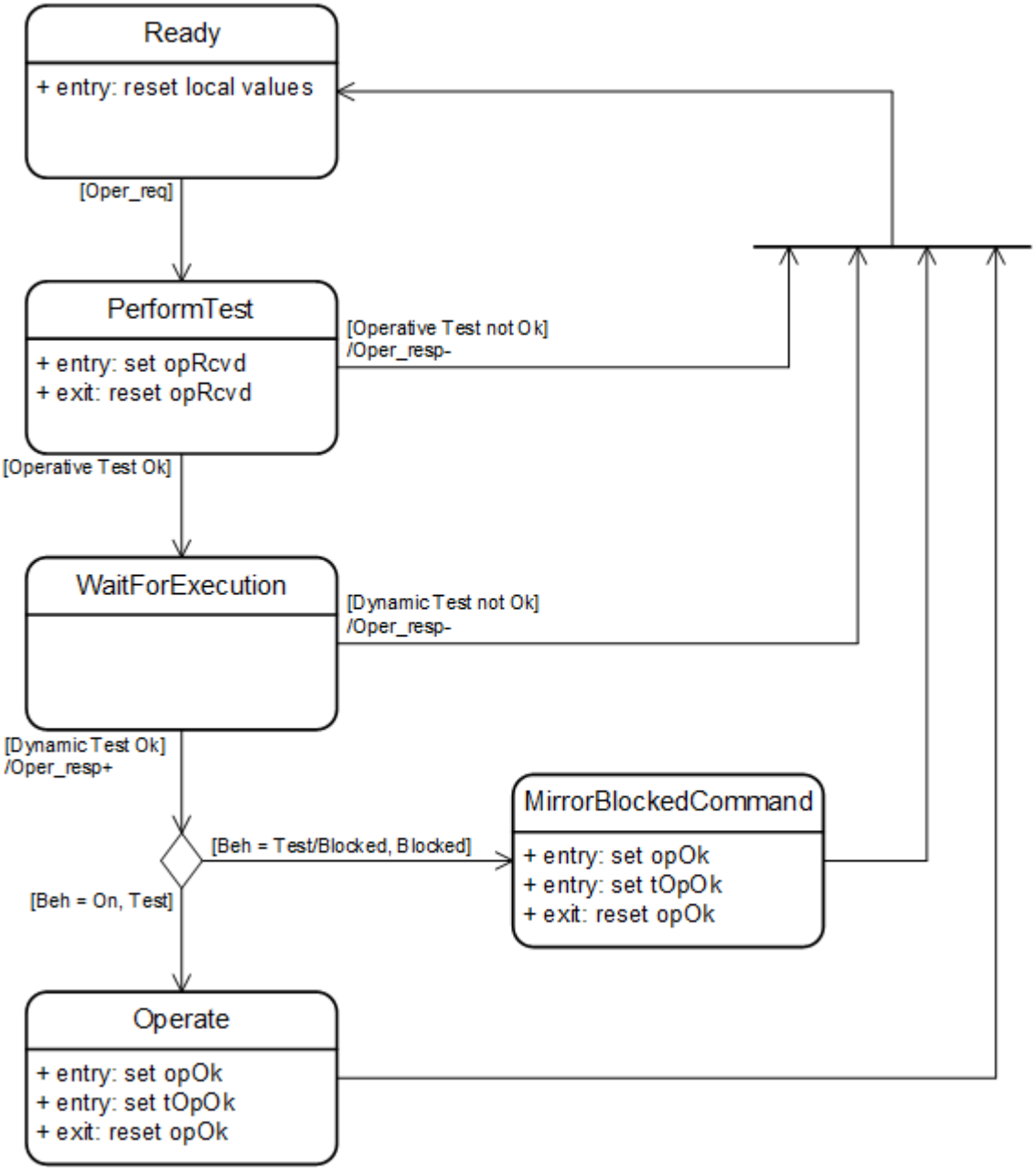
Methods of the "FB_DirectControl" function block

Method name	Description
_CheckCtlVal	In accordance with the controllable CDC of the data object, a check is made to determine whether the value of "ctlVal" to be written is valid. If the value is invalid, a corresponding error code is returned.
_DynamicTest	This method performs the dynamic test. Since this is not needed in this sample, a successful result is always returned. A custom dynamic test can be implemented in this method.
_Init	Initializes the "FB_DirectControl" function block by determining the CDC of the data object to be controlled.
_OperativeTest	This method performs the operational test. The system checks whether the state of the logical node that is superordinate to the data object to be controlled permits control and whether the value to be written is valid. If the test is successful, the address of the client is cached to identify which client triggered the control model. If the test fails, a "LastApplError" message is sent to the client.
_Reset	Resets the cached values for a DirectControl.
_ResetOperValues	Resets the values of the "Oper" data attribute to the configured initial values.
_SendLastApplError	Sends a "LastApplError" message to the Client.
_Set[CDC]Value	A method is added for each controllable CDC, as most CDCs use different types, for example, "APC" -> "REAL" and "SPC" -> "BOOL". As the CDCs use different types, each of these methods also has a different input parameter for the value (e.g. ctlVal). This parameter is used to set the value of the data object to be controlled.
_SetValue	Sets the value of the data object to be controlled according to the value of "ctlVal". Depending on the CDC of the data object, the corresponding "_Set[CDC]Val" method is called for this purpose.
_UpdateTimestampValue	Sets the value of the data attribute "t" of the data object to be controlled to the current system time.
Execute	The state machine for the control model is implemented in this method. The other methods from this function block are called according to the state machine in this method.
OnAbort	Is called when the server disconnects and stops sending the "LastApplError" message.
OnCompletionWaitEvent	This method is called by the server implementation after each write operation. This determines whether an "Operate" request was received. If this is the case, the server implementation is informed that it must wait until the DirectControl is connected before sending the response to the "Operate" request.
OnGetVEvent	This method is part of the "I_AcsiDataAccessEventClass" interface but is not needed for this sample.
OnGetVPreEvent	This method is part of the "I_AcsiDataAccessEventClass" interface but is not needed for this sample.
OnSetVEvent	This method is part of the "I_AcsiDataAccessEventClass" interface but is not needed for this sample.
OnSetVPreEvent	This method is part of the "I_AcsiDataAccessEventClass" interface and is called before each write access to the data object to be controlled. This functionality is used to prevent the "Oper" structure from being overwritten, for example, by another client, during a control operation.

Properties of the "FB_DirectControl" function block

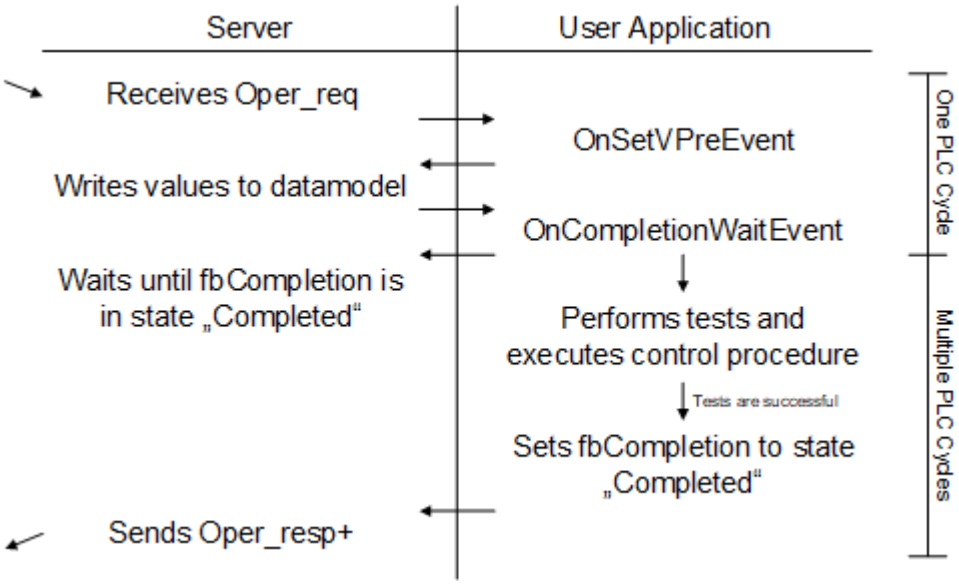
Property name	Description
eType	Specifies the controllable CDC (e.g. "APC").
ipBehVal	Interface pointer to the value of the "Beh" data object of the logical node that is superordinate to the data object to be controlled.
ipCtrl	Interface pointer to the data object to be controlled.
ipStepSize	Optional interface pointer to the "stepSize" data attribute. This can be used for setting the value of a "BAC" data object.

The state machine shown in the figure below, which is implemented in the "Execute" method of the "FB_DirectControl" function block, is used for the DirectControl process. The transitions between the individual states are controlled by Boolean variables. In addition, the Boolean variable "bNewState" is used to detect new states and implement the corresponding behavior when the state changes.



Since there may be several PLC cycles between receiving the "Operate" request and executing it, the server application must be told how long it must wait before sending a response. An instance of the "FB_AcsiServiceResultClass" function block called "fbCompletion" is used for this. When an "Operate" request is received, it is set to the "Busy" state in the "OnCompletionWaitEvent" method and transferred to the Server application as an interface pointer. This now waits until "fbCompletion" is in the "Completed" state

and then sends the response to the "Operate" request. If the "Operate" request is to be rejected directly, this is done in the "OnSetVPreEvent" method. The process of an "Operate" request is shown in the following figure.



This is just an example of the implementation of the DirectControl function in the TwinCAT PLC.

7.20 Server - Direct Control with enhanced security

This sample shows the implementation of ctlModel := 3 (Direct control with enhanced security, direct-operate) in a server for the following CDCs:

- SPC (controllable single point)
- DPC (controllable double point)
- APC (controllable analogue process value)
- BAC (binary controlled analog process value)
- BSC (binary controlled step position information)
- ENC (controllable enumerated status)
- INC (controllable integer status)
- ISC (integer controlled step position information)

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/10698256779.zip

The example described here uses the state machine that is described in the "General Server project structure [▶ 700]" chapter. The States 0, 1, 10 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

A new "FB_DirectControl" function block has been added. The state machine for the control model is implemented in this. In order for this to be executed for a data object, an "FB_DirectControl" instance must be declared as follows in the "VAR_INPUT" part of the function block of the data object to be controlled:

```
fbControl: FB_DirectControl := (ipCtrl:=THIS^.AddServerJob(ipEvent:=fbControl, ipCompletion:=fbControl));
```

Each of these instances has an "Execute" method that must be called every cycle to monitor the control model. In this sample, this is accomplished in the "FB_IEDServerSession" function block. This method contains the state machine and the monitoring for sending "LastApplError" and "CommandTermination" messages. In addition to the "Execute" method, the "FB_DirectControl" function block has further methods and properties that are required for the control model. These are explained in the following tables.

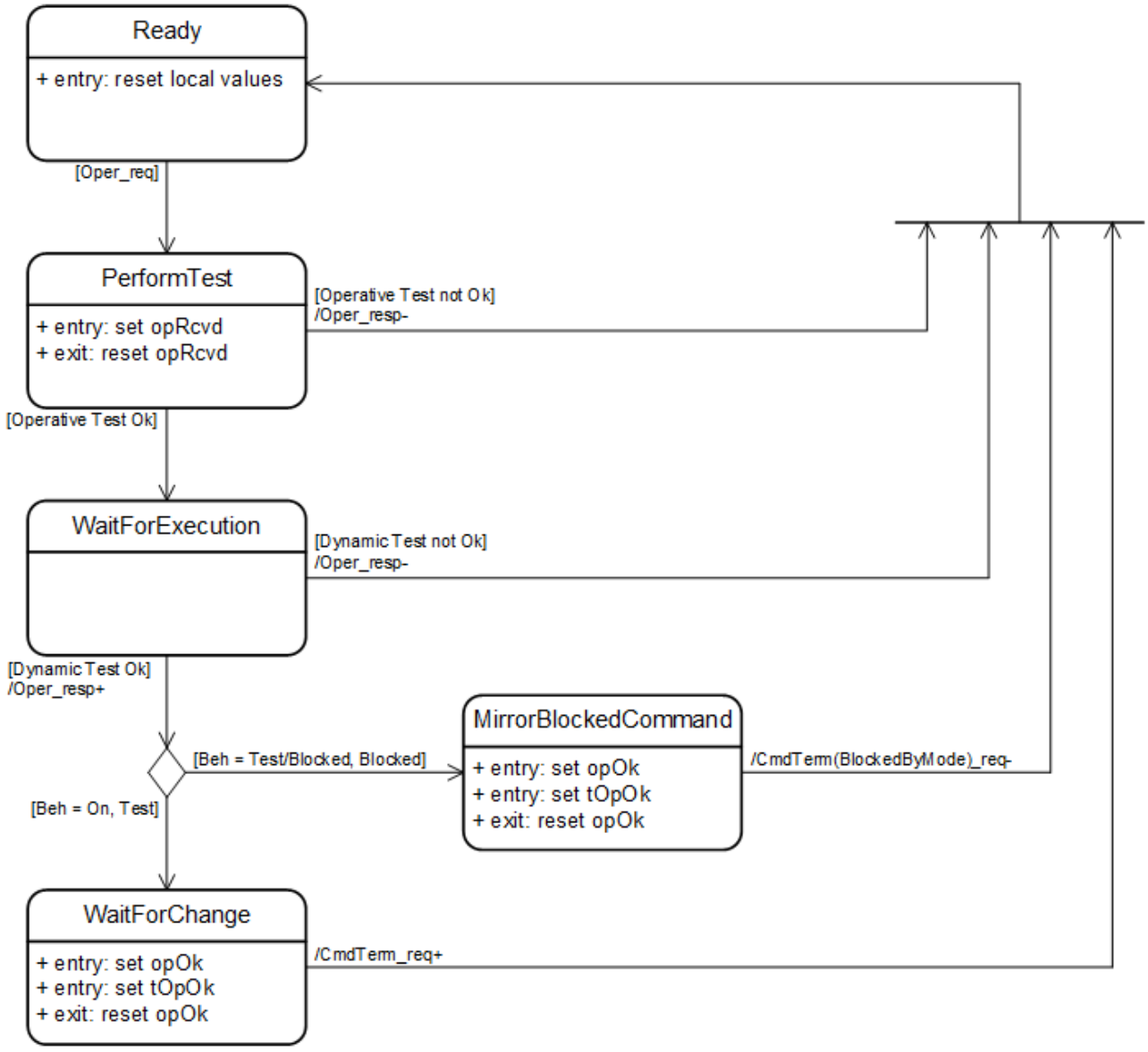
Methods of the "FB_DirectControl" function block

Method name	Description
_CheckCtlVal	In accordance with the controllable CDC of the data object, a check is made to determine whether the value of "ctlVal" to be written is valid. If the value is invalid, a corresponding error code is returned.
_DynamicTest	This method performs the dynamic test. Since this is not needed in this sample, a successful result is always returned. A custom dynamic test can be implemented in this method.
_Init	Initializes the "FB_DirectControl" function block by determining the CDC of the data object to be controlled.
_OperativeTest	This method performs the operational test. The system checks whether the state of the logical node that is superordinate to the data object to be controlled permits control and whether the value to be written is valid. If the test is successful, the address of the client is cached to identify which client triggered the control model. If the test fails, a "LastApplError" message is sent to the client.
_Reset	Resets the cached values for a DirectControl.
_ResetOperValues	Resets the values of the "Oper" data attribute to the configured initial values.
_SendCommandTerminationReqNeg	Sends a negative "CommandTermination" request to the client.
_SendCommandTerminationReqPos	Sends a positive "CommandTermination" request to the client.
_SendLastApplError	Sends a "LastApplError" message to the Client.
_Set[CDC]Value	A method is added for each controllable CDC, as most CDCs use different types, for example, "APC" -> "REAL" and "SPC" -> "BOOL". As the CDCs use different types, each of these methods also has a different input parameter for the value (e.g. ctlVal). This parameter is used to set the value of the data object to be controlled.
_SetValue	Sets the value of the data object to be controlled according to the value of "ctlVal". Depending on the CDC of the data object, the corresponding "_Set[CDC]Val" method is called for this purpose.
_UpdateTimestampValue	Sets the value of the data attribute "t" of the data object to be controlled to the current system time.
Execute	The state machine for the control model is implemented in this method. The other methods from this function block are called according to the state machine in this method.
OnAbort	Is called when the server disconnects and stops sending the "LastApplError" and "CommandTermination" message.
OnCompletionWaitEvent	This method is called by the server implementation after each write operation. This determines whether an "Operate" request was received. If this is the case, the server implementation is informed that it must wait until the DirectControl is connected before sending the response to the "Operate" request.
OnGetVEvent	This method is part of the "I_AcsiDataAccessEventClass" interface but is not needed for this sample.
OnGetVPreEvent	This method is part of the "I_AcsiDataAccessEventClass" interface but is not needed for this sample.
OnSetVEvent	This method is part of the "I_AcsiDataAccessEventClass" interface but is not needed for this sample.
OnSetVPreEvent	This method is part of the "I_AcsiDataAccessEventClass" interface and is called before each write access to the data object to be controlled. This functionality is used to prevent the "Oper" structure from being overwritten, for example, by another client, during a control operation.

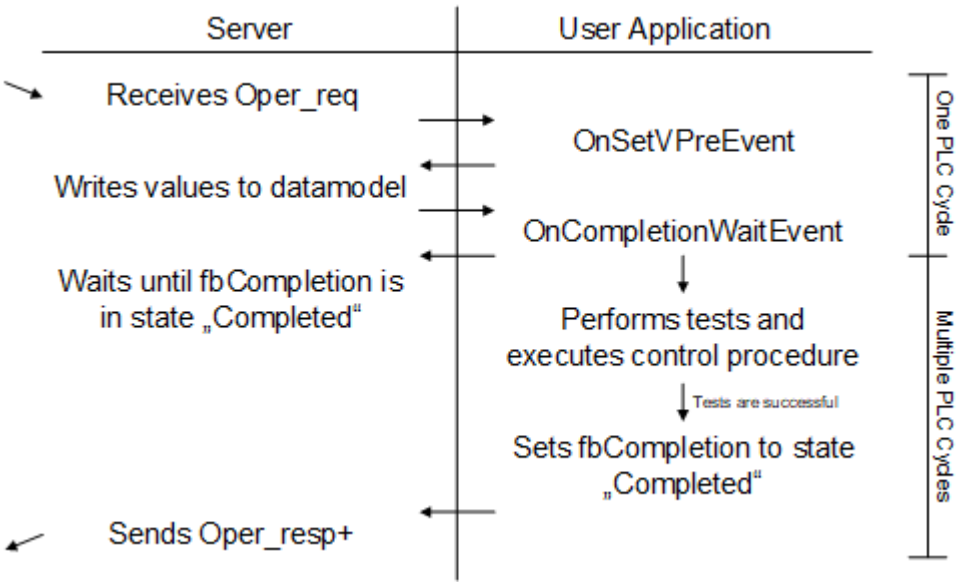
Properties of the "FB_DirectControl" function block

Property name	Description
eType	Specifies the controllable CDC (e.g. "APC").
ipBehVal	Interface pointer to the value of the "Beh" data object of the logical node that is superordinate to the data object to be controlled.
ipCtrl	Interface pointer to the data object to be controlled.
ipStepSize	Optional interface pointer to the "stepSize" data attribute. This can be used for setting the value of a "BAC" data object.

The state machine shown in the figure below, which is implemented in the "Execute" method of the "FB_DirectControl" function block, is used for the DirectControl process. The transitions between the individual states are controlled by Boolean variables. In addition, the Boolean variable "bNewState" is used to detect new states and implement the corresponding behavior when the state changes.



Since there may be several PLC cycles between receiving the "Operate" request and executing it, the server application must be told how long it must wait before sending a response. An instance of the "FB_AcsiServiceResultClass" function block called "fbCompletion" is used for this. When an "Operate" request is received, it is set to the "Busy" state in the "OnCompletionWaitEvent" method and transferred to the Server application as an interface pointer. This now waits until "fbCompletion" is in the "Completed" state and then sends the response to the "Operate" request. If the "Operate" request is to be rejected directly, this is done in the "OnSetVPreEvent" method. The process of an "Operate" request is shown in the following figure.



This is just an example of the implementation of the DirectControl function in the TwinCAT PLC.

7.21 Server - SBO Control with normal security

This sample shows the implementation of ctlModel := 2 (SBO control with normal security, operate-once or operate-many) in a server for the following CDCs:

- SPC (controllable single point)
- DPC (controllable double point)
- APC (controllable analogue process value)
- BAC (binary controlled analog process value)
- BSC (binary controlled step position information)
- ENC (controllable enumerated status)
- INC (controllable integer status)
- ISC (integer controlled step position information)

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/10698254091.zip

The example described here uses the state machine that is described in the "General Server project structure [▶ 700]" chapter. The States 0, 1, 10 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

A new "FB_SboControl" function block has been added. The state machine for the control model is implemented in this. In order for this to be executed for a data object, an "FB_SboControl" instance must be declared as follows in the "VAR_INPUT" part of the function block of the data object to be controlled:

```
fbControl: FB_SboControl := (ipCtrl:=THIS^.AddServerJob(ipEvent:=fbControl, ipCompletion:=fbControl)
);
```

Each of these instances has an "Execute" method that must be called every cycle to monitor the control model. In this sample, this is accomplished in the "FB_IEDServerSession" function block. This method contains the state machine and the monitoring for sending "LastApplError" messages. In addition to the "Execute" method, the "FB_SboControl" function block has further methods and properties that are required for the control model. These are explained in the following tables.

Methods of the "FB_SboControl" function block

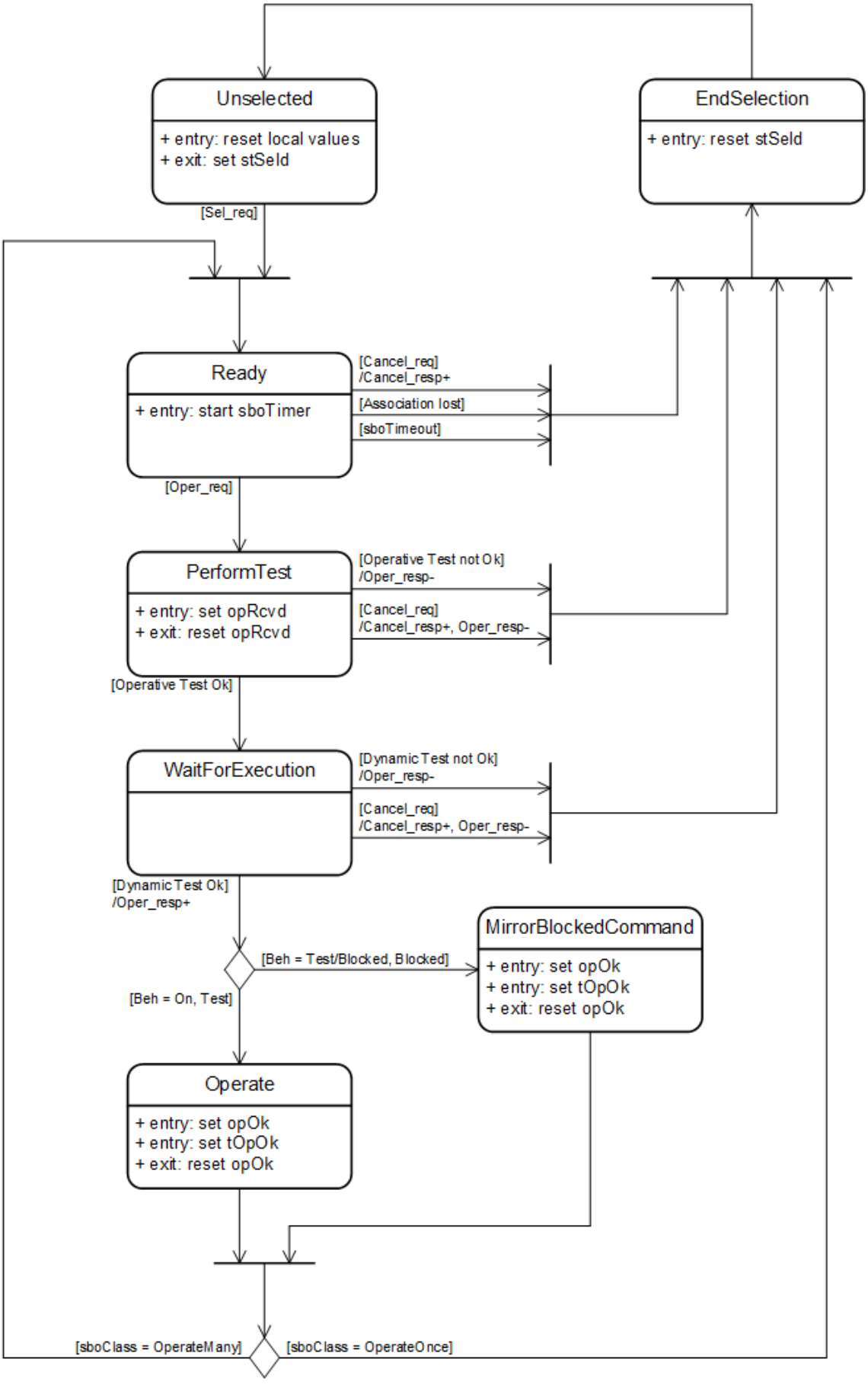
Method name	Description
_CheckCancelReq	Is called when a "Cancel" request is received and checks if it is valid. If this is invalid, a corresponding "LastApplError" message is sent.
_CheckCtlVal	In accordance with the controllable CDC of the data object, a check is made to determine whether the value of "ctlVal" to be written is valid. If the value is invalid, a corresponding error code is returned.
_DynamicTest	This method performs the dynamic test. Since this is not needed in this sample, a successful result is always returned. A custom dynamic test can be implemented in this method.
_Init	Initializes the "FB_DirectControl" function block by determining the CDC of the data object to be controlled.
_OperativeTest	This method performs the operational test. The system checks whether the state of the logical node that is superordinate to the data object to be controlled permits control and whether the value to be written is valid. If the test is successful, the address of the client is cached to identify which client triggered the control model. If the test fails, a "LastApplError" message is sent to the client.
_Reset	Resets the values cached for an SBOControl.
_ResetCancelValues	Resets the values of the "Cancel" data attribute to the configured initial values.
_ResetOperValues	Resets the values of the "Oper" data attribute to the configured initial values.
_SendLastApplError	Sends a "LastApplError" message to the Client.
_Set[CDC]Value	A method is added for each controllable CDC, as most CDCs use different types, for example, "APC" -> "REAL" and "SPC" -> "BOOL". As the CDCs use different types, each of these methods also has a different input parameter for the value (e.g. ctlVal). This parameter is used to set the value of the data object to be controlled.
_SetValue	Sets the value of the data object to be controlled according to the value of "ctlVal". Depending on the CDC of the data object, the corresponding "_Set[CDC]Val" method is called for this purpose.
_UpdateTimestampValue	Sets the value of the data attribute "t" of the data object to be controlled to the current system time.
Execute	The state machine for the control model is implemented in this method. The other methods from this function block are called according to the state machine in this method.
OnAbort	Is called when the server disconnects and stops sending the "LastApplError" message.
OnCompletionWaitEvent	This method is called by the server implementation after each write operation. This determines whether an "Operate" request was received. If this is the case, the server implementation is informed that it must wait until the SBOControl is connected before sending the response to the "Operate" request.
OnGetVEvent	This method is part of the "I_AcsiDataAccessEventClass" interface and is called after each read access to the data object to be controlled. This functionality is used to reset the value of the data attribute "SBO" after a "Select" request.
OnGetVPreEvent	This method is part of the "I_AcsiDataAccessEventClass" interface and is called before each read access to the data object to be controlled. This functionality is used to review "Select" requests and accept or reject them accordingly.
OnSetVEvent	This method is part of the "I_AcsiDataAccessEventClass" interface but is not needed for this sample.

Method name	Description
OnSetVPreEvent	This method is part of the "I_AcsiDataAccessEventClass" interface and is called before each write access to the data object to be controlled. This functionality is used to prevent overwriting of the "Oper" and "Cancel" structure as well as the "SBO" data attribute during a control operation, for example, by another client.

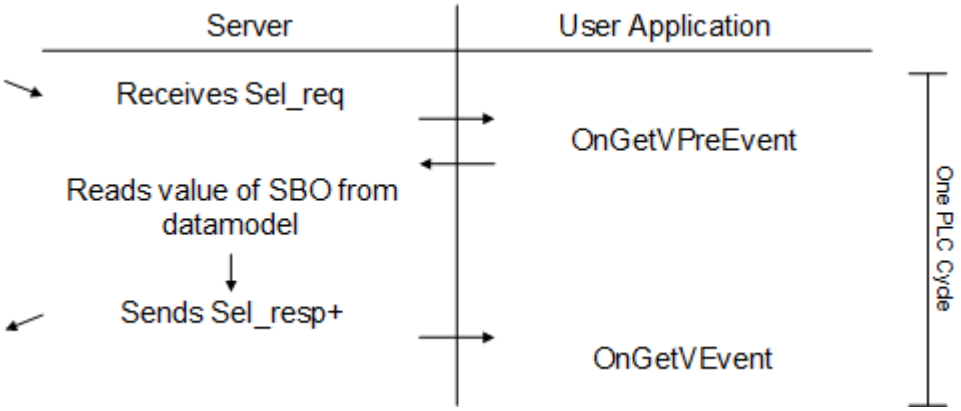
Properties of the "FB_SboControl" function block

Property name	Description
eType	Specifies the controllable CDC (e.g. "APC").
ipBehVal	Interface pointer to the value of the "Beh" data object of the logical node that is superordinate to the data object to be controlled.
ipCtrl	Interface pointer to the data object to be controlled.
ipStepSize	Optional interface pointer to the "stepSize" data attribute. This can be used for setting the value of a "BAC" data object.

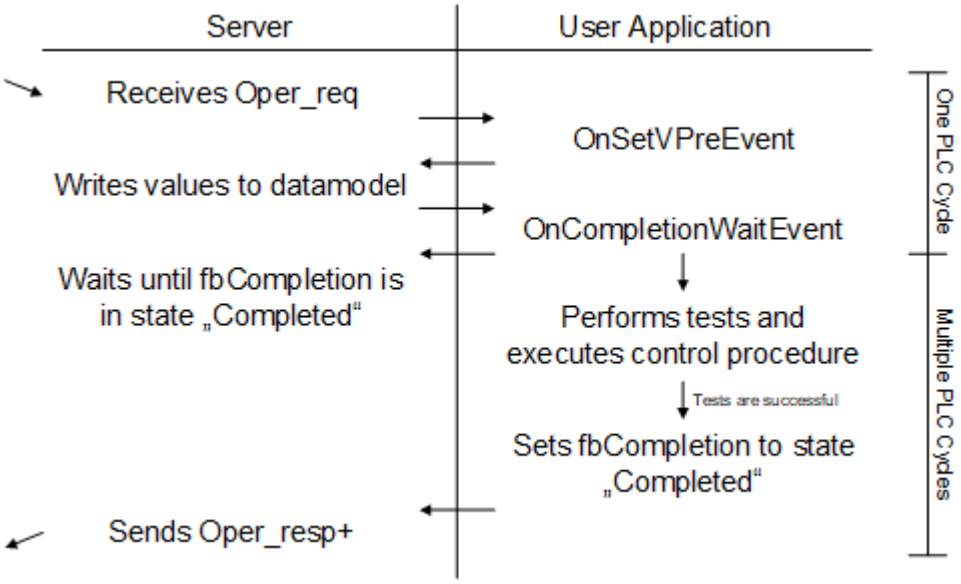
The state machine shown in the figure below, which is implemented in the "Execute" method of the "FB_SboControl" function block, is used for the SBOControl process. The transitions between the individual states are controlled by Boolean variables. In addition, the Boolean variable "bNewState" is used to detect new states and implement the corresponding behavior when the state changes.


Unselected
+ entry: reset local values
+ exit: set stSeld
[Sel_req]
EndSelection
+ entry: reset stSeld
Ready
+ entry: start sboTimer
[Cancel_req]
/Cancel_resp+
[Association lost]
[sboTimeout]
[Oper_req]
PerformTest
+ entry: set opRcvd
+ exit: reset opRcvd
[Operative Test not Ok]
/Oper_resp-
[Cancel_req]
/Cancel_resp+, Oper_resp-
[Operative Test Ok]
WaitForExecution
[Dynamic Test not Ok]
/Oper_resp-
[Cancel_req]
/Cancel_resp+, Oper_resp-
[Dynamic Test Ok]
/Oper_resp+
[Beh = Test/Blocked, Blocked]
MirrorBlockedCommand
+ entry: set opOk
+ entry: set tOpOk
+ exit: reset opOk
[Beh = On, Test]
Operate
+ entry: set opOk
+ entry: set tOpOk
+ exit: reset opOk
[sboClass = OperateMany]
[sboClass = OperateOnce]

Since a "Select" request is a read request on the "SBO" data attribute, it is checked in the "OnGetVPreEvent" method. In this method, the "Select" request can be accepted by setting the value of "SBO" to the reference to the "SBO" data attribute. To reject the request, the value is set to an empty string. After sending the response, the "OnGetVEvent" method is called, in which the value of the data attribute can be reset. The process of a successful "Select" request is shown in the following figure.



Since there may be several PLC cycles between receiving the "Operate" request and executing it, the server application must be told how long it must wait before sending a response. An instance of the "FB_AcsiServiceResultClass" function block called "fbCompletion" is used for this. When an "Operate" request is received, it is set to the "Busy" state in the "OnCompletionWaitEvent" method and transferred to the Server application as an interface pointer. This now waits until "fbCompletion" is in the "Completed" state and then sends the response to the "Operate" request. If the "Operate" request is to be rejected directly, this is done in the "OnSetVPreEvent" method. The process of an "Operate" request is shown in the following figure.



This is just an example of the implementation of the SBOControl function in the TwinCAT PLC.

7.22 Server - SBO Control with enhanced security

This sample shows the implementation of ctlModel := 4 (SBO control with enhanced security, operate-once or operate-many) in a server for the following CDCs:

- SPC (controllable single point)
- DPC (controllable double point)
- APC (controllable analogue process value)
- BAC (binary controlled analog process value)

- BSC (binary controlled step position information)
- ENC (controllable enumerated status)
- INC (controllable integer status)
- ISC (integer controlled step position information)

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/10698259467.zip

The example described here uses the state machine that is described in the "General Server project structure [▶ 700]" chapter. The States 0, 1, 10 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

A new "FB_SboControl" function block has been added. The state machine for the control model is implemented in this. In order for this to be executed for a data object, an "FB_SboControl" instance must be declared as follows in the "VAR_INPUT" part of the function block of the data object to be controlled:

```
fbControl: FB_SboControl := (ipCtrl:=THIS^.AddServerJob(ipEvent:=fbControl, ipCompletion:=fbControl)
);
```

Each of these instances has an "Execute" method that must be called every cycle to monitor the control model. In this sample, this is accomplished in the "FB_IEDServerSession" function block. This method contains the state machine and the monitoring for sending "LastApplError" and "CommandTermination" messages. In addition to the "Execute" method, the "FB_SboControl" function block has further methods and properties that are required for the control model. These are explained in the following tables.

Methods of the "FB_SboControl" function block

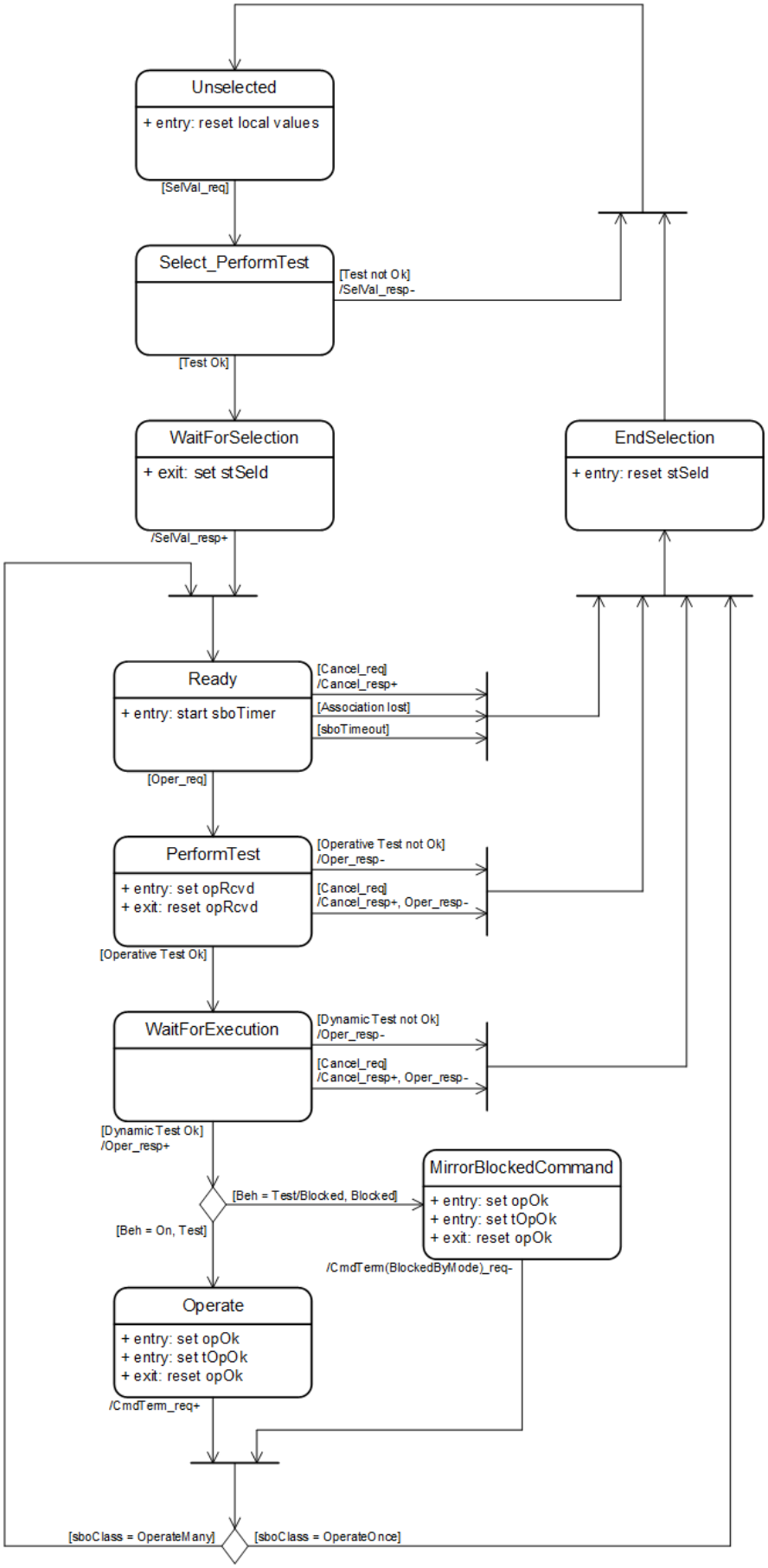
Method name	Description
_CompareCtlVal	Checks whether two values of "ctlVal" are equal in accordance with the controllable CDC of the data object to be controlled. This method is used to check if the "SelectWithValue" and "Operate" requests contain identical values.
_CheckCancelReq	Is called when a "Cancel" request is received and checks if it is valid. If this is invalid, a corresponding "LastApplError" message is sent.
_CheckCtlVal	In accordance with the controllable CDC of the data object, a check is made to determine whether the value of "ctlVal" to be written is valid. If the value is invalid, a corresponding error code is returned.
_DynamicTest	This method performs the dynamic test. Since this is not needed in this sample, a successful result is always returned. A custom dynamic test can be implemented in this method.
_Init	Initializes the "FB_DirectControl" function block by determining the CDC of the data object to be controlled.
_OperativeTest	This method performs the operational test. The system checks whether the state of the logical node that is superordinate to the data object to be controlled permits control and whether the value to be written is valid. If the test is successful, the address of the client is cached to identify which client triggered the control model. If the test fails, a "LastApplError" message is sent to the client.
_Reset	Resets the values cached for an SBOControl.
_ResetCancelValues	Resets the values of the "Cancel" data attribute to the configured initial values.
_ResetSBOwValues	Resets the values of the "SBOw" data attribute to the configured initial values.
_ResetOperValues	Resets the values of the "Oper" data attribute to the configured initial values.
_SendCommandTerminationReqNeg	Sends a negative "CommandTermination" request to the client.
_SendCommandTerminationReqPos	Sends a positive "CommandTermination" request to the client.
_SendLastApplError	Sends a "LastApplError" message to the Client.
_Set[CDC]Value	A method is added for each controllable CDC, as most CDCs use different types, for example, "APC" -> "REAL" and "SPC" -> "BOOL". As the CDCs use different types, each of these methods also has a different input parameter for the value (e.g. ctlVal). This parameter is used to set the value of the data object to be controlled.
_SetValue	Sets the value of the data object to be controlled according to the value of "ctlVal". Depending on the CDC of the data object, the corresponding "_Set[CDC]Val" method is called for this purpose.
_Test	This method perform a test on the data transferred with the "SelectWithValue" request. The system checks whether the state of the logical node that is superordinate to the data object to be controlled permits control and whether the value to be written is valid. If the test is successful, the address of the client is cached to identify which client triggered the control model. If the test fails, a "LastApplError" message is sent to the client.
_UpdateTimestampValue	Sets the value of the data attribute "t" of the data object to be controlled to the current system time.
Execute	The state machine for the control model is implemented in this method. The other methods from this function block are called according to the state machine in this method.
OnAbort	Is called when the server disconnects and stops sending the "LastApplError" message.

Method name	Description
OnCompletionWaitEvent	This method is called by the server implementation after each write operation. This determines whether an "Operate" or "SelectWithValue" request was received. If this is the case, the server implementation is informed that it must wait until the SBOControl is connected before sending the response to the "Operate" or "SelectWithValue" request.
OnGetVEvent	This method is part of the "I_AcsiDataAccessEventClass" interface but is not needed for this sample.
OnGetVPreEvent	This method is part of the "I_AcsiDataAccessEventClass" interface but is not needed for this sample.
OnSetVEvent	This method is part of the "I_AcsiDataAccessEventClass" interface but is not needed for this sample.
OnSetVPreEvent	This method is part of the "I_AcsiDataAccessEventClass" interface and is called before each write access to the data object to be controlled. This functionality is used to prevent the "Oper", "SBOw" and "Cancel" structure from being overwritten, for example, by another client, during a control operation.

Properties of the "FB_SboControl" function block

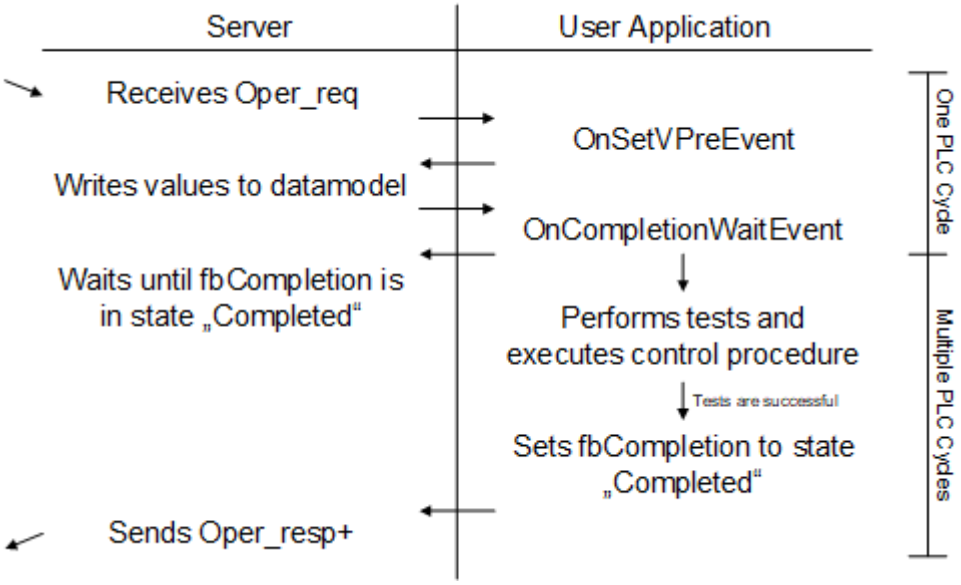
Property name	Description
eType	Specifies the controllable CDC (e.g. "APC").
ipBehVal	Interface pointer to the value of the "Beh" data object of the logical node that is superordinate to the data object to be controlled.
ipCtrl	Interface pointer to the data object to be controlled.
ipStepSize	Optional interface pointer to the "stepSize" data attribute. This can be used for setting the value of a "BAC" data object.

The state machine shown in the figure below, which is implemented in the "Execute" method of the "FB_SboControl" function block, is used for the SBOControl process. The transitions between the individual states are controlled by Boolean variables. In addition, the Boolean variable "bNewState" is used to detect new states and implement the corresponding behavior when the state changes.



Since there may be several PLC cycles between receiving the "Operate" request and executing it, the server application must be told how long it must wait before sending a response. An instance of the "FB_AcsiServiceResultClass" function block called "fbCompletion" is used for this. When an "Operate"

request is received, it is set to the "Busy" state in the "OnCompletionWaitEvent" method and transferred to the Server application as an interface pointer. This now waits until "fbCompletion" is in the "Completed" state and then sends the response to the "Operate" request. If the "Operate" request is to be rejected directly, this is done in the "OnSetVPreEvent" method. The process of an "Operate" request is shown in the following figure.



Since the "SelectWithValue" request is a write request, the same sequence as for "Operate" requests described in the previous paragraph is used to process it.



This is just an example of the implementation of the SBOControl function in the TwinCAT PLC.

7.23 Server - User defined timestamp clock source

This sample shows the implementation of a user-defined clock/time for time stamping tasks in a TwinCAT IEC 61850 server project. The spontaneous time required by the server can be specified from an external source (e.g. from a GPS clock). Among other things, a current time is required when sending reports.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/10698262155.zip

The example described here uses the state machine that is described in the "General Server project structure [▶ 700]" chapter. The States 0, 1, 10 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

The TwinCAT IEC 61850 server implementation uses the interface: "I_ScsmSystemClockEventSink" to query the time for time stamping tasks. It is possible to use a custom function block that implements this special interface and configures the server function block to use the new time source. In the Global Variable List "TcTelecontrol", the new time source is assigned to the server function block by setting the property "ipSystemClock:=fbMyClock". The server function block will then call the method: "OnGetSystemTime" each time it needs a new time stamp from it. In our sample, this is the case when reporting is active and a new report is to be sent. For demonstration purposes, the external time source is also queried when the attribute: "IEDLD1/MMXU1.TotW" or "IEDLD1/MMXU1.TotW.t" is read by the client. Whenever client reads the data, the method: "OnGetVPreEvent" of the function block: "FB_DO_IED_LD1_MMXU1_TotW" is called. In the method implementation the method: "GetSystemTime(tT=>tT)" is called via the interface pointer: "ipAssociation". This time request is forwarded to our custom clock.

The client-server communication uses time stamps in at least two different formats: UTC time and binary time. For this reason, this method returns the time stamp as output variables in these two formats.

```
VAR_GLOBAL
    ipCreator: I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=3).SetGuiVer(major
:=1, minor:=1, build:=94, revision:=4);
```

```
    fbIED: FB_IED_IED;
    fbMyClock : FB_MyClock;
    fbIEDServer: FB_iec61850ServerClass := (ipIED:=fbIED, ipSystemClock:=fbMyClock, settings:=(bEnab
le:=TRUE, sLocalHost:='127.0.0.1'));
    fbIEDServerSession1: FB_IEDServerSession := (fbConnection:=(ipServer:=fbIEDServer, settings:=(bE
nable:=TRUE)));
END_VAR
```

The sample demonstrates a very simple software clock, which was implemented with the help of the RTC_EX function block.

```
FUNCTION_BLOCK FB_MyClock IMPLEMENTS I_ScsmSystemClockEventSink
VAR
    _tT : T_UtcTime:=(secondSinceEpoch:=DT#2021-04-01-00:00:00, quality:=(ClockNotSynchronized:=TRUE
, ClockFailure:=FALSE, LeapSecondsKnown:=FALSE), fractionOfSecond:=[0,0,0]);(* Actual UTC time. *)
    _tB : T_BinaryTime;(* Actual binary-time (EntryTime) *)
    clock : RTC_EX:=(EN:=TRUE, PDT:=DT#2021-04-01-00:00:00, PMSEK:=0);
    refreshTimer : TON:=(IN:=TRUE, PT:=T#1S);
END_VAR
```

```
METHOD FINAL Execute : BOOL
VAR_INPUT
END_VAR
```

```
refreshTimer();
IF Execute:=refreshTimer.Q THEN
    refreshTimer(IN:=FALSE); refreshTimer(IN:=TRUE);
    Update();
END_IF
```

```
METHOD FINAL OnGetSystemTime : BOOL
VAR_INPUT
    ipAA: I_ScsmAssociationClass;(* Application association. If = 0 => optional or unknown. *)
END_VAR
VAR_OUTPUT
    tT: T_UtcTime;(* UTC-time. *)
    tB: T_BinaryTime;(* Binary-time *)
END_VAR
```

```
tT:=_tT;
tB:=_tB;
OnGetSystemTime:=TRUE;
```

```
METHOD FINAL SetClock
VAR_INPUT
    tSet : DT;(* New time to set *)
END_VAR
```

```
clock(EN:=FALSE);
clock(EN:=TRUE, PDT:=tSet);
_tT.quality.ClockNotSynchronized:=FALSE;
Update();
```

```
METHOD FINAL Update
VAR_INPUT
END_VAR
```

```
clock();(* update clock time *)

(* convert to utc-time format *)
_tT.secondSinceEpoch:=clock.CDT;
_tT.fractionOfSecond:=LTIME_TO_UtcTimeFractionofSecond(in:=TIME_TO_LTIME(DWORD_TO_TIME(clock.CMSEK))
);
Accuracy_To_UtcTimeQualityAccuracy(in:=E_UtcTimeAccuracy._03, bAccuracy0=>_tT.quality.Accuracy0, bAc
curacy1=>_tT.quality.Accuracy1, bAccuracy2=>_tT.quality.Accuracy2, bAccuracy3=>_tT.quality.Accuracy3
, bAccuracy4=>_tT.quality.Accuracy4);

(* convert to binary-time format *)
_tB.day:=Date_To_BinaryTime6Day(in:=DT_TO_DATE(clock.CDT));
_tB.timeOfDay:=DT_TO_TOD(clock.CDT) + DWORD_TO_TIME(clock.CMSEK);
```

7.24 Server - Unbuffered Reporting

This sample shows the Unbuffered Reporting implementation in a TwinCAT IEC61850 server project. The UrCBs (unbuffered report control blocks) and Datasets required for reporting can be configured in the TwinCAT Telecontrol Configurator or imported from an existing SCL file (e.g. ICD file). The instances of the UrCBs and the Datasets are created automatically during code generation (usually below LLN0). The initial values of the UrCB attributes can already be configured in the TwinCAT Telecontrol Configurator.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/10698264843.zip

The example described here uses the state machine that is described in the "General Server project structure [▶ 700]" chapter. The States 0, 1, 10 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

The implementation of the unbuffered reporting functionality in a TwinCAT IEC 61850 server project is practically a "black box". This means that the complete functionality is already encapsulated in a function block and the application only has to call this function block. Each UrCB has such a function block subelement with the name: "Server". The PLC application only has to call the method: "Execute" at the "Server" block cyclically. This method is responsible for sending the reports and updating the status information in the associated UrCB (see code below).

The reports can be configured either on the server side directly on the UrCB or on the client side via client-server services (GetUrCBDataValues and SetUrCBDataValues). The IED data to be communicated via reporting are defined with the help of the datasets. Multiple data references (Dataset Members) can be configured in a dataset.

The sample contains a simple simulation of some Dataset member values. If the variable "bSimulation" is set to TRUE, then the values are modified every 2 seconds. How does the UrCB Server implementation recognize that a value has been modified and a report is to be generated? This is done through write access to the attribute value (for example: bValue, tValue, iValue, eValue, nValue properties). However, a new report is not generated immediately for each individual write access to an attribute value. This would be undesirable. The "Server" implementation marks the modified dataset members internally as modified. A report is not generated and sent until the "Execute" method is called.

Please note that no reports are generated when writing or forcing values in TwinCAT online mode. In this way, the "Server" implementation cannot detect when a value has been modified. Integrity reports and General Interrogation reports are automatically generated and sent in accordance with the configuration on the UrCB. The sending of these reports does not have to be triggered in a special way in the PLC application. The cyclic call of the server "Execute" method also handles the sending of these reports.

The time behavior of the report messages can be configured independently for each UrCB server instance via two properties "bReplaceBufferedMX" and "bOverwriteSameCycleChanges". For more information on how these properties work, see the description of the sample: Server-Buffered Reporting [▶ 729].

```
PROGRAM MAIN
VAR
    bSimulation                  : BOOL:=TRUE; (* Enables/disables simulation of data update/
change *)
    tSimulation                  : TIME:=T#2S; (* Cycle time of simulated data update/change *)
    fbTimer                      : TON;
    tT                           : T_UtcTime;

    bReplaceBufferedMX           : BOOL; (* Toggles bReplaceBufferedMX property of urcb201 *)
    bOverwriteSameCycleChanges   : BOOL; (* Toggles bOverwriteSameCycleChanges property of urcb301
*)
END_VAR
```

```
P_IEC61850MAIN();

fbIED.IEDLD1.LLN0.urcb201.Server.bReplaceBufferedMX:= bReplaceBufferedMX;
fbIED.IEDLD1.LLN0.urcb301.Server.bOverwriteSameCycleChanges:= bOverwriteSameCycleChanges;

fbTimer(IN:=bSimulation, PT:=tSimulation);
IF fbTimer.Q THEN(* Simulate server data update *)
    fbTimer(IN:=FALSE);
    fbTimer(IN:=bSimulation);

    fbIEDServer.GetSystemTime(ipAA:=0, tT=>tT);

    (* Simulate "IEDLD1/LLN0.DS1" member value change *)
    fbIED.IEDLD1.LEDGGIO1.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO1.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO1.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO2.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO2.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO2.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO3.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO3.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO3.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO4.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO4.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO4.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO5.SPCSO1.t.tValue:= tT;
```

```
    fbIED.IEDLD1.LEDGGIO5.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO5.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO6.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO6.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO6.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO7.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO7.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO7.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO8.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO8.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO8.SPCSO1.stVal.bValue;

    (* Simulate "IEDLD1/LLN0.DS2" member value change *)
    fbIED.IEDLD1.MMXU1.TotW.t.tValue:= tT;
    fbIED.IEDLD1.MMXU1.TotW.mag.f.fValue:= fbIED.IEDLD1.MMXU1.TotW.mag.f.fValue + 0.1;

    (* Simulate "IEDLD1/LLN0.DS3" member value change *)
    fbIED.IEDLD1.XCBR1.Pos.t.tValue:= tT;
    IF fbIED.IEDLD1.XCBR1.Pos.stVal.eValue = E_AcsiDbpos.On THEN
        fbIED.IEDLD1.XCBR1.Pos.stVal.eValue:= E_AcsiDbpos.Off;
    ELSE
        fbIED.IEDLD1.XCBR1.Pos.stVal.eValue:= E_AcsiDbpos.On;
    END_IF
    fbIED.IEDLD1.XCBR1.Pos.q.OldData:= NOT fbIED.IEDLD1.XCBR1.Pos.q.OldData;
    fbIED.IEDLD1.XCBR1.Pos.q.eValidity:= SEL(fbIED.IEDLD1.XCBR1.Pos.q.OldData, E_AcsiQualityValidity
.Good, E_AcsiQualityValidity.Questionable);
END_IF
```

```
FUNCTION_BLOCK FB_IEDServerSession IMPLEMENTS I_ScsmAbortIndEventSink, I_ScsmAssociateIndEventSink,
I_ScsmReleaseIndEventSink
VAR_INPUT
    fbConnection     : FB_iec61850ConnectionClass := (ipAbortInd:=THIS^, ipAssociateInd:=THIS^, ipRe
leaseInd:=THIS^);
END_VAR
VAR
    _bAbort          : BOOL;
    _bDisconnect     : BOOL;
    state            : BYTE;
    eState           : E_AsyncEnvironmentState;
    bBusy            : BOOL;
    bSuccess         : BOOL;
    ipResult         : I_AsyncServiceResultClass;
    sLastErrorResult : T_MaxString;
    fbAbortReason    : FB_ServiceErrorClass := (stError:=SUCCESS_EVENT);
    sLastAbortReason : T_MaxString;
    bSimulation      : BOOL:=TRUE;
    tSimulation      : TIME:=T#2S;
    fbTimer          : TON;
    tT               : T_UtcTime;
END_VAR
```

```
fbConnection.Execute();
eState:= fbConnection.eState;

fbTimer(IN:=bSimulation, PT:=tSimulation);
IF fbTimer.Q THEN(* Simulate server data update *)
    fbTimer(IN:=FALSE);
    fbTimer(IN:=bSimulation);

    fbConnection.GetSystemTime(ipAA:=0, tT=>tT);

    (* Simulate "IEDLD1/LLN0.DS1" member value change *)
    fbIED.IEDLD1.LEDGGIO1.SPCSO1.t.tValue:=tT;
    fbIED.IEDLD1.LEDGGIO1.SPCSO1.stVal.bValue:=NOT fbIED.IEDLD1.LEDGGIO1.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO2.SPCSO1.t.tValue:=tT;
    fbIED.IEDLD1.LEDGGIO2.SPCSO1.stVal.bValue:=NOT fbIED.IEDLD1.LEDGGIO2.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO3.SPCSO1.t.tValue:=tT;
    fbIED.IEDLD1.LEDGGIO3.SPCSO1.stVal.bValue:=NOT fbIED.IEDLD1.LEDGGIO3.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO4.SPCSO1.t.tValue:=tT;
    fbIED.IEDLD1.LEDGGIO4.SPCSO1.stVal.bValue:=NOT fbIED.IEDLD1.LEDGGIO4.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO5.SPCSO1.t.tValue:=tT;
    fbIED.IEDLD1.LEDGGIO5.SPCSO1.stVal.bValue:=NOT fbIED.IEDLD1.LEDGGIO5.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO6.SPCSO1.t.tValue:=tT;
    fbIED.IEDLD1.LEDGGIO6.SPCSO1.stVal.bValue:=NOT fbIED.IEDLD1.LEDGGIO6.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO7.SPCSO1.t.tValue:=tT;
    fbIED.IEDLD1.LEDGGIO7.SPCSO1.stVal.bValue:=NOT fbIED.IEDLD1.LEDGGIO7.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO8.SPCSO1.t.tValue:=tT;
    fbIED.IEDLD1.LEDGGIO8.SPCSO1.stVal.bValue:=NOT fbIED.IEDLD1.LEDGGIO8.SPCSO1.stVal.bValue;

    (* Simulate "IEDLD1/LLN0.DS2" member value change *)
    fbIED.IEDLD1.MMXU1.TotW.t.tValue:=tT;
    fbIED.IEDLD1.MMXU1.TotW.mag.f.fValue:=fbIED.IEDLD1.MMXU1.TotW.mag.f.fValue + 0.1;

    (* Simulate "IEDLD1/LLN0.DS3" member value change *)
    fbIED.IEDLD1.XCBR1.Pos.t.tValue:=tT;
```

```
    IF fbIED.IEDLD1.XCBR1.Pos.stVal.eValue = E_AcsiDbpos.On THEN
        fbIED.IEDLD1.XCBR1.Pos.stVal.eValue:= E_AcsiDbpos.Off;
    ELSE
        fbIED.IEDLD1.XCBR1.Pos.stVal.eValue:= E_AcsiDbpos.On;
    END_IF
    fbIED.IEDLD1.XCBR1.Pos.q.OldData:= NOT fbIED.IEDLD1.XCBR1.Pos.q.OldData;
END_IF

(* Execute server control block implementations *)
fbIED.IEDLD1.LLN0.urcb101.Server.Execute();
fbIED.IEDLD1.LLN0.urcb201.Server.Execute();
fbIED.IEDLD1.LLN0.urcb301.Server.Execute();
fbIED.IEDLD1.LLN0.brcb101.Server.Execute();
fbIED.IEDLD1.LLN0.brcb201.Server.Execute();
fbIED.IEDLD1.LLN0.brcb301.Server.Execute();

CASE state OF
    0: (* Initial state *)
        IF _bAbort THEN (* Abort connection => execute AbortReq() command *)
            _bAbort:= FALSE;
            bSuccess:= fbConnection.AbortReq(ipReason:=fbAbortReason, ipSink:=0, ipResult=>ipResult)
;
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Established AND _bDisconnect THEN (* Close/
release connection => execute ReleaseReq() command *)
            _bDisconnect:= FALSE;
            bSuccess:= fbConnection.ReleaseReq(ipSink:=0, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Established THEN (* Connection established => exchang
e IED data *)
            state:= 10;
        END_IF
        _bDisconnect:= FALSE;

    1: (* Wait for AbortReq() or ReleaseReq() command completion *)
        IF ipResult <> 0 THEN
            ipResult.Execute();
            IF NOT (bBusy:=ipResult.IsBusy()) THEN
                state:= SEL(ipResult.IsCompleted(), 100(* failed or aborted *), 0(* succeeded *));
            END_IF
        END_IF

    10: (* connection established *)
        state:= 0;

    100: (* Error state *)
        state:= 0;
        IF ipResult <> 0 THEN
            sLastErrorResult:= ipResult.Dump();
        END_IF
END_CASE
```

7.25 Server - Data access events

This example shows possible applications for data access events.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/10698267531.zip

The example described here uses the state machine that is described in the "General Server project structure [▶ 700]" chapter. The States 0, 1, 10 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

When accessing a value in the data model via the property or communication services, user-specific behavior can be implemented by using the event methods. For this purpose, certain methods can be added to each data object function block, which are called by the Server application before and after write and read accesses. These are the following four methods:

Method name	Description
OnGetVPreEvent	Is called on read accesses before the data is read from the data model. In this method, the data to be read can be adapted or the read request can be rejected with an error code.
OnGetVEvent	Is called on read accesses after the data has been read from the data model. This method can be used to reset values set for a read request, for example.
OnSetVPreEvent	Is called on write accesses before the data is written to the data model. In this method, the data to be written can be checked and the write request rejected if necessary.
OnSetVEvent	Is called on in connection with write accesses after the data has been written to the data model. This method can be used to update other values according to the written value, for example.

The exemplary use cases for these methods as implemented in this example are explained in the following.

Substitution

The use of substitute values (substitution) can be implemented with the methods "OnGetVPreEvent" and "OnGetVEvent". In the example, this is implemented in the function blocks "FB_DO_IED_LD1_MMXU1_PhV_phsA" and "FB_DO_IED_LD1_MMXU1_PhV_phsB". In the first function block, if substitution is enabled, then the process value is overwritten with the substitute value in cases of read requests. In the second function block, the current process value is also cached in order to set it again after the read request.

Checking setting values

In the "FB_DO_IED_LD1_CCGR1_OilTmpSet" function block, the checking of write requests for setting values is implemented with the "OnSetVPreEvent" method. In the function block, the minimum and maximum valid value for the data attribute "setMag" can be set via the data attributes "minVal" and "maxVal". With each write request to "setMag", a check is now made to determine whether the value to be written is within the set range. If this is not the case, an error code is returned to the Server application. Then the value is not written and the error is sent to the Client.

Rounding of measured values

In the "OnSetVEvent" method, rounded measured values can be calculated and value ranges can be determined. This is implemented in the function block "FB_DO_IED_LD1_MMDC1_Vol". To do this, the value of the "IEDL1/MMDC1.Vol.instMag.f" data attribute is overwritten with a random value in the "FB_IEDServerSession" function block. With write access via the property, the "OnSetVEvent" method is called after the value has been transferred to the data model. In this, first the value of the data attribute "instMag.f" is rounded off and applied in the "mag.f" data attribute. Afterwards, the configured value ranges are used to determine in which value range the current value of "instMag.f" is located. In addition, a check is made to determine whether this is outside the defined range.

7.26 Server - Buffered Reporting

This sample shows the buffered reporting implementation in a TwinCAT IEC61850 server project. The BRCBs (buffered report control blocks) and datasets required for reporting can be configured in the TwinCAT Telecontrol Configurator or imported from an existing SCL file (e.g. ICD file). The instances of the BRCBs and the Datasets are created automatically during code generation (usually below LLN0). The initial values of the BRCB attributes can already be configured in the TwinCAT Telecontrol Configurator.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/11740489611.zip

The example described here uses the state machine that is described in the "General Server project structure [▶ 700]" chapter. The States 0, 1, 10 and 100 are identical to the state machine described there. Other states were modified for the example or new states were also added.

The implementation of the buffered reporting functionality in a TwinCAT IEC 61850 server project is practically a "black box". This means that the complete functionality is already encapsulated in a function block and the application only has to call this function block. Each BRCB has a function block subelement

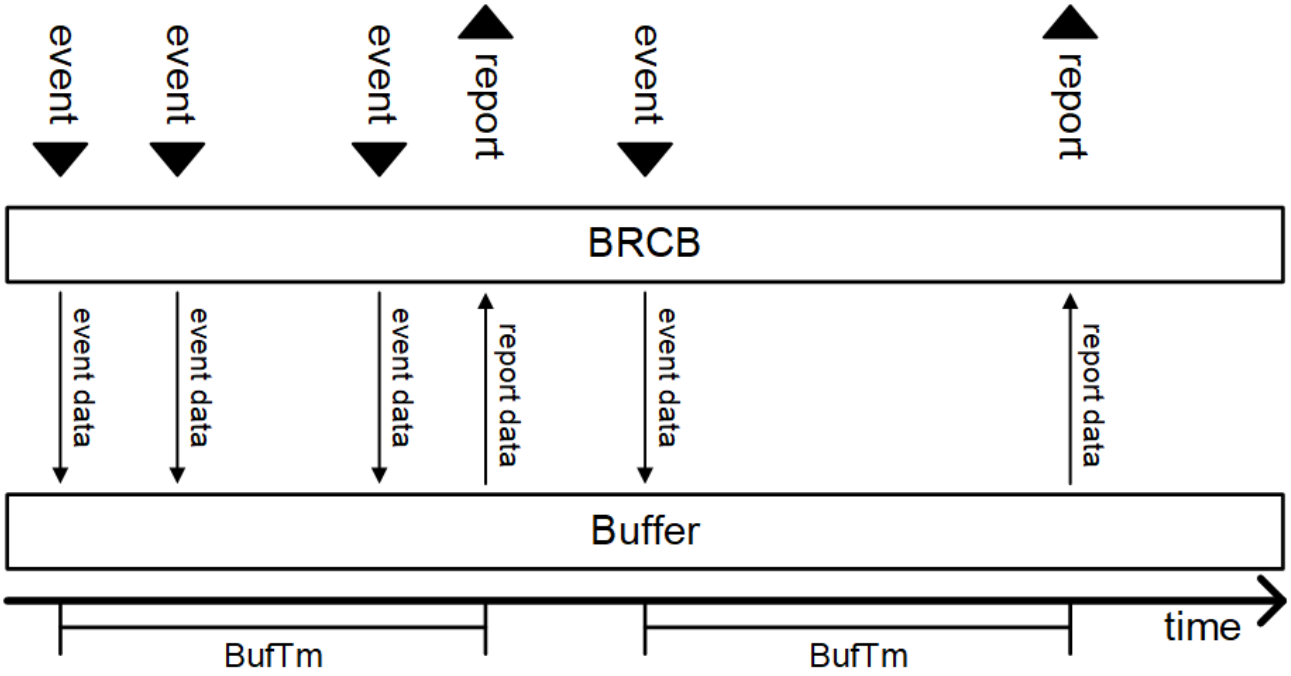
with the name: "Server". The PLC application only has to call the method: "Execute" at the "Server" block cyclically. This method is responsible for sending the reports and updating the status information in the associated BRCB (see code below).

The reports can be configured either on the server side directly on the BRCB or on the client side via client-server services (GetBRCBDataValues and SetBRCBDataValues). The IED data to be communicated via reporting are defined with the help of the datasets. Multiple data references (Dataset Members) can be configured in a dataset.

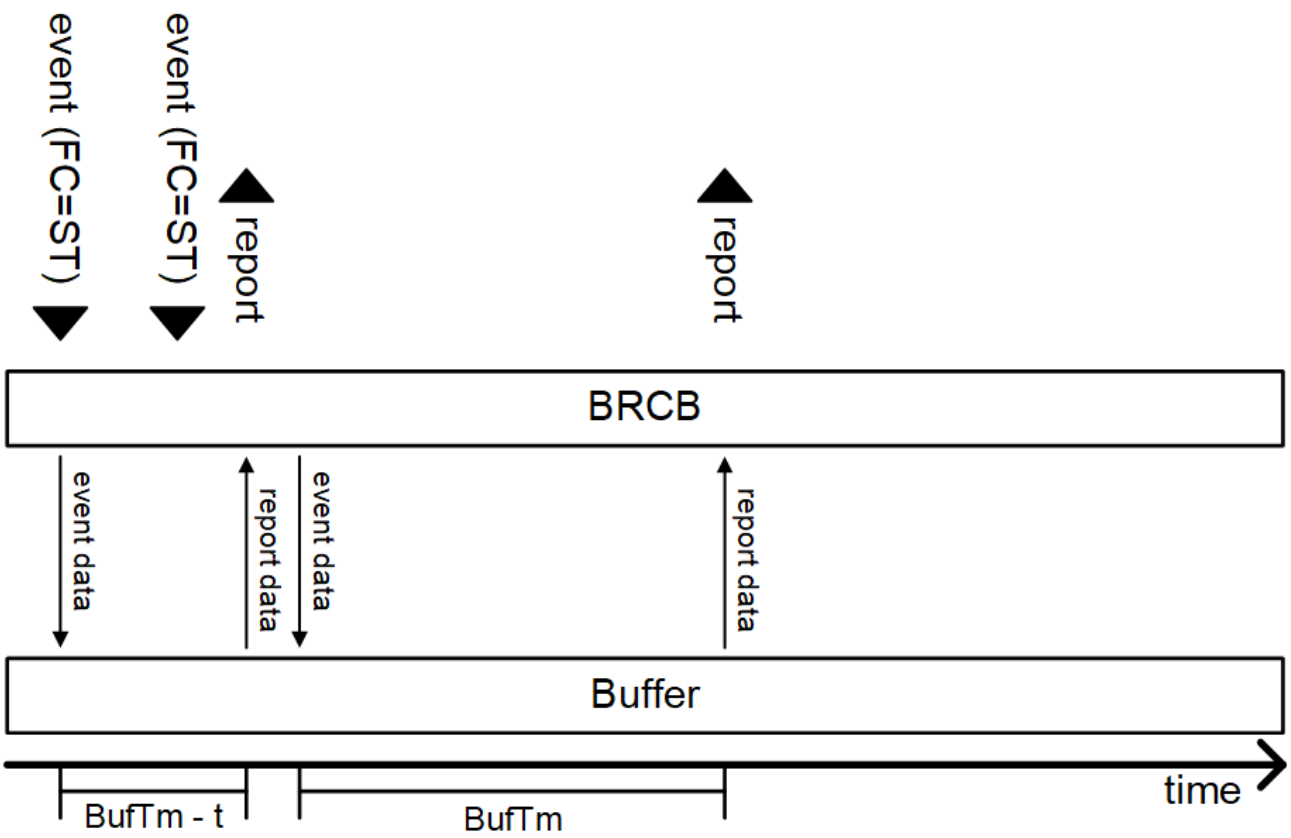
Reports that cannot be sent are stored in the buffer. When the maximum number of entries in the buffer is reached, the oldest report is deleted. When declaring a BRCB, the maximum number of entries can be set via the property "nMaxBufferEntries". In the sample this can be seen in the declaration of "brcb101":

```
brcb101: FB_ScsmBrCBImplClass := (Server:=(nMaxBufferEntries:=15), RptID:=(sValue:='IEDLD1/
LLN0.brcb101'), DatSet:=(sValue:='IEDLD1/
LLN0.DS1'), ConfRev:=(nValue:=1), OptFlds:=(SequenceNumber:=TRUE, ReportTimeStamp:=TRUE, ReasonForIn
clusion:=TRUE, DataSetName:=TRUE, DataReference:=TRUE, BufferOverflow:=TRUE, EntryID:=TRUE, ConfRevi
sion:=TRUE), TrgOps:=(DataChange:=TRUE, QualityChange:=TRUE, DataUpdate:=TRUE, Integrity:=TRUE, Gene
ralInterrogation:=TRUE), IntgPd:=(nValue:=5000), bLinkResult:=THIS^.AddBufferedReportControlBlockToC
ontainer(ipBufferedReportControlBlock:=brcb101));
```

The sample contains a simple simulation of some dataset member values. If the variable "bSimulation" is set to TRUE, then the values are modified every 2 seconds.
How does the BRCB Server implementation recognize that a value has been modified and a report is to be generated? This is done through write access to the attribute value (for example: bValue, tValue, iValue, eValue, nValue properties). However, a new report is not generated immediately for each individual write access to an attribute value. This would be undesirable. The "Server" implementation marks the modified dataset members internally as modified. A report is not generated and sent until the "Execute" method is called. The data sent in the report correspond to the data at the time of the dataset member's value change. The values are buffered when changed and coded into a report and sent after the time configured in the "BufTm" attribute has elapsed. If a new write access occurs within the buffer time, the buffered data are overwritten. This process is shown in the following figure:



A different behavior for buffering is defined in IEC 61850-7-2 for data of the functional group "ST". If the value is written again before "BufTm" expires, the buffered data are immediately coded to a report and sent. The new data are then written into the buffer and the monitoring of the buffer time is restarted. The following graphic shows the described behavior:

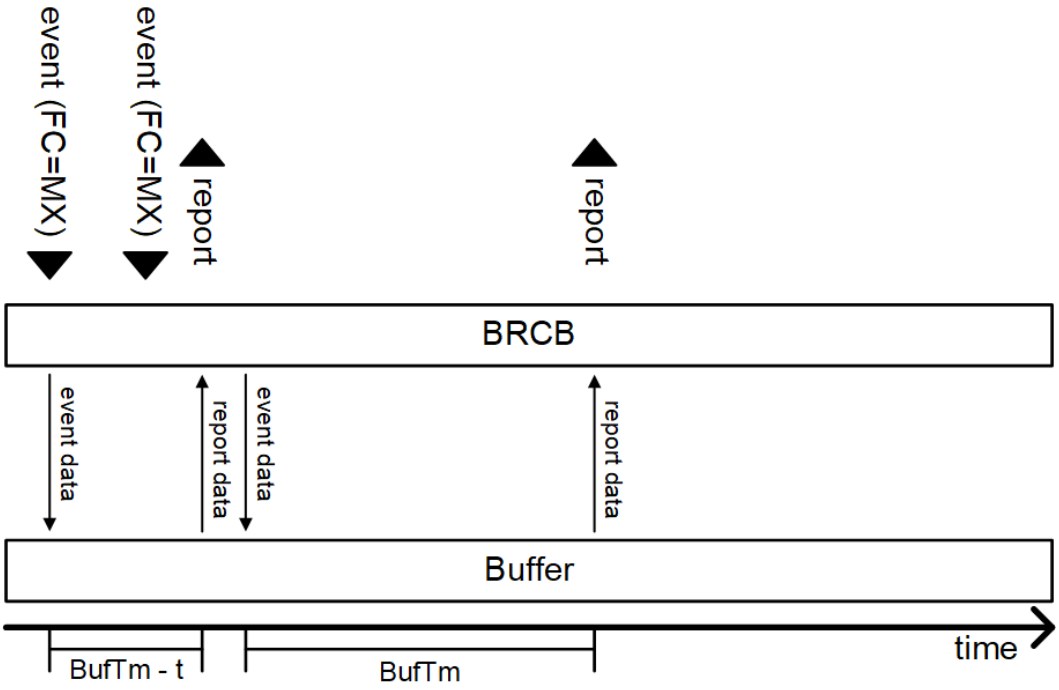


For data of the functional group "MX" it can be selected whether the BRCB should handle them separately, like data of the functional group "ST". This can be set via the property "bReplaceBufferedMX". In the sample, the special treatment is enabled at "brcb201":

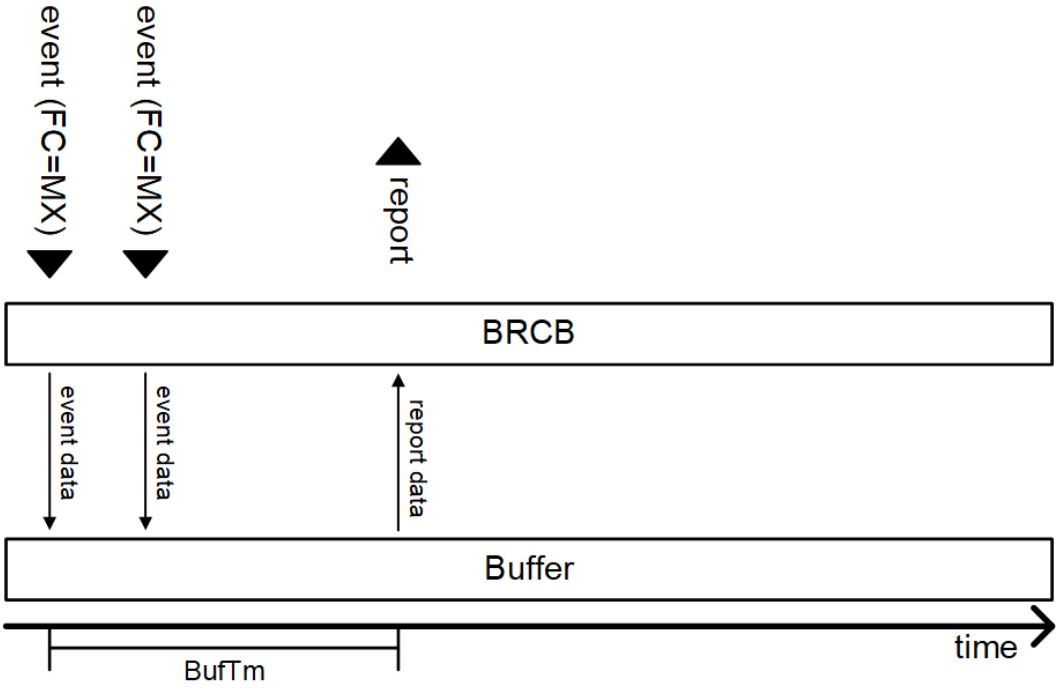
```
brcb201: FB_ScsmBrCBImplClass := (Server:=(bReplaceBufferedMX:=TRUE), RptID:=(sValue:='IEDLD1/
LLN0.brcb201'), DatSet:=(sValue:='IEDLD1/
LLN0.DS2'), ConfRev:=(nValue:=1), OptFlds:=(SequenceNumber:=TRUE, ReportTimeStamp:=TRUE, ReasonForIn
clusion:=TRUE, DataSetName:=TRUE, DataReference:=TRUE, BufferOverflow:=TRUE, EntryID:=TRUE, ConfRevi
sion:=TRUE), BufTm:=(nValue:=5000), TrgOps:=(DataChange:=TRUE, QualityChange:=TRUE, DataUpdate:=TRUE
, Integrity:=TRUE, GeneralInterrogation:=TRUE), IntgPd:=(nValue:=5000), bLinkResult:=THIS^.AddBuffer
edReportControlBlockToContainer(ipBufferedReportControlBlock:=brcb201));
```

How "MX" data are handled depending on the setting of "bReplaceBufferedMX" is shown in the following figure:

bReplaceBufferedMX = FALSE:



bReplaceBufferedMX = TRUE:



Sending reports immediately when a value of the assigned dataset is changed repeatedly can result in more reports being generated than desired. This is the case, for example, when a data object of the functional group "ST" is referenced in a dataset and then several attributes of the data object are written individually. Since each change in the value of a data attribute within a data object also represents a change in the value of the data object, multiple reports are generated. If the property "bOverwriteSameCycleChanges" is set to TRUE, multiple value changes within a PLC cycle are treated as a single value change. The recognition of which values were written in the current cycle is reset by calling the "Execute" method. A value change before and after the call are therefore interpreted as not being a change in the same cycle.

Please note that no reports are generated when writing or forcing values in TwinCAT online mode. In this way, the "Server" implementation cannot detect when a value has been modified.

Integrity reports and General Interrogation reports are automatically generated and sent in accordance with the configuration of the BRCB. The sending of these reports does not have to be triggered in a special way in the PLC application. It is also taken from the cyclic call to the server "Execute" method.

```
PROGRAM MAIN
VAR
    bSimulation                  : BOOL:=TRUE; (* Enables/disables simulation of data update/
change *)
    tSimulation                  : TIME:=T#2S; (* Cycle time of simulated data update/change *)
    fbTimer                      : TON;
    tT                           : T_UtcTime;

    bReplaceBufferedMX           : BOOL; (* Toggles bReplaceBufferedMX property of brcb201 *)
    bOverwriteSameCycleChanges   : BOOL; (* Toggles bOverwriteSameCycleChanges property of brcb301
*)
END_VAR
```

```
P_IEC61850MAIN();

fbIED.IEDLD1.LLN0.brcb201.Server.bReplaceBufferedMX:= bReplaceBufferedMX;
fbIED.IEDLD1.LLN0.brcb301.Server.bOverwriteSameCycleChanges:= bOverwriteSameCycleChanges;

fbTimer(IN:=bSimulation, PT:=tSimulation);
IF fbTimer.Q THEN(* Simulate server data update *)
    fbTimer(IN:=FALSE);
    fbTimer(IN:=bSimulation);

    fbIEDServer.GetSystemTime(ipAA:=0, tT=>tT);

    (* Simulate "IEDLD1/LLN0.DS1" member value change *)
    fbIED.IEDLD1.LEDGGIO1.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO1.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO1.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO2.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO2.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO2.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO3.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO3.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO3.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO4.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO4.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO4.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO5.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO5.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO5.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO6.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO6.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO6.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO7.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO7.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO7.SPCSO1.stVal.bValue;
    fbIED.IEDLD1.LEDGGIO8.SPCSO1.t.tValue:= tT;
    fbIED.IEDLD1.LEDGGIO8.SPCSO1.stVal.bValue:= NOT fbIED.IEDLD1.LEDGGIO8.SPCSO1.stVal.bValue;

    (* Simulate "IEDLD1/LLN0.DS2" member value change *)
    fbIED.IEDLD1.MMXU1.TotW.t.tValue:= tT;
    fbIED.IEDLD1.MMXU1.TotW.mag.f.fValue:= fbIED.IEDLD1.MMXU1.TotW.mag.f.fValue + 0.1;

    (* Simulate "IEDLD1/LLN0.DS3" member value change *)
    fbIED.IEDLD1.XCBR1.Pos.t.tValue:= tT;
    IF fbIED.IEDLD1.XCBR1.Pos.stVal.eValue = E_AcsiDbpos.On THEN
        fbIED.IEDLD1.XCBR1.Pos.stVal.eValue:= E_AcsiDbpos.Off;
    ELSE
        fbIED.IEDLD1.XCBR1.Pos.stVal.eValue:= E_AcsiDbpos.On;
    END_IF
    fbIED.IEDLD1.XCBR1.Pos.q.OldData:= NOT fbIED.IEDLD1.XCBR1.Pos.q.OldData;
    fbIED.IEDLD1.XCBR1.Pos.q.eValidity:= SEL(fbIED.IEDLD1.XCBR1.Pos.q.OldData, E_AcsiQualityValidity
.Good, E_AcsiQualityValidity.Questionable);
END_IF
```

```
FUNCTION_BLOCK FB_IEDServerSession IMPLEMENTS I_ScsmAbortIndEventSink, I_ScsmAssociateIndEventSink,
I_ScsmReleaseIndEventSink
VAR_INPUT
    fbConnection     : FB_iec61850ConnectionClass := (ipAbortInd:=THIS^, ipAssociateInd:=THIS^, ipRe
leaseInd:=THIS^);
END_VAR
VAR
    _bAbort          : BOOL; (* Rising edge activates AbortReq() command execution *)
    _bDisconnect     : BOOL; (* Rising edge activates ReleaseReq() command execution *)

    state            : BYTE;
    eState           : E_AsyncEnvironmentState; (* Environment state *)
```

```
    bBusy            : BOOL; (* If TRUE => command execution is busy *)
    bSuccess         : BOOL;
    ipResult         : I_AsyncServiceResultClass;
    sLastErrorResult : T_MaxString;
    fbAbortReason    : FB_ServiceErrorClass := (stError:=SUCCESS_EVENT);
    sLastAbortReason : T_MaxString;
END_VAR
```

```
fbConnection.Execute();
eState:= fbConnection.eState;

(* Execute server control block implementations *)
fbIED.IEDLD1.LLN0.brcb101.Server.Execute();
fbIED.IEDLD1.LLN0.brcb201.Server.Execute();
fbIED.IEDLD1.LLN0.brcb301.Server.Execute();

CASE state OF
     0: (* Initial state *)
          IF _bAbort THEN (* Abort connection => execute AbortReq() command *)
               _bAbort:= FALSE;
               bSuccess:= fbConnection.AbortReq(ipReason:=fbAbortReason, ipSink:=0, ipResult=>ipResu
lt);
               state:= SEL(bSuccess, 100, 1);
          ELSIF eState = E_AsyncEnvironmentState.Established AND _bDisconnect THEN (* Close/
release connection => execute ReleaseReq() command *)
               _bDisconnect:= FALSE;
               bSuccess:= fbConnection.ReleaseReq(ipSink:=0, ipResult=>ipResult);
               state:= SEL(bSuccess, 100, 1);
          ELSIF eState = E_AsyncEnvironmentState.Established THEN (* Connection established => excha
nge IED data *)
               state:= 10;
          END_IF
          _bDisconnect:= FALSE;

     1: (* Wait for AbortReq() or ReleaseReq() command completion *)
          IF ipResult <> 0 THEN
               ipResult.Execute();
               IF NOT (bBusy:=ipResult.IsBusy()) THEN
                    state:= SEL(ipResult.IsCompleted(), 100(* failed or aborted *), 0(* succeeded *)
);
               END_IF
          END_IF

     10: (* connection established *)
          state:= 0;

     100: (* Error state *)
          state:= 0;
          IF ipResult <> 0 THEN
               sLastErrorResult:= ipResult.Dump();
          END_IF
END_CASE
```

7.27 Multiple GOOSE publishers/subscribers on one network adapter (without client-server communication)

This sample shows the implementation of multiple GOOSE publishers and subscribers in a TwinCAT project. Several publishers and subscribers are combined into IED groups and configured in such a way that they only require one TwinCAT real-time Ethernet adapter for the exchange of GOOSE messages.

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/12863957387.zip

General information about this sample project

Sometimes there are several identical IED devices (Intelligent Electronic Devices) in a GOOSE network. These identical devices then logically have the same IEC 61850 data model (Logical Nodes, Data Objects, Data Attributes, etc.). If the TwinCAT application is to exchange GOOSE messages with several identical devices, then it is advantageous to create and use an array of IED instances for this purpose in the TwinCAT project. An IED array element then corresponds to a GOOSE publisher or subscriber device instance in the field. The array elements should preferably be either all of type: GOOSE publisher or all of type: GOOSE subscriber. This simplifies the implementation of the individual GOOSE functions in the PLC application. The PLC application can then, for example, enable or disable all publishers in a FOR loop. Other configurations

(some IED array elements are subscribers and other publishers) are also possible, but are not presented in this sample to keep it simple. The devices in the field logically have different IED and logical device names. However, all IED array elements initially have the same IED and logical device names. However, this is not a problem for the PLC application. The individual IED array elements can be reconfigured accordingly at PLC program start. Each IED array element is given a new, unique IED and LD name. Because of this renaming, some attribute values, such as “GoID” or “DatSet” must also be adjusted.

The IEC 61850 data model for these identical IED devices is then only present once in the TwinCAT project and does not have to be duplicated and renamed for each individual IED device instance. This saves resources and increases the performance of the application.

For projects with only one subscriber or publisher, the link from the “TwinCAT real-time Ethernet adapter” (RT Ethernet adapter) to the PLC is established via the instances of the “FB_[IEDName]Gse” function block. The IED data model of a subscriber or publisher to be communicated is passed to the function block via the “ipIED” property. For more information see: RT Ethernet adapter configuration [▶ 35] and in the samples: Client - GOOSE subscriber [▶ 693] and Server - GOOSE publisher [▶ 704].
It is theoretically possible to link up to 20 instances of the “FB_[IEDName]Gse” function block with a “TwinCAT RT Ethernet adapter” in this way. This allows up to 20 publishers or subscribers to be implemented in one project. However, it is more resource-efficient to use, for example, only one instance of the adapter function block for a group of publishers and another instance for the group of subscribers.

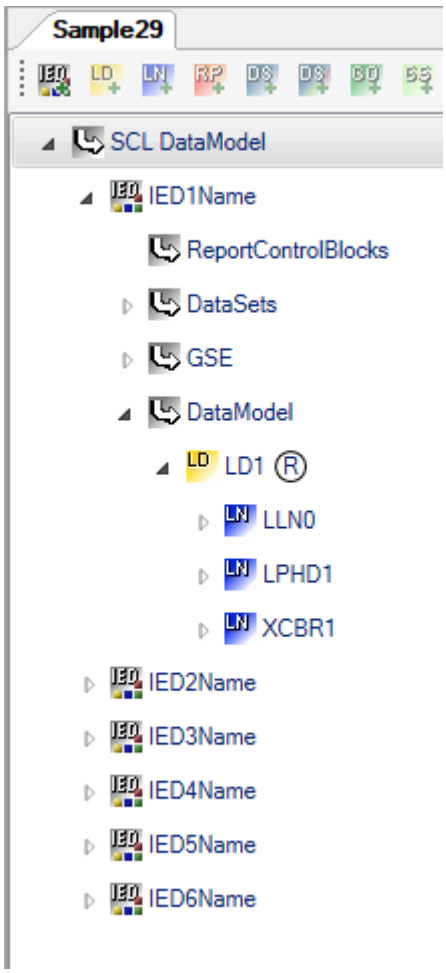
If all devices are in a common GOOSE network, then all publishers/subscribers in the TwinCAT application can use the same “Real-Time Ethernet adapter (Multiple Protocol Handler)”. How has it been realized in the sample? First, several publishers were assigned to an IED group. All subscribers were assigned to a second IED group. Each IED group was given a unique group name. The IED groups are instances of the function block: “FB_GROUP_[GroupName]”. The IED group to be communicated is communicated to the “FB_[GroupName]Gse” adapter function block via the “ipGroup” property. The two instances of the adapter function block “FB_[GroupName]Gse” (one belongs to the publisher group and the other to the subscriber group) are then linked to the “TwinCAT real-time Ethernet adapter” as described in the RT Ethernet adapter configuration [▶ 35].

The sample consists of two separate TwinCAT IEC 61850 projects. After unpacking the zip archive, you will find two TwinCAT projects in the subfolders: SystemA and SystemB: **|SystemA|Sample29|Sample29.sln** and **|SystemB|Sample29|Sample29.sln.** For the sake of simplicity, only the SystemA project is described here: |**SystemA|Sample29|Sample29.sln**.

The second project: |SystemB|Sample29|Sample29.sln can be used as a counterpart for a communication test if you download, activate, and start the project on a second TwinCAT runtime system. Both projects are coordinated with each other. The publishers from the SystemA project are implemented as subscribers in the SystemB project and the subscribers from the SystemA project are implemented as publishers in the SystemB project.

Generation of the IED data model

The IEC 61850 data model of the IEDs in this sample was first generated with the TwinCAT Telecontrol Configurator from the SCL file: *Sample29.icd*. The file is also located in the TwinCAT project zip archive. This ICD file describes the data model of 6 different IED devices. The image below shows the IEDs in the TwinCAT Telecontrol Configurator:



Each IED has a DataSet and a GOOSE control block. After automatic PLC code generation (e.g., as a client project), the PLC project was modified manually and the unneeded function blocks for client/server communication were removed.

Definition of the IED groups

From the 6 automatically generated IED data models, a total of 3 publishers and 11 subscribers are to be implemented in this sample project. All publishers were assigned to an IED group named “PubGroupName” and subscribers were assigned to an IED group named “SubGroupName”.

Global variable list TcTelecontrol

In this GVL, a publisher group “fbPubGroupName” and a subscriber group “fbSubGroupName” are instantiated respectively. There is also an instance of the publisher adapter function block “fbPubGroupNameGse” and an instance of the subscriber adapter function block “fbSubGroupNameGse”.

The “ipGroup” property is used to assign the publisher IED group to the “fbPubGroupNameGse” and the subscriber IED group to the “fbSubGroupNameGse”.

```
VAR_GLOBAL
    ipCreator: I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=3).SetGuiVer(major
:=3, minor:=1, build:=96, revision:=3);

    fbSubGroupName: FB_GROUP_SubGroupName:=(bLinkResult:=fbSubGroupName.Init());
    fbPubGroupName: FB_GROUP_PubGroupName:=(bLinkResult:=fbPubGroupName.Init());

    fbSubGroupNameGse: FB_SubGroupNameGse := (fbAdapter:=(ipGroup:=fbSubGroupName, settings:=(sMulti
castAddr:='00-00-00-00-00-00', eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
    fbPubGroupNameGse: FB_PubGroupNameGse := (fbAdapter:=(ipGroup:=fbPubGroupName, settings:=(sMulti
castAddr:='00-00-00-00-00-00', eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
END_VAR
```

Subscriber group

The subscriber group “FB_GROUP_SubGroupName” configures 11 subscribers. “IED5Name” is instantiated as an array[1..8] in the subscriber group. Each array element shall be a subscriber that receives GOOSE messages from another identical device in the field. The table below lists the main parameters of the subscribers in this IED group.

IED	IED name/LD name	SrcAddress	APPID	GoID	DatSet
IED1Name	IED1Name/IED1NameLD1	01-0C-CD-01-00-01	16#0010	'IED1NameLD1/LLN0.gocb01'	'IED1NameLD1/LLN0.ds01'
IED2Name	IED2Name/IED2NameLD1	01-0C-CD-01-00-01	16#0020	'IED2NameLD1/LLN0.gocb01'	'IED2NameLD1/LLN0.ds01'
IED3Name	IED3Name/IED3NameLD1	01-0C-CD-01-00-01	16#0030	'IED3NameLD1/LLN0.gocb01'	'IED3NameLD1/LLN0.ds01'
IED5Name[1]	CH1/CH1LD1	01-0C-CD-01-00-01	16#0051	'CH1LD1/LLN0.GoID.gocb01'	'CH1LD1/LLN0.ds01'
IED5Name[2]	CH2/CH2LD1	01-0C-CD-01-00-02	16#0052	'CH2LD1/LLN0.GoID.gocb01'	'CH2LD1/LLN0.ds01'
IED5Name[3]	CH3/CH3LD1	01-0C-CD-01-00-03	16#0053	'CH3LD1/LLN0.GoID.gocb01'	'CH3LD1/LLN0.ds01'
IED5Name[4]	CH4/CH4LD1	01-0C-CD-01-00-04	16#0054	'CH4LD1/LLN0.GoID.gocb01'	'CH4LD1/LLN0.ds01'
IED5Name[5]	CH5/CH5LD1	01-0C-CD-01-00-05	16#0055	'CH5LD1/LLN0.GoID.gocb01'	'CH5LD1/LLN0.ds01'
IED5Name[6]	CH6/CH6LD1	01-0C-CD-01-00-06	16#0056	'CH6LD1/LLN0.GoID.gocb01'	'CH6LD1/LLN0.ds01'
IED5Name[7]	CH7/CH7LD1	01-0C-CD-01-00-07	16#0057	'CH7LD1/LLN0.GoID.gocb01'	'CH7LD1/LLN0.ds01'
IED5Name[8]	CH8/CH8LD1	01-0C-CD-01-00-08	16#0058	'CH8LD1/LLN0.GoID.gocb01'	'CH8LD1/LLN0.ds01'

The “FB_GROUP_SubGroupName” group function block is derived from the “FB_AcsiCommonIEDGroupClass” function block and serves as a container for several IEDs. The “AddIEDToContainer” method adds the individual IEDs to the subscriber group. For IED array elements, each individual array element must be added to the subscriber group in this way.

The “Tag” property can be used to provide the individual array elements of “IED5Name” with additional information. In the sample, the array element numbers are set there. This number can be queried in the “Init” method or later at runtime in the methods of the IED. Each array element can also be configured a little differently depending on the number.

```
FUNCTION_BLOCK FB_GROUP_SubGroupName EXTENDS FB_AcsiCommonIEDGroupClass
VAR CONSTANT
    Max_GROUP_IED5Name: DINT:=8; (* Max. number of IED5Name instances *)
END_VAR
VAR_INPUT
    IED1Name: FB_IED_IED1Name:=(bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED1Name));
    IED2Name: FB_IED_IED2Name:=(bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED2Name));
    IED3Name: FB_IED_IED3Name:=(bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED3Name));
    IED5Name: ARRAY[1..Max_GROUP_IED5Name] OF FB_IED_IED5Name:=
    [
        (Tag:=1, sObjectName:='CH1', bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED5Name[1])),
```

```
        (Tag:=2, sObjectName:='CH2', bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED5Name[2])),
        (Tag:=3, sObjectName:='CH3', bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED5Name[3])),
        (Tag:=4, sObjectName:='CH4', bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED5Name[4])),
        (Tag:=5, sObjectName:='CH5', bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED5Name[5])),
        (Tag:=6, sObjectName:='CH6', bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED5Name[6])),
        (Tag:=7, sObjectName:='CH7', bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED5Name[7])),
        (Tag:=8, sObjectName:='CH8', bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED5Name[8]))
    ];
END_VAR
VAR
    fbError     : FB_ServiceErrorClass;
    ipLastError : I_ServiceErrorClass;
    i           : DINT; (* Temp array index variable *)
END_VAR
```

Subscriber group methods

The function block “FB_GROUP_SubGroupName” implements several methods. “Enable” enables all subscribers. “Disable” disables all subscribers. “Execute” must be called cyclically. “Init” initializes all IEDs, LDs, LNs, etc. in the group. The method “Init” is executed once by the initialization code (already before the first PLC cycle).

Adjustment of IED name and LD name for IED5Name array

The setter of the “sObjectName” property of the “FB_IED_IED5Name” function block was overwritten. When setting a new IED name, the new setter implementation is called and the new IED name is applied. The logical device name of the function block “FB_LD_IED5NameLD1” is also corrected because it also contains the IED name.

```
PROPERTY FINAL sObjectName : T_AcsiObjectName
VAR
END_VAR
```

```
(* Adopt the modified IED name *)
SUPER^.sObjectName:= sObjectName;

(* Adjust the logical-device names too *)
IED6NameLD1.sObjectName:= CONCAT(sObjectName, 'LD1');
```

Adjustment of APPID, GoID and DatSet for IED5Name array

The adjustment is made in the “Init” method of the logical node “LLN0”.

```
METHOD FINAL Init : BOOL
VAR_INPUT
    IED: REFERENCE TO FB_IED_IED5Name;
    LogicalDevice: REFERENCE TO FB_LD_IED5NameLD1;
END_VAR
VAR
    mac: ETHERNET_ADDRESS;
END_VAR
```

```
IF NOT __ISVALIDREF(IED) THEN RETURN; END_IF
IF NOT __ISVALIDREF(LogicalDevice) THEN RETURN; END_IF

(* Example: Adjust subscribers GoID string *)
gocb01.GoID.sValue:= CONCAT(CONCAT(LogicalDevice.sObjectName, '/'), CONCAT(CONCAT(THIS^.sObjectName,
 '.'), gocb01.sObjectName));

(* Example: Adjust subscribers data-set reference string *)
gocb01.DatSet.sValue:= CONCAT(CONCAT(LogicalDevice.sObjectName, '/'), CONCAT(CONCAT(THIS^.sObjectNam
e, '.'), ds01.sObjectName));

(* Example: Adjust publishers multicast address by use of IED's Tag property value *)
mac:= gocb01.DstAddress.macAddr;// Get current address
mac.b[5]:= TO_BYTE(IED.Tag);// Modifiy last digit
gocb01.DstAddress.macAddr:= mac;// Set new address

(* Example: Adjust subscribers APPID by use of IED's Tag property value *)
gocb01.DstAddress.APPID.nValue:= WORD#16#0050 + TO_WORD(IED.Tag);

(* Forward IED reference to all data-sets *)
Init:= ds01.Init(IED:=IED);
```

Publisher group

The publisher group “FB_GROUP_PubGroupName” configures 3 publishers. “IED6Name” is instantiated as array[1..2] in the publisher group. Each array element shall be a publisher that sends the GOOSE messages to a different target address and with a special APPID. The table below lists the main parameters of the publishers in this group.

IED	IED name/LD name	DstAddress	APPID	GoID	DatSet
IED4Name	IED4Name/ IED4NameLD1	01-0C-CD-01-00-00	16#0040	'IED4NameLD1/ LLN0.gocb01'	'IED4NameLD1/ LLN0.ds01'
IED6Name[1]	CNT1/CNT1LD1	01-0C-CD-01-00-01	16#0061	'CNT1LD1/ LLN0.gocb01'	'CNT1LD1/LLN0.ds01'
IED6Name[2]	CNT2/CNT2LD1	01-0C-CD-01-00-02	16#0062	'CNT2LD1/ LLN0.gocb01'	'CNT2LD1/LLN0.ds01'

The “FB_GROUP_PubGroupName” group function block is derived from the “FB_AcsiCommonIEDGroupClass” function block and serves as a container for several IEDs. The “AddIEDToContainer” method adds the individual IEDs to the publisher group. For IED array elements, each individual array element must be added to the publisher group in this way.

The “Tag” property can be used to provide the individual array elements of “IED6Name” with additional information. In the sample, the array element numbers are set there. This number can be queried in the “Init” method or later at runtime in the methods of the IED. Each array element can also be configured a little differently, depending on the number.

```
FUNCTION_BLOCK FB_GROUP_PubGroupName EXTENDS FB_AcsiCommonIEDGroupClass
VAR CONSTANT
    Max_GROUP_IED6Name: DINT:=2; (* Max. number of IED6Name instances *)
END_VAR
VAR_INPUT
    IED4Name: FB_IED_IED4Name:=(bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED4Name));
    IED6Name: ARRAY[1..Max_GROUP_IED6Name] OF FB_IED_IED6Name:=
    [
        (Tag:=1, sObjectName:='CNT1', bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED6Name[1])),
        (Tag:=2, sObjectName:='CNT2', bLinkResult:=THIS^.AddIEDToContainer(ipIED:=IED6Name[2]))
    ];
END_VAR
VAR
    bUpdate_IED4NameLD1_LLN0_gocb01: BOOL; (* If TRUE => Update 'IED4Name.IED4NameLD1.LLN0.gocb01' p
ublisher *)
    aUpdate_IED6NameLD1_LLN0_gocb01: ARRAY[1..Max_GROUP_IED6Name] OF BOOL; (* If array element n is
TRUE => Update 'IED6Name[n].IED6NameLD1.LLN0.gocb01' publisher *)

    fbError          : FB_ServiceErrorClass;
    ipLastError      : I_ServiceErrorClass;
    i                : DINT; (* Temp array index variable *)
    iUpdateElement   : DINT; (* Array index of first array element that needs to be updated *)

    bSimulation      : BOOL := TRUE; (* Enables/disables simulation of cyclic GOOSE data update *)
    tSimulation      : TIME := T#5S; (* Cycle time of simulated GOOSE data update *)
    fbUpdateTimer    : TON;
    iLiveCounter     : LINT; (*Live counter (example) *)
    tT               : T_UtcTime; (* Temp time-stamp value *)
END_VAR
```

Publisher group methods

The function block “FB_GROUP_PubGroupName” implements several methods. “Start” starts all publishers. “Stop” stops all publishers. “Execute” must be called cyclically. “Update” performs an update of the publisher data. “Init” initializes all IEDs, LDs, LNs, etc. in the group. The method “Init” is executed once by the initialization code (already before the first PLC cycle). In the method “MapSimuPubData” publisher data are simulated/modified.

Adjustment of IED name and LD name for IED6Name array

The setter of the “sObjectName” property of the “FB_IED_IED6Name” function block was overwritten. When setting a new IED name, the new setter implementation is called and the new IED name is applied. The logical device name of the function block “FB_LD_IED6NameLD1” is also corrected because it also contains the IED name.

```
PROPERTY FINAL sObjectName : T_AcsiObjectName
VAR
END_VAR
```

```
(* Adopt the modified IED name *)
SUPER^.sObjectName:= sObjectName;

(* Adjust the logical-device names too *)
IED6NameLD1.sObjectName:= CONCAT(sObjectName, 'LD1');
```

Adjustment of APPID, GoID and DatSet for IED6Name array

The adjustment takes place in the “Init” method of the logical node “LLN0”.

```
METHOD FINAL Init : BOOL
VAR_INPUT
    IED: REFERENCE TO FB_IED_IED6Name;
    LogicalDevice: REFERENCE TO FB_LD_IED6NameLD1;
END_VAR
VAR
    mac: ETHERNET_ADDRESS;
END_VAR
```

```
IF NOT __ISVALIDREF(IED) THEN RETURN; END_IF
IF NOT __ISVALIDREF(LogicalDevice) THEN RETURN; END_IF

(* Example: Adjust publishers GoID string *)
gocb01.GoID.sValue:= CONCAT(CONCAT(LogicalDevice.sObjectName, '/'), CONCAT(CONCAT(THIS^.sObjectName,
 '.'), gocb01.sObjectName));

(* Example: Adjust publishers data-set reference string *)
gocb01.DatSet.sValue:= CONCAT(CONCAT(LogicalDevice.sObjectName, '/'), CONCAT(CONCAT(THIS^.sObjectNam
e, '.'), ds01.sObjectName));

(* Example: Adjust publishers multicast address by use of IED's Tag property value *)
mac:= gocb01.DstAddress.macAddr;// Get current address
mac.b[5]:= TO_BYTE(IED.Tag);// Modifiy last digit
gocb01.DstAddress.macAddr:= mac;// Set new address

(* Example: Adjust publishers APPID by use of IED's Tag property value *)
gocb01.DstAddress.APPID.nValue:= WORD#16#0060 + TO_WORD(IED.Tag);

(* Forward IED reference to all data-sets *)
Init:= ds01.Init(IED:=IED);
```

In the sample code above, the last digit of the MAC address of “DstAddress” is set equal to the value of the “Tag” property. If a completely different MAC address is to be configured for the two publishers then the “Tag” property can also be used in the following way:

```
IF IED.Tag = 1 THEN
    THIS^.gocb01.DstAddress.sAddr:='01-0C-CD-01-02-56';
ELSIF IED.Tag = 2 THEN
    THIS^.gocb01.DstAddress.sAddr:='01-0C-CD-01-01-78';
END_IF
```

Network adapter function block FB_SubGroupNameGse

A rising edge at the Boolean variable “bSubscribe” (initial value is already “TRUE”) enables all subscribers in the subscriber group when the PLC is started. A rising edge at the Boolean variable “bUnsubscribe” disables all subscribers in the group. The rest of the time, the “Execute” method of the subscriber group and the network adapter must be called cyclically. This is necessary so that the GOOSE messages are received and passed on to the subscriber instances.

```
FUNCTION_BLOCK FB_SubGroupNameGse IMPLEMENTS I_GseLinkStatusEventSink
VAR_INPUT
    fbAdapter   : FB_GseAdapterClass := (ipLinkStatus:=THIS^);
END_VAR
VAR
    eLinkStatus : E_GseLinkStatus;
    bSuccess    : BOOL;
    ipError     : I_ServiceErrorClass;
    bSubscribe  : BOOL := TRUE; (* Rising edge enables subscriber GoCB's *)
    bUnsubscribe: BOOL; (* Rising edge disables subscriber GoCB's *)
END_VAR
```

```
bSuccess:= fbAdapter.Execute(ipError=>ipError); (* Execute network adapter *)

IF bSubscribe THEN (* Enable subscriber GoCB's *)
    bSubscribe:= FALSE;
```

```
    bSuccess:= fbSubGroupName.Enable(ipAdapter:=fbAdapter, ipError=>ipError);
ELSIF bUnsubscribe THEN (* Disable subscriber GoCB's *)
    bUnsubscribe:= FALSE;
    bSuccess:= fbSubGroupName.Disable(ipError=>ipError);
ELSE (* Execute subscriber GoCB's *)
    bSuccess:= fbSubGroupName.Execute(ipError=>ipError);
END_IF
```

Network adapter function block FB_PubGroupNameGse

A rising edge at the Boolean variable “bStart” (initial value is already “TRUE”) enables all publishers in the publisher group when the PLC is started. A rising edge at the Boolean variable “bStop” disables all publishers in the group. On a rising edge at the Boolean variable “bUpdate”, the “Update” method of the publisher group is called. But the “bUpdate” variable becomes “TRUE” only if the “MapSimuPubData” method also returns “TRUE” because the data was changed or updated by at least one publisher. The rest of the time, the “Execute” method of the publisher group and the network adapter must be called cyclically. This is necessary so that the publisher instances can send their GOOSE messages.

```
FUNCTION_BLOCK FB_PubGroupNameGse IMPLEMENTS I_GseSystemClockEventSink, I_GseLinkStatusEventSink
VAR_INPUT
    fbAdapter   : FB_GseAdapterClass := (ipSystemClock:=THIS^, ipLinkStatus:=THIS^);
END_VAR
VAR
    eLinkStatus : E_GseLinkStatus;
    bSuccess    : BOOL;
    ipError     : I_ServiceErrorClass;
    bStart      : BOOL := TRUE; (* Rising edge starts publishing of GOOSE data *)
    bStop       : BOOL; (* Rising edge stops publishing of GOOSE data *)
    bUpdate     : BOOL; (* Rising edge updates published GOOSE data *)
    bSync       : BOOL := TRUE; (* Rising edge synchronizes the GSE system clock to external time *)
    tSync       : T_UtcTime := String_TO_UtcTime(in:='UT#2019-07-12-12:00:00.000000000|000|
3'); (* External (test) time *)
    fbClock     : FB_GseSystemClock;
END_VAR
```

```
IF bSync THEN (* Synchronize clock time to external time source *)
    bSync:= FALSE;
    bSuccess:= fbClock.SetToUtcTime(in:=tSync);
ELSE (* Execute clock instance *)
    fbClock.Execute();
END_IF

(* Map/simulate GOCB's data-set member values: 'IED4Name.IED4NameLD1.LLN0.ds01' and/
or 'ARRAY OF IED6Name[1..Max_GROUP_IED6Name].IED6NameLD1.LLN0.ds01' *)
bUpdate:= fbPubGroupName.MapSimuPubData(ipClock:=fbClock);

bSuccess:= fbAdapter.Execute(ipError=>ipError); (* Execute network adapter *)

IF bStart THEN (* Start publishing of GOOSE data *)
    bStart:= FALSE;
    bSuccess:= fbPubGroupName.Start(ipAdapter:=fbAdapter, ipError=>ipError);
ELSIF bStop THEN (* Stop publishing of GOOSE data *)
    bStop:= FALSE;
    bSuccess:= fbPubGroupName.Stop(ipError=>ipError);
ELSIF bUpdate THEN (* Update published GOOSE data *)
    bUpdate:= FALSE;
    bSuccess:= fbPubGroupName.Update(ipError=>ipError);
ELSE (* Execute publisher GoCB's *)
    bSuccess:= fbPubGroupName.Execute(ipError=>ipError);
END_IF
```

Simulation of publisher data in the sample project

If the Boolean variable “bSimulation” has the value “TRUE”, then some changing publisher data will be simulated in the method “MapSimuPubData”. E.g. a counter value “iLiveCounter” or some quality flags.

In addition, a very simple data concentrator was implemented there for testing purposes. Some subscriber data of the subscriber group IEDs are copied over to the publisher data when the value changes. In case of a value change, a Boolean variable such as „bUpdate_GROUP_IED4NameLD1_LLN0_gocb01“ is also set to “TRUE”. The “Update” method can detect via this variable that an update of the publisher data of the control block “gocb01” is to be performed.

Test of the subscriber group

There are other ICD files in the zip archive that can be used for testing or simulation purposes of the subscriber group:

IED1Name.icd
IED2Name.icd
IED3Name.icd
CH1_To_CH8.icd

You can use these files in a third party tool to simulate the “IED1Name”, “IED2Name”, “IED3Name”, and “CH1”…“CH8” publishers. Furthermore, you can also create a TwinCAT publisher project from these files. The subscribers in this sample project can receive the data from these publishers.

Publisher group test

There are two more ICD files in the zip archive that can be used when testing the publisher group in this sample project:

CNT1_To_CNT2.icd
IED4Name.icd

You can use these files in a third party tool to simulate the “CNT1”.. “CNT8” and “IED4Name” subscribers. The sent data of the publisher group in this sample project can be received by these subscribers.

7.28 TLS - Secure client-server communication using certificates or PSK (pre-shared key)

This example shows how secure data communication can be implemented using TLS (Transport Layer Security) in a TwinCAT IEC 61850 client and server application. The TLS protocol only allows encryption of client-server communication. The GOOSE publisher/subscriber data cannot be encrypted in this way. If you are using an older version of the TwinCAT IEC 61850 Telecontrol Configurator, the TLS functionality may not be supported there yet. To compile and test this sample project without errors you need a newer version of the TF6510 IEC 61850 Telecontrol.

System requirement

TF6510 IEC 61850 Telecontrol v3.1.96.7 or newer

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/13651402763.zip

General information about this sample project

The TLS sample consists of two separate TwinCAT IEC 61850 projects. After unpacking the zip archive, you will find a TwinCAT IEC 61850 client project in the subfolder *\ClientWithTLS\Sample30* and a TwinCAT IEC 61850 server project in the subfolder *\ServerWithTLS\Sample30*.

The here described: Client - Basic sample project [▶ 669] and here described: Server - Basic sample project [▶ 704] served as basis for the TLS sample project.

This sample mainly shows two ways of TLS data encryption:

1. TLS with certificates (e.g.: own, self-signed certificates).
2. TLS with PSK (Pre-Shared Key).

Every TwinCAT IEC 61850 client/server project can be extended with TLS functionality in a relatively simple way. In order for the encryption/decryption of the data to work on both sides, this must also logically be done on both sides, the client and the server. This extension consists mainly of the configuration of private keys, passwords, certificates or PSK keys. When configuring the private keys and certificates, for example, the server or client function block is informed of the file paths to the folders where the private key files and certificates have been saved. The required TLS configuration parameters are saved in advance in a function block instance of the type: FB_SocketTlsSettingsClass [▶ 244] provided for this purpose. This information is

transferred to the IEC 61850 client/server function blocks via a defined interface of the type I_SocketTlsSettingsClass [▶ 394], which is implemented by the FB_SocketTlsSettingsClass [▶ 244] function block. The IEC 61850 server and client function blocks access the saved TLS configuration and thus also the private key, certificate files via this interface during connection establishment. Upon success, a secure communication connection is established.

Format of private keys and certificates

The certificate files must be in PEM (Privacy-Enhanced Mail) format. To ensure that the client or server can easily access the key and certificate files, the recommended location for the files is, for example, the following folder on the Windows target system: *\TwinCAT\3.1\Target\Certificates\IEC61850*.

The private key files and certificate files are text files. A private key file starts with the line:
“-----BEGIN PRIVATE KEY-----”.
This is followed by the key and ends with the line:
“-----END PRIVATE KEY-----”.
Sample:

```
-----BEGIN PRIVATE KEY-----
MIIEzAIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDIx4bVTxEHDhuKWl0XJAkLZfqj
NvMkD26sv/VViprpeMCbU/fSun+2oJO1iczd7Ut66SQBBePxsFEQhfAhO0TaGDDrfF9WP7WbkGb7
...
V0um2x+kKJ8hsD9vfORBf7KnwJi0QitwG41PoGDUi6RD8IUybKbLc5a/hx1C5hR0TBnXuxpguzed
pq0NPKFQk7d0ArahQPrYmUySyfagDTALBgNVHQ8xBAMCAJA=
-----END PRIVATE KEY-----
```

The certificate files start with the line:
“-----BEGIN CERTIFICATE-----”.
This is followed by the certificate data and ends with the line:
“-----END CERTIFICATE-----”.
Sample:

```
-----BEGIN CERTIFICATE-----
MIIDyDCCArCgAwIBAgIQRGxL9tvn64FPEmnMEI8FLjANBgkqhkiG9w0BAQsFADBl
MWMwYQYDVQQDDFpSb290IENBLE9VPVR3aW5DQVQsTz1CZWNraG9mZiBBdXRvbWF0
...
6D2L4WEjrzIMR07EWJC4JKvDqxiQMAHsUkpy4vS817ZBsuL0/M8EbG9sTdEGxZp
9+GR0Np4ku7TkdJ5
-----END CERTIFICATE-----
```

TLS sample configuration with the FB_IEDTLS function block

In order to further simplify the TLS configuration with the help of the FB_SocketTlsSettingsClass [▶ 244] function block, an additional auxiliary “FB_IEDTLS” function block was implemented in the “ClientWithTLS\Sample30” and also in the “ServerWithTLS\Sample30” project around the FB_SocketTlsSettingsClass [▶ 244] function block.

The “FB_IEDTLS” function block now has only one method: “Enable”. Using the “Enable” method parameter, you can enable or disable the TLS configuration and specify the type of TLS encryption in the example.

```
METHOD FINAL Enable : I_SocketTlsSettingsClass
VAR_INPUT
    bEnable : BOOL;
    bPSK    : BOOL;
END_VAR
```

bEnable: Enables/disables the TLS configuration. If “TRUE”, all necessary TLS configuration settings are saved in the local instance “fbTls” of the FB_SocketTlsSettingsClass [▶ 244] function block. The TLS configuration should be enabled in this case. The value “FALSE” disables the TLS configuration.

bPSK: Configures the type of encryption. If “TRUE”, a secure TLS connection is established by using a PSK (Pre-Shared Key). If “FALSE”, a secure TLS connection is established by exchanging the certificates.

Return parameter: If a valid interface pointer to the TLS configuration settings is returned (return value <> 0), then a secure TLS client-server connection shall be established. If the return value is null, a conventional connection (without TLS) should be established.

Server TLS sample configuration

The implementation of the “FB_IEDTLS.Enable” method differs in the client and server projects. In the server project, the paths of the certificates and private keys of the server are configured, and in the client project, those of the client. The root CA certificate is the same certificate on both sides. The root CA certificate must be present on both systems.

```
VAR
    key       : ARRAY[0..14] OF BYTE:=[16#1B,16#D0,16#6F,16#D2,16#56,16#16,16#7D,16#C1,16#E8,16#C7,1
6#48,16#2A,16#8E,16#F5,16#FF];
    sIdentity : STRING(TCPADS_TLS_PSK_IDENTITY_SIZE)    :='MyIdentity';
    sCaPath   : STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE):='C:
\TwinCAT\3.1\Target\Certificates\IEC61850\rootCA.pem';
    sCertPath : STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE):='C:
\TwinCAT\3.1\Target\Certificates\IEC61850\127.0.0.1.pem';
    sKeyPath  : STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE):='C:
\TwinCAT\3.1\Target\Certificates\IEC61850\127.0.0.1.key';
    sKeyPwd   : STRING(TCPADS_TLS_KEY_PASSWORD_SIZE)    :='ServerPass';
    sCrlPath  : STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE):='';
    flags     : ST_TlsListenFlags :=DEFAULT_TLSLISTENFLAGS;
END_VAR
```

```
IF bEnable THEN
    Enable:=fbTls.Reset();
    IF bPSK THEN
        fbTls.AddPsk(key:=key, sIdentity:=sIdentity);
    ELSE
        fbTls.AddCa(sCaPath:=sCaPath);
        fbTls.AddCert(sCertPath:=sCertPath, sKeyPath:=sKeyPath, sKeyPwd:=sKeyPwd);
        IF sCrlPath <> '' THEN
            fbTls.AddCrl(sCrlPath:=sCrlPath);
        END_IF
        fbTls.SetListenFlags(flags:=flags);
    END_IF
END_IF
```

Client TLS sample configuration

```
VAR
    key       : ARRAY[0..14] OF BYTE:=[16#1B,16#D0,16#6F,16#D2,16#56,16#16,16#7D,16#C1,16#E8,16#C7,1
6#48,16#2A,16#8E,16#F5,16#FF];
    sIdentity : STRING(TCPADS_TLS_PSK_IDENTITY_SIZE)    :='MyIdentity';
    sCaPath   : STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE):='C:
\TwinCAT\3.1\Target\Certificates\IEC61850\rootCA.pem';
    sCertPath : STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE):='C:
\TwinCAT\3.1\Target\Certificates\IEC61850\client.pem';
    sKeyPath  : STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE):='C:
\TwinCAT\3.1\Target\Certificates\IEC61850\client.key';
    sKeyPwd   : STRING(TCPADS_TLS_KEY_PASSWORD_SIZE)    :='ClientPass';
    sCrlPath  : STRING(TCPADS_TLS_CERTIFICATE_PATH_SIZE):='';
    flags     : ST_TlsConnectFlags :=DEFAULT_TLSCONNECTFLAGS;
END_VAR
```

```
IF bEnable THEN
    Enable:=fbTls.Reset();
    IF bPSK THEN
        fbTls.AddPsk(key:=key, sIdentity:=sIdentity);
    ELSE
        fbTls.AddCa(sCaPath:=sCaPath);
        fbTls.AddCert(sCertPath:=sCertPath, sKeyPath:=sKeyPath, sKeyPwd:=sKeyPwd);
        IF sCrlPath <> '' THEN
            fbTls.AddCrl(sCrlPath:=sCrlPath);
        END_IF
        fbTls.SetConnectFlags(flags:=flags);
    END_IF
END_IF
```

Initialization of the TLS configuration in the IEC 61850 server or client

The “FB_IEDTLS” function block is instantiated in the “TcTelecontrol” global variable list. The TLS configuration settings are passed to the IEC 61850 server or client by setting the “ipTLS” property in the server or client protocol settings. By assigning a valid interface pointer to the “ipTLS” property, the TLS configuration settings and thus TLS encryption are enabled. By assigning a null to the “ipTLS” property, the TLS configuration and encryption are disabled. In this sample, we use the return value of the “Enable” method to enable (return value <> null) or disable (return value = null) TLS encryption. The return value of the “Enable” method is always null if the “bEnable” method parameter is “FALSE”.

Set the server TLS configuration

The server in this sample has the IP address: “127.0.0.1”. This address may need to be adjusted.

```
VAR_GLOBAL
…
    fbIEDTLS: FB_IEDTLS;
    fbIEDServer: FB_iec61850ServerClass := (ipIED:=fbIED, settings:=(bEnable:=TRUE, sLocalHost:='127
.0.0.1', ipTLS:=fbIEDTLS.Enable(bEnable:=TRUE, bPSK:=TRUE));
…
END_VAR
```

Setting the client TLS configuration

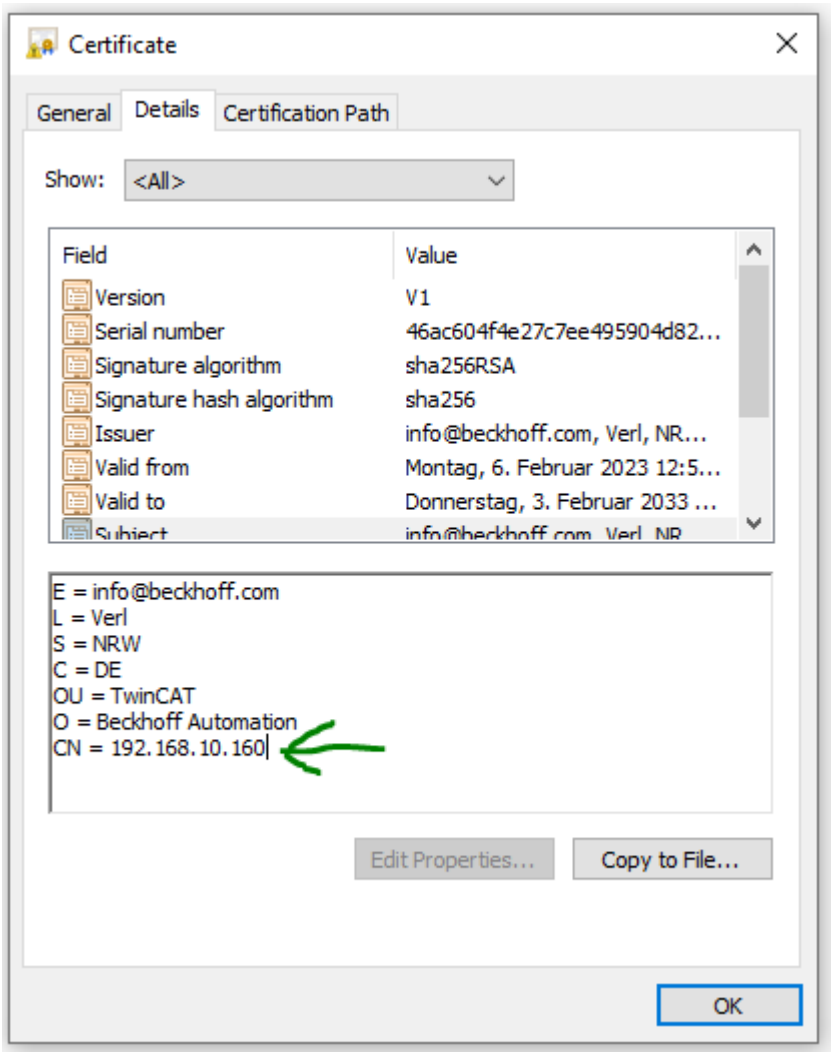
The client in this sample connects to a server with the IP address: “127.0.0.1”. This address may also need to be adjusted.

```
VAR_GLOBAL
…
    fbIEDTLS: FB_IEDTLS;
    fbIEDClient: FB_IEDClient := (fbConnection:=(ipIED:=fbIED, settings:=(sRemoteHost:='127.0.0.1',
ipTLS:=fbIEDTLS.Enable(bEnable:=TRUE, bPSK:=TRUE))));
…
END_VAR
```

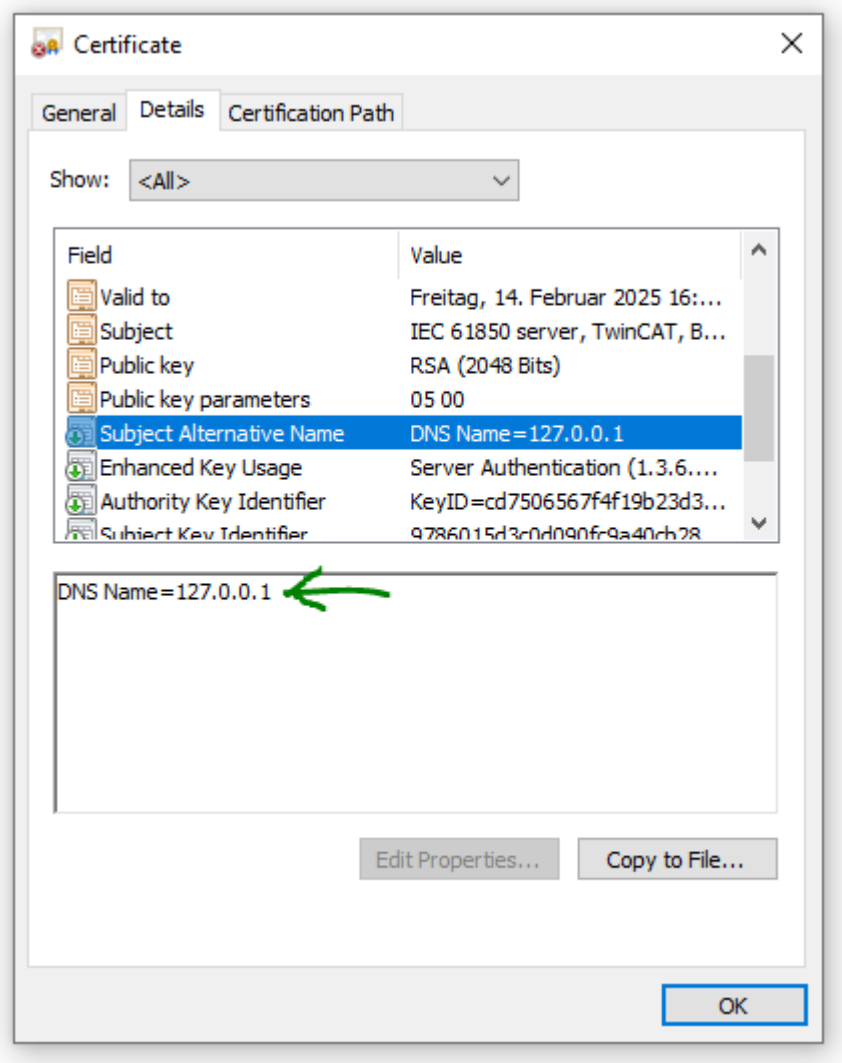
Special features of the server certificate

For TLS data encryption with certificates, the host name or IP address of the server is configured as either a “CommonName” or “Subject Alternative Name” parameter in the certificate. I.e. if you have adjusted the IP addresses in this sample and want to test the TLS encryption with certificates then you must also set the “CommonName” or the “Subject Alternative Name” in your certificate accordingly. Self-signed certificates can be used for testing. Such certificates can be created, for example, under Windows using a Power Shell script. Under Windows, the “CommonName” or the “Subject Alternative Name” can be checked relatively easily.

Sample certificate with “CommonName”=‘192.168.10.160’:



Sample certificate with “Subject Alternative Name” = ‘127.0.0.1’:



Secure ISO TP0 port

It is recommended to use the (Secure ISO TP0) port number specified by IANA for active TLS communication (default value: 3782). Via the property “FB_SocketTlsSettingsClass.nSecurePort” this port number (for the server and client) can be reconfigured to another value. In our implementation, this port number is automatically used once TLS configuration has been enabled by setting the “ipTLS” property in the IEC 61850 client or server function block. If no TLS configuration is active then the port number: “fbIEDServer.settings.nLocalPort” is used for the server and the port number: “fbIEDClient.fbConnection.settings.nRemotePort” is used for the client (default value: 102).

Test of the secure TLS connection with PSK

In both, client/server sample projects, TLS with PSK (Pre-Shared Key) is configured and enabled by default. No certificates are required for TLS with PSK. For a simple function test, only the IP addresses in both projects need to be adjusted.

Test of the secure TLS connection with certificates

If you want to test TLS with certificates, then first copy the certificates and private keys to the server and client target systems.

Server target system:

- Root CA certificate (e.g.: *C:\TwinCAT\3.1\Target\Certificates\IEC61850\rootCA.pem*)
- Server certificate (e.g.: *C:\TwinCAT\3.1\Target\Certificates\IEC61850\server.pem*)
- Private key of the server certificate (e.g.: *C:\TwinCAT\3.1\Target\Certificates\IEC61850\server.key*)

Client target system:

- Root CA certificate (e.g.: *C:\TwinCAT\3.1\Target\Certificates\IEC61850\rootCA.pem*)
- Client certificate (e.g.: *C:\TwinCAT\3.1\Target\Certificates\IEC61850\client.pem*)
- Private key of the client certificate (e.g.: *C:\TwinCAT\3.1\Target\Certificates\IEC61850\client.key*)

The file names above are only example names. You can customize the paths and file names in the “Enable” method of the “FB_IEDTLS” function block. To enable the TLS configuration with certificates, the “bPSK” parameter in the “FB_IEDTLS.Enable” method must be set to “FALSE”.

Enable TLS with certificate in the server project:

```
VAR_GLOBAL
…

fbIEDServer: FB_iec61850ServerClass := (ipIED:=fbIED, settings:=(bEnable:=TRUE, sLocalHost:='192.168
.10.141', ipTLS:=fbIEDTLS.Enable(bEnable:=TRUE, bPSK:=FALSE)));
…
END_VAR
```

Enable TLS with certificate in the client project:

```
VAR_GLOBAL
…

fbIEDClient: FB_IEDClient := (fbConnection:=(ipIED:=fbIED, settings:=(sRemoteHost:='192.168.10.141',
 ipTLS:=fbIEDTLS.Enable(bEnable:=TRUE, bPSK:=FALSE))));
…
END_VAR
```

7.29 Authentication, negotiation of supported services and optional communication parameters

This sample shows how a plain text password can be activated and configured to authenticate the connection in a TwinCAT IEC 61850 client or server application. This sample also shows how the server application can control the authentication process itself at runtime. This means that the application can accept or reject the connection depending on the password received. This sample also shows how the services and parameters supported by the IEC 61850 server are enabled and configured in the TwinCAT client and server application. The sample also shows how the server can control the negotiation of the supported services and parameters itself at runtime. The TwinCAT IEC 61850 server application can, for example, reject the services and/or parameters proposed by the client depending on the password received and thus restrict access to these services or parameters for certain connections or users. If you are using an older version of the TwinCAT IEC 61850 Telecontrol Configurator, the functionalities presented in this sample may not yet be supported. To be able to compile and test this sample without errors, you need a newer version of the TF6510 IEC 61850 Telecontrol.

System requirement

TF6510 IEC 61850 Telecontrol v3.2.98.0 or newer

Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/16992409099.zip

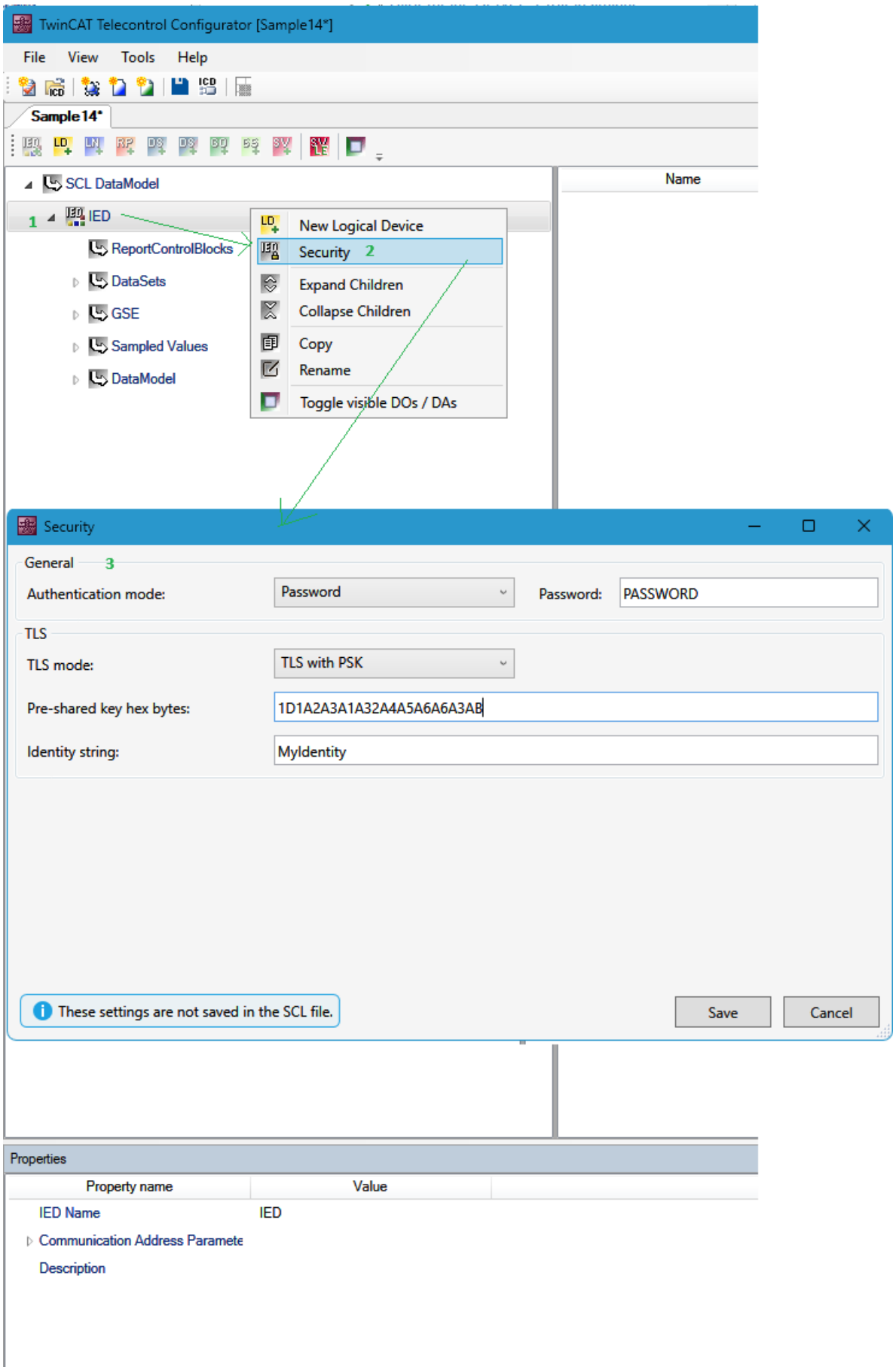
General information about this sample project

The sample consists of two separate TwinCAT IEC 61850 projects. After unpacking the zip file, you will find a TwinCAT IEC 61850 client project in the subfolder *\Client\Sample31* and a TwinCAT IEC 61850 server project in the subfolder *\Server\Sample31*. Both projects are coordinated, only the IP address of the server has to be adjusted in the code of both projects. The client will establish a connection to the server when you download, activate and start the projects on two TwinCAT target systems connected via LAN.

The Client - basic sample project [▶ 669] and Server - basic sample project [▶ 704] serve as the basis for the sample project.

Activation and configuration of the authentication function in the TwinCAT Telecontrol Configurator

In the simplest variant, the authentication function can already be activated in the TwinCAT Telecontrol Configurator. The settings required for this can be found in the Security configuration dialog. This dialog can be accessed via the context menu of the IED device (right-mouse click on the IED device and then select Security from the menu). As shown in the image below, please set the parameter: **Authentication mode** to the value: **Password**.


TwinCAT Telecontrol Configurator [Sample14*]
File View Tools Help
Sample14*
SCL DataModel
1 IED
ReportControlBlocks
DataSets
GSE
Sampled Values
DataModel
New Logical Device
Security 2
Expand Children
Collapse Children
Copy
Rename
Toggle visible DOs / DAs
Name
Security
General 3
Authentication mode: Password
Password: PASSWORD
TLS
TLS mode: TLS with PSK
Pre-shared key hex bytes: 1D1A2A3A1A32A4A5A6A6A3AB
Identity string: MyIdentity
These settings are not saved in the SCL file.
Save
Cancel
Properties
Property name
Value
IED Name
IED
Communication Address Parameter
Description

The IEC 61850 client or server project code generation generates the code required to activate the authentication function and configures a standard password with the value: “PASSWORD” and a standard mechanism with the name: “2.2.3.1”. Bit 0 of the protocol setting: “nRequirements” activates the authentication function.

Standard authentication parameters in the generated TwinCAT IEC 61850 project

Namespace: TcTelecontrol

Type: Global Variable List (GVL)

Client project:

```
VAR_GLOBAL
    ipCreator: I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=5).SetGuiVer(major:=3, minor:=2, build:=97, revision:=7);

    (* IED data model *)
    fbIED: FB_IED_IED;
    {warning 'Please configure your server IP-address in the initialization code of the client function block below:'}
    {warning 'Please configure your authentication password in the initialization code of the client function block below:'}
    (* Client connection *)
    fbIEDClient: FB_IEDClient := (fbConnection:=(ipIED:=fbIED, settings:=(sRemoteHost:='127.0.0.1', sAuthent_Value:='PASSWORD', sMechanism_Name:='2.2.3.1', nRequirements:=16#01)));
END_VAR
```

Server project:

```
VAR_GLOBAL
    ipCreator: I_AcsiCodeCreatorClass := GVL_AcsiVars.Creator.SetCodeRev(codeRev:=5).SetGuiVer(major:=3, minor:=2, build:=97, revision:=7);

    (* IED data model *)
    fbIED: FB_IED_IED;
    {warning 'Please configure your server IP-address in the initialization code of the server function block below:'}
    {warning 'Please configure your authentication password in the initialization code of the server function block below:'}
    (* Server container *)
    fbIEDServer: FB_iec61850ServerClass := (ipIED:=fbIED, settings:=(bEnable:=TRUE, sLocalHost:='127.0.0.1', sAuthent_Value:='PASSWORD', sMechanism_Name:='2.2.3.1', nRequirements:=16#01));
    (* Server connections *)
    fbIEDServerSession1: FB_IEDServerSession := (fbConnection:=(ipServer:=fbIEDServer, settings:=(bEnable:=TRUE)));
    (*
    fbIEDServerSession2: FB_IEDServerSession := (fbConnection:=(ipServer:=fbIEDServer, settings:=(bEnable:=TRUE)));
    ...
    fbIEDServerSessionN: FB_IEDServerSession := (fbConnection:=(ipServer:=fbIEDServer, settings:=(bEnable:=TRUE)));
    *)
END_VAR
```

The default values can only be adjusted in the generated PLC code and not in the TwinCAT Telecontrol Configurator, as the authentication password and the mechanism name cannot be saved in the SCL file. Currently, the TwinCAT IEC 61850 implementation only supports the “Password” authentication method defined in the SCL. Other authentication methods “weak”, “strong” or “certificate” are not supported. However, you can use TLS to encrypt all client-server communication. You can find a sample of this here: TLS - Secure client-server communication with the help of certificates or PSK [▶ 742].

Unilateral vs. mutual authentication mode

The unilateral authentication mode (one-way authentication) is the standard mode for IEC 61850 authentication and is also supported by most devices. The mutual authentication mode (two-way authentication) is rather uncommon. With unilateral authentication, the password is only sent in one direction, from the client to the server. The server verifies the password and, if successful, the connection is maintained (accepted) or closed (rejected) in the event of an error. With mutual authentication, both communication partners verify each other’s password. First, the server verifies the password received from the client. If successful, the server also sends a password back to the client as a response. The client also verifies the password received from the server and, if successful, the connection is maintained (accepted) or closed (rejected) if an error occurs. The unilateral authentication mode is configured by default in the TwinCAT IEC 61850 implementation.

Configuration of the optional client and server communication parameters

Some communication parameters in the protocol settings are optional. It is not absolutely necessary for these parameter values to be transmitted. Nevertheless, it is possible that missing optional communication parameters may lead to the connection being terminated or communication errors. The connection request is often rejected by the other communication partner due to missing optional parameters. There are also devices that respond to the receipt of the optional parameters by terminating the connection. In such cases, you can customize the protocol settings and activate or deactivate the transmission of the optional communication parameters according to the requirements. Adjust the protocol settings in the global variable list “TcTelecontrol” of the server or client project. In the sample projects described here (server and client), the transmission of the following optional parameters is activated:

- “nCalled_AP_InvID:=0”

- “nCalled_AE_InvID:=0”
- “nCalling_AP_InvID:=0”
- “nCalling_AE_InvID:=0”

The value: 0 (null) is a valid value for these parameters and this value is also transmitted. The value: 16#FFFFFFFF configures a parameter as optional and in this case no value is transmitted.

Simple authentication with a master password

In the simplest variant of authentication, the server only accepts a single authentication password. In this case, you only need to customize the already generated standard password: “PASSWORD” in the created server or client application. For this functionality, no further changes or extensions to the generated PLC code are required for the time being.

This one password can also be referred to as the master password. Any client using this password can successfully connect to the server. If your server implementation allows multiple connections, then multiple clients can also establish a connection to the server with the same master password. In the documented server sample, the main password is: “GeneralPass”. This password can be configured via the server’s protocol settings. In the sample code, you will find the parameter value in the global variable list: “TcTelecontrol” (see image below). The important/relevant code sections are highlighted in bold.

Namespace: TcTelecontrol

Type: Global Variable List (GVL)

```
VAR_GLOBAL

    (*…*)

    (* IED data model *)
    fbIED: FB_IED_IED;
    bMutual: BOOL:=FALSE;
    (* Server container *)
    fbIEDServer: FB_iec61850ServerClass:=(
        ipIED:=fbIED,
        settings:=( bEnable:=TRUE,
                    sLocalHost:='127.0.0.1',
                    sAuthent_Value:='GeneralPass',
                    sMechanism_Name:='2.2.3.1',
                    nRequirements:=16#01,
                    eAcse_Authent:=SEL(bMutual, E_AcseAuthentication.Unilateral, E_AcseAuthenticatio
n.Mutual),
                    sResponding_Value:='RespondingPass',
                    nCalled_AP_InvID:=0,
                    nCalled_AE_InvID:=0,
                    nCalling_AP_InvID:=0,
                    nCalling_AE_InvID:=0));
    (* Server connections *)
    fbIEDServerSession1: FB_IEDServerSession:=(
        fbConnection:=(
            ipServer:=fbIEDServer,
            settings:=( bEnable:=TRUE,
                        sAuthent_Value:='Session1Pass')));

    (*…*)

END_VAR
```

Authentication with a connection-specific password

In the TwinCAT IEC 61850 server implementation, an additional connection-specific password can optionally be configured for each available server connection in addition to the main password. This password can be configured via the configuration settings of the server connection (e.g. “fbIEDServerSession1.fbConnection.settings”). The server in our sample above only accepts one connection. The configured connection-specific password is “Session1Pass”. The server only accepts the server connection if the client sends either the main password or the configured connection-specific password via this server connection (in the sample above “GeneralPass” or “Session1Pass”). For servers with only one configured (permitted) server connection, you can assign a second password equivalent to the main password in this way and use it like a main password.

During the connection establishment at the TCP/IP level, a free (unoccupied) server connection instance is assigned to a client that wants to connect to the server by default. However, it is possible to bind a client to a specific server connection via the client's IP address. In the example below, a client with the IP address "192.168.10.140" is connected to the connection instance "fbIEDServerSession1" and another client with the IP address "192.168.10.160" to the connection instance "fbIEDServerSession2".

A client with the IP address: "192.168.10.140" can only connect to the instance "fbIEDServerSession1" if it sends the master password or the password "Session1Pass". The client with the IP address "192.168.10.160" can only connect to the instance "fbIEDServerSession2" if it sends the master password or the password "Session2Pass".

Namespace: TcTelecontrol

Type: Global Variable List (GVL)

```
VAR_GLOBAL

    (*…*)

    fbIEDServerSession1: FB_IEDServerSession:=(
        fbConnection:=(
            ipServer:=fbIEDServer,
            settings:=(bEnable:=TRUE,
                       sAuthent_Value:='Session1Pass',
                       sAcceptHost:='192.168.10.140',
                       eAccept:=E_SocketAcceptMode.eACCEPT_SEL_HOST)));

    fbIEDServerSession2: FB_IEDServerSession:=(
        fbConnection:=(
            ipServer:=fbIEDServer,
            settings:=(bEnable:=TRUE,
                       sAuthent_Value:='Session2Pass',
                       sAcceptHost:='192.168.10.160',
                       eAccept:=E_SocketAcceptMode.eACCEPT_SEL_HOST)));

    (*…*)

END_VAR
```

Authentication with several additional passwords

If the server is to accept multiple passwords, or the client is to establish the connection to the server not only with one password, but with several different passwords (sometimes with one, sometimes with the other), then you must add a few lines to the generated code yourself.

Six different sample passwords were defined in the "passList" array variable of the client sample project in order to present the different authentication options:

- "GeneralPass"
- "Session1Pass"
- "AnotherPass"
- "MorePass"
- "FurtherPass"
- "UnknownPass"

The variable "passIndex" selects the password that the client should use the next time a connection is established (see below). The global variable "bMutual" can be used to switch between the two authentication modes unilateral and mutual. By default, unilateral authentication mode is configured in the sample.

Namespace: TcTelecontrol

Type: Global Variable List (GVL)

```
VAR_GLOBAL

    (*…*)

    (* IED data model *)
    fbIED: FB_IED_IED;
    bMutual: BOOL:=FALSE;
    passIndex: DINT(0..5):=0;
```

```
    (* Authentication passwords *)
    passList: ARRAY[0..5] OF STRING:=['GeneralPass',
                                      'Session1Pass',
                                      'AnotherPass',
                                      'MorePass',
                                      'FurtherPass',
                                      'UnknownPass'];
    (* Client connection *)
    fbIEDClient : FB_IEDClient:=(
        fbConnection:=( ipIED:=fbIED,
                        settings:=( sRemoteHost:='127.0.0.1',
                                    sAuthent_Value:=passList[LIMIT(0, passIndex, 5)],
                                    sMechanism_Name:='2.2.3.1',
                                    nRequirements:=16#01,
                                    eAcse_Authent:=SEL(bMutual, E_AcseAuthentication.Unilateral, E_A
cseAuthentication.Mutual),
                                    sResponding_Value:='RespondingPass',
                                    nCalled_AP_InvID:=0,
                                    nCalled_AE_InvID:=0,
                                    nCalling_AP_InvID:=0,
                                    nCalling_AE_InvID:=0)));

    (*…*)

END_VAR
```

You will find some additional variables in both the client and the server sample application. These variables, such as “bAccepted”, “eAareResult”, “eAareDiagnostic” or “sPassword” can be used for visual monitoring in online mode or for diagnostic purposes. The values of these variables are set, for example, in the client sample project in the implementation code of the “OnAssociateCnf” event handling routine of the “FB_IEDClient” function block. In the server sample project, these values are set in the implementation code of the “OnAssociateInd” event handling routine of the “FB_IEDServerSession” function block.

```
FUNCTION_BLOCK FB_IEDClient IMPLEMENTS I_ScsmAbortIndEventSink,
                    I_ScsmAssociateCnfEventSink,
                    I_ScsmGetServerDirectoryCnfEventSink,
                    I_ScsmGetLogicalDeviceDirectoryCnfEventSink,
                    I_ScsmGetLogicalNodeDirectoryCnfEventSink,
                    I_ScsmGetDataValuesCnfEventSink,
                    I_ScsmSetDataValuesCnfEventSink,
                    I_ScsmReportIndEventSink,
                    I_ScsmControlCnfEventSink,
                    I_ScsmLastApplErrorIndEventSink,
                    I_ScsmCommandTerminationIndEventSink
VAR_INPUT
    fbConnection : FB_iec61850ClientClass:=(ipAbortInd:=THIS^,
                        ipAssociateCnf:=THIS^,
                        ipGetServerDirectoryCnf:=THIS^,
                        ipGetLogicalDeviceDirectoryCnf:=THIS^,
                        ipGetLogicalNodeDirectoryCnf:=THIS^,
                        ipGetDataValuesCnf:=THIS^,
                        ipSetDataValuesCnf:=THIS^,
                        ipReportInd:=THIS^,
                        ipControlCnf:=THIS^,
                        ipLastApplErrorInd:=THIS^,
                        ipCommandTerminationInd:=THIS^);
END_VAR
VAR

    (*…*)

    bAccepted       : BOOL:=FALSE;
    sPassword       : STRING:='';
    sMechanismName  : STRING:='';
    sRespondingPass : STRING:='';
    eAareResult     : E_ACSE_AareResult:=E_ACSE_AareResult.Accepted;
    eAareDiagnostic : E_ACSE_AareDiagnostic:=E_ACSE_AareDiagnostic.Null;
    eAbrtDiagnostic : E_ACSE_AbrtDiagnostic:=E_ACSE_AbrtDiagnostic.Null;

    (*…*)

END_VAR
```

```
    (*…*)

CASE state OF
    0:
        IF _bAbort THEN
```

```
            _bAbort:= FALSE;
            bSuccess:= fbConnection.AbortReq(ipReason:=fbAbortReason, ipSink:=0, ipResult=>ipResult)
;
            state:= SEL(bSuccess, 100, 1);
        ELSIF eState = E_AsyncEnvironmentState.Idle AND (_bConnect OR _bReconnect) THEN
            _bConnect:= FALSE;
            bGetAllServerValues:= SEL(_bReadAllData, bGetAllServerValues, TRUE);

            fbConnection.settings.eAcse_Authent:= SEL(TcTelecontrol.bMutual, E_AcseAuthentication.Un
ilateral, E_AcseAuthentication.Mutual);
            fbConnection.settings.sAuthent_Value:= TcTelecontrol.passList[LIMIT(0, passIndex, 5)];
            bSuccess:= fbConnection.AssociateReq(ipSink:=0, ipResult=>ipResult);
            state:= SEL(bSuccess, 100, 1);

    (*…*)

END_CASE
```

Control of the authentication process in the server application

The server in the standard implementation automatically verifies the mechanism name and the received password internally. If the result of this check is positive, the connection is accepted (maintained) by the server and rejected (closed) if the result is negative. This functionality does not require any special extensions to the automatically generated PLC code. However, it is also possible to actively control the authentication process at runtime in the server application. This means that the server can decide itself (spontaneously) whether to accept or reject a connection. In this case, if conditions allow, a connection rejected by internal verification can still be accepted, and an internally accepted connection can still be rejected.

As an example of the unilateral authentication mode with additional passwords, the following behavior is implemented in the server sample:

- With the main password “GeneralPass”, the server should accept the connection. This password is configured in the global variable list “TcTelecontrol” and the server would have accepted the password even without a special code extension.
- The client password “Session1Pass” should be rejected by the server. In the standard implementation, the server would have accepted the connection as this password is also configured as a connection-specific password in the global variable list “TcTelecontrol”.
- Three additional client passwords “AnotherPass”, “MorePass” and “FurtherPass” should be accepted by the server. In the standard implementation, the internal verification would have rejected the connection requests with these passwords.
- The connection with the password “UnknownPass” and all other unknown passwords should be rejected by the server. The internal verification in the standard implementation would also have rejected the connection requests with these passwords.

An additional PLC code in the “OnAssciateInd” event handling routine allows the server application to subsequently change the result of the internal verification; this is used in the server sample. The values of the additional local variables of the “FB_IEDServerSession” function block are set in the implementation code of the event handling routine: “OnAssociateInd”.

```
FUNCTION_BLOCK FB_IEDServerSession IMPLEMENTS I_ScsmAbortIndEventSink, I_ScsmAssociateIndEventSink,
I_ScsmReleaseIndEventSink
VAR_INPUT
    fbConnection : FB_iec61850ConnectionClass:=(ipAbortInd:=THIS^,
                                        ipAssociateInd:=THIS^,
                                        ipReleaseInd:=THIS^);
END_VAR
VAR

    (*…*)

    bAccepted       : BOOL:=FALSE;
    sPassword       : STRING:='';
    sMechanismName  : STRING:='';
    sRespondingPass : STRING:='';
    eAareResult     : E_ACSE_AareResult:=E_ACSE_AareResult.Accepted;
    eAareDiagnostic : E_ACSE_AareDiagnostic:=E_ACSE_AareDiagnostic.Null;
    eAbrtDiagnostic : E_ACSE_AbrtDiagnostic:=E_ACSE_AbrtDiagnostic.Null;

    (*…*)

END_VAR
```

The Boolean variable “bAccepted” will be “TRUE” if the internal verification accepts the authentication data and thus also the connection. However, this variable will also be “TRUE” if the server authentication function has not been activated in the protocol settings.

User-defined OnAssociateInd event handling routine of the server

The result of the internal verification can only be subsequently influenced if the authentication function of both communication partners is active and all other framework conditions (communication parameters such as mechanism name and password) have been correctly configured or received. The server application can access the interface pointer “ipAuthentication” of type I_ScsmAuthenticationClass [▶ 440] via the input variable “ipAA” of the event handling routine “OnAssociateInd”. The interface pointer belongs to the authentication object, which also carries out the verification internally. The internal verification is already carried out by the server before the “OnAssociateInd” event handling routine is called. The first result is already known when the method is called. The interface pointer “ipAuthentication” provides the application methods to query the result of the verification, to accept a connection rejected by the internal verification, or to reject an accepted connection. A connection that has already been rejected by internal verification cannot be rejected again and a connection that has been accepted by internal verification cannot be accepted again. In this case, calls to the “Accept” or ‘Reject’ methods will return a value of “FALSE” (error).

```
METHOD OnAssociateInd : BOOL
VAR_INPUT
    ipAA     : I_ScsmAssociationClass;
END_VAR
VAR_OUTPUT
    ipResult : I_AsyncServiceResultClass;
END_VAR
```

```
(*…*)

bAccepted:= FALSE;
sMechanismName:= '';
sPassword:= '';
sRespondingPass:= '';
eAareResult:= E_ACSE_AareResult.Accepted;
eAareDiagnostic:= E_ACSE_AareDiagnostic.Null;
eAbrtDiagnostic:= E_ACSE_AbrtDiagnostic.Null;
IF (ipAA <> 0) AND_THEN (ipAA.ipAuthentication <> 0) THEN
    IF ipAA.ipAuthentication.GetVerifyResult(bAccepted=>bAccepted,
                            sMechanismName=>sMechanismName,
                            sPassword=>sPassword,
                            sRespondingPass=>sRespondingPass,
                            eResult=>eAareResult,
                            eDiagnostic=>eAareDiagnostic,
                            eAbrtDiagnostic=>eAbrtDiagnostic)
    THEN
        IF bAccepted THEN
            IF (sPassword = 'GeneralPass') THEN
                ;
            ELSIF (sPassword = 'Session1Pass') THEN
                bSuccess:= ipAA.ipAuthentication.Reject();
            ELSE
                ;
            END_IF
        ELSE
            IF (sPassword = 'AnotherPass') OR_ELSE (sPassword = 'MorePass') THEN
                bSuccess:= ipAA.ipAuthentication.Accept(sRespondingPass:='RespondingPass');
            ELSIF (sPassword = 'FurtherPass') THEN
                bSuccess:= ipAA.ipAuthentication.Accept(sRespondingPass:='InvalidRespondingPass');
            ELSIF (sPassword = 'UnknownPass') THEN
                ;
            ELSE
                ;
            END_IF
        END_IF
        ipAA.ipAuthentication.GetVerifyResult(bAccepted=>bAccepted,
                            sMechanismName=>sMechanismName,
                            sPassword=>sPassword,
                            sRespondingPass=>sRespondingPass,
                            eResult=>eAareResult,
                            eDiagnostic=>eAareDiagnostic,
                            eAbrtDiagnostic=>eAbrtDiagnostic);
    END_IF

    (* Negotiation example of max. number of outgoing called and calling services: *)
    ipAA.ipAuthentication.nNegotiatedMaxServOutCalled:= 5; (* Default: 10 *)
```

```
    ipAA.ipAuthentication.nNegotiatedMaxServOutCalling:= 5; (* Default: 10 *)

    (* Example of supported services negotiation: *)
    IF ipAA.ipAuthentication.ipNegotiatedServicesSupported <> 0 THEN
        (* Disable support for file services *)
        ipAA.ipAuthentication.ipNegotiatedServicesSupported.bFileClose:= FALSE;
        ipAA.ipAuthentication.ipNegotiatedServicesSupported.bFileDelete:= FALSE;
        ipAA.ipAuthentication.ipNegotiatedServicesSupported.bFileDirectory:= FALSE;
        ipAA.ipAuthentication.ipNegotiatedServicesSupported.bFileOpen:= FALSE;
        ipAA.ipAuthentication.ipNegotiatedServicesSupported.bFileRead:= FALSE;
        ipAA.ipAuthentication.ipNegotiatedServicesSupported.bFileRename:= FALSE;
        ipAA.ipAuthentication.ipNegotiatedServicesSupported.bObtainFile:= FALSE;
    END_IF
END_IF

(*…*)

OnAssociateInd:= fbConnection.AssociateRsp(ipResult=>ipResult);
```

In this server sample, the “GetVerifyResult” method call first reads the result of the internal verification and some diagnostic information. The internally already accepted connection with the password “Session1Pass” is then rejected with the “Reject” method call and the internally already rejected connections with the passwords “AnotherPass”, “MorePass” and “FurtherPass” are still accepted with the “Accept” method call. Finally, the verification result is read again. Only then may the response be sent to the client with the “AssociateRsp” method call.

Dynamic negotiation of supported server services and parameters

In most use cases, the services and parameters supported by the server can be configured via protocol settings in the global variable list “TcTelecontrol”. The settings made there then apply to all connections on the server. This means that all clients can only negotiate with the server those services that have been globally configured (activated) for all connections.

In some use cases, however, it may be necessary to negotiate different server services and parameters depending on the client connection (e.g. depending on the client’s IP address or authentication password). The server could allow/support more services with a specific client connection and fewer with another client connection.

The event handling routine “OnAssociateInd” can be used to negotiate the supported services and parameters at runtime (connection-dependent). A server application can access the internal result of the negotiated services and parameters via the interface variable “ipAA.ipAuthentication”. The result can still be manipulated as required at this stage. In the sample code above, the maximum number of incoming and outgoing confirmed requests (MMS request) is limited from the default value of 10 to 5 and support for all MMS file services is disabled. Only then is the new result sent back to the client in the answer telegram.

Simple authentication in mutual authentication mode

The mutual authentication mode must be supported by both communication partners and is rather unusual in IEC 61850. In this mode, the server sends a password back to the client as a response. This response password is already configured in the server sample project with the value “RespondingPass” in the global variable list “TcTelecontrol”. In the client sample project, a password “RespondingPass” is also configured in the global variable list “TcTelecontrol”. This is the password that the client expects as a response from the server. The client’s internal verification function will compare this password with the password received from the server. If successful, the connection is maintained and, if unsuccessful, it is closed again. In the simplest configuration, the response passwords only need to be adjusted/configured in the generated server and client sample projects and the mutual authentication mode activated in the protocol settings. Additional code in the “OnAssociateInd” or “OnAssociateCnf” event handling routines is not required for mutual authentication in the basic function.

User-defined OnAssociateCnf event handling routine of the client

In mutual authentication mode, the client, like the server, can also actively control the authentication process at runtime. However, the server must have already accepted the password received from the client. A connection rejected by the server (negative response) can no longer be accepted on the client side. In unilateral mode, it is not necessary to implement the “OnAssociateCnf” event handling routine. The user-defined implementation of the “OnAssociateCnf” event handling routine presented here is only useful in mutual authentication mode.

To present the various available options, the following behavior was implemented in the sample:

- The server responds to the client password “FurtherPass” with an incorrect password “InvalidRespondingPass”. In the standard implementation, the client would have closed the connection again due to the incorrect response. In our sample, the wrong password is accepted and the connection is maintained.
- With the client password “Session1Pass”, “UnknownPass”, the connection is already rejected by the server.
- The server responds to all other client passwords with the password “RespondingPass”. With the client password “MorePass”, however, the client should still reject the connection.

```
METHOD OnAssociateCnf
VAR_INPUT
    ipAA   : I_ScsmAssociationClass;
    eError : E_AcsiServiceError;
END_VAR
VAR
END_VAR
```

```
sLastAbortReason:= '';
sLastErrorResult:= '';

bAccepted:= FALSE;
sMechanismName:= '';
sPassword:= '';
sRespondingPass:= '';
eAareResult:= E_ACSE_AareResult.Accepted;
eAareDiagnostic:= E_ACSE_AareDiagnostic.Null;
eAbrtDiagnostic:= E_ACSE_AbrtDiagnostic.Null;
IF (ipAA <> 0) AND_THEN (ipAA.ipAuthentication <> 0) THEN
    IF ipAA.ipAuthentication.GetVerifyResult(bAccepted=>bAccepted,
                            sMechanismName=>sMechanismName,
                            sPassword=>sPassword,
                            sRespondingPass=>sRespondingPass,
                            eResult=>eAareResult,
                            eDiagnostic=>eAareDiagnostic,
                            eAbrtDiagnostic=>eAbrtDiagnostic)
    THEN
        IF bAccepted THEN
            IF (sPassword = 'GeneralPass')
            OR_ELSE (sPassword = 'AnotherPass') THEN
                ;
            ELSIF (sPassword = 'MorePass') THEN
                bSuccess:= ipAA.ipAuthentication.Reject();
            ELSE
                ;
            END_IF
        ELSE
            IF (sPassword = 'FurtherPass') THEN
                bSuccess:= ipAA.ipAuthentication.Accept(sRespondingPass:='RespondingPass');
            ELSE
                ;
            END_IF
        END_IF

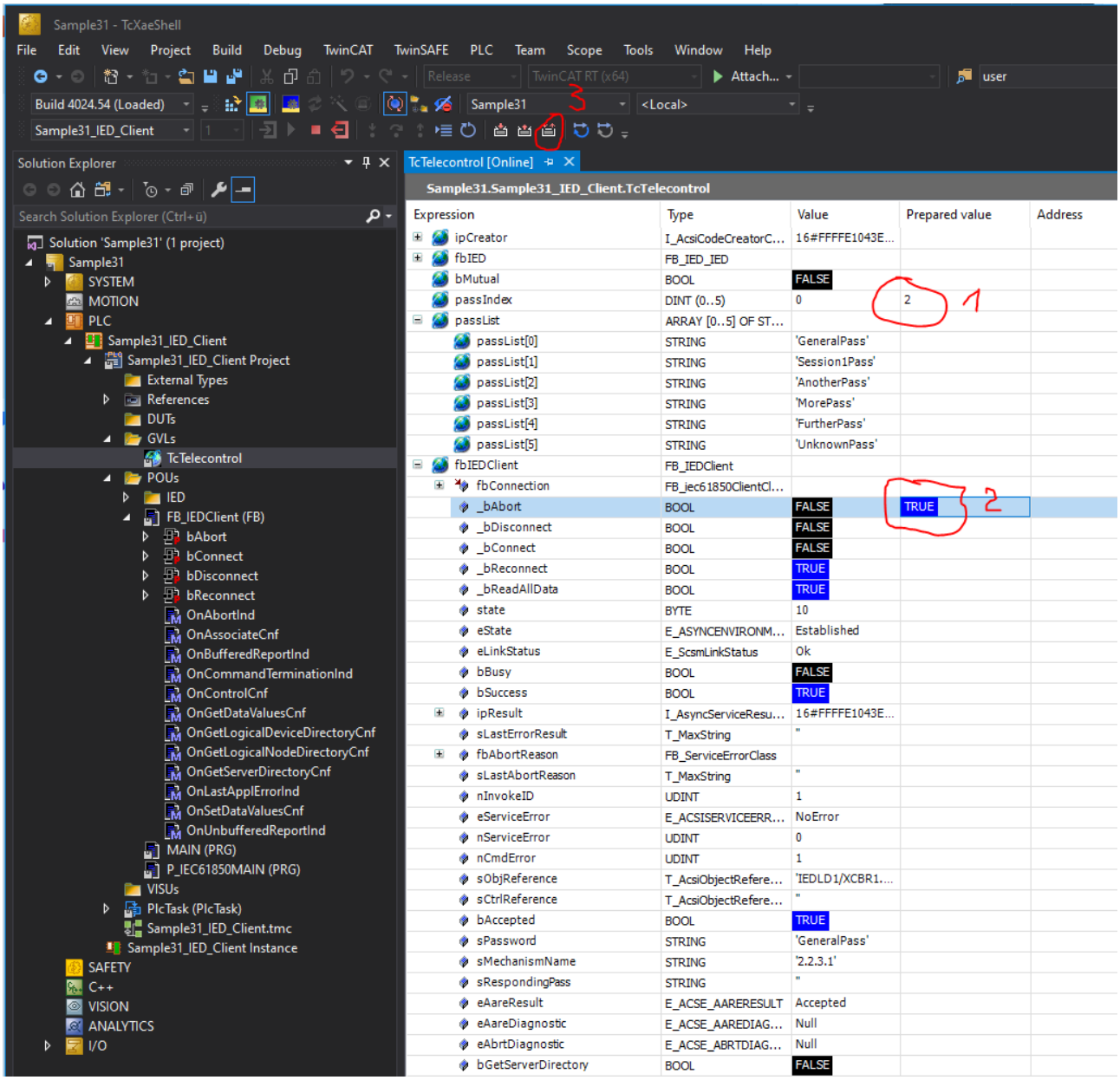
        ipAA.ipAuthentication.GetVerifyResult(bAccepted=>bAccepted,
                            sMechanismName=>sMechanismName,
                            sPassword=>sPassword,
                            sRespondingPass=>sRespondingPass,
                            eResult=>eAareResult,
                            eDiagnostic=>eAareDiagnostic,
                            eAbrtDiagnostic=>eAbrtDiagnostic);
    ELSE
        ;
    END_IF
END_IF
```

As with the “OnAssociateInd” event handling routine of the server described above, the client application can also access the “ipAuthentication” interface pointer of type I_ScsmAuthenticationClass [▶ 440] via the “ipAA” input variable of the “OnAssociateCnf” event handling routine. With the “GetVerifyResult” method call, the client application can query the result of the internal verification and modify it if necessary.

Test

To be able to carry out a simple test, the network address (IPv4 address) of the TwinCAT IEC 61850 server must first be adapted in the server and client sample project. If the client and server sample project is downloaded and started on two TwinCAT target systems, the client will use the “GeneralPass” password for authentication and establish a connection to the server. The password used is selected in the global variable list “TcTelecontrol” of the client project using the index variable “passIndex” from the array “passList”. In the sample, the variable “passIndex” has the initial value “0” and selects the first password from the “passList” array with the value “GeneralPass”.

Next, you can set the password index variable “passIndex” to the value “2” and the Boolean variable “fbIEDClient._bAbort” to the value “TRUE” in online mode, for example (see image below).



The client will first close the old connection and then establish a new one with the password “AnotherPass”. In this way, you can try out the different passwords implemented in the sample. If the “passIndex” value is 1 or 5, for example, the server will reject the connection request. Please note that it may take a few seconds before a new connection can be established. By default, the minimum time for re-establishing a connection is set to 45 seconds in the configuration settings “fbIEDClient.fbConnection.settings.tConnect”.

7.30 SV Publisher/Subscriber according to UCA 61850-9-2 Light Edition - Transmission of sampled measured values

This example shows the TwinCAT IEC 61850 implementation of the transmission of sampled digitized measured values using SV messages (Sampled Value Messages) according to UCA (Utility Communications Architecture) 61850-9-2LE (Light Edition) [1].

The Light Edition (9-2LE) is not a standard, but a technical implementation guide for digital data exchange. This guideline is intended to ensure interoperability between devices from different manufacturers and defines a subset of the IEC 61850-9-2 standard. The implementation of this subset should contribute to a faster market launch of the standard.

System requirement

TwinCAT TF6510 IEC 61850 Telecontrol v3.3.99.0 or higher.

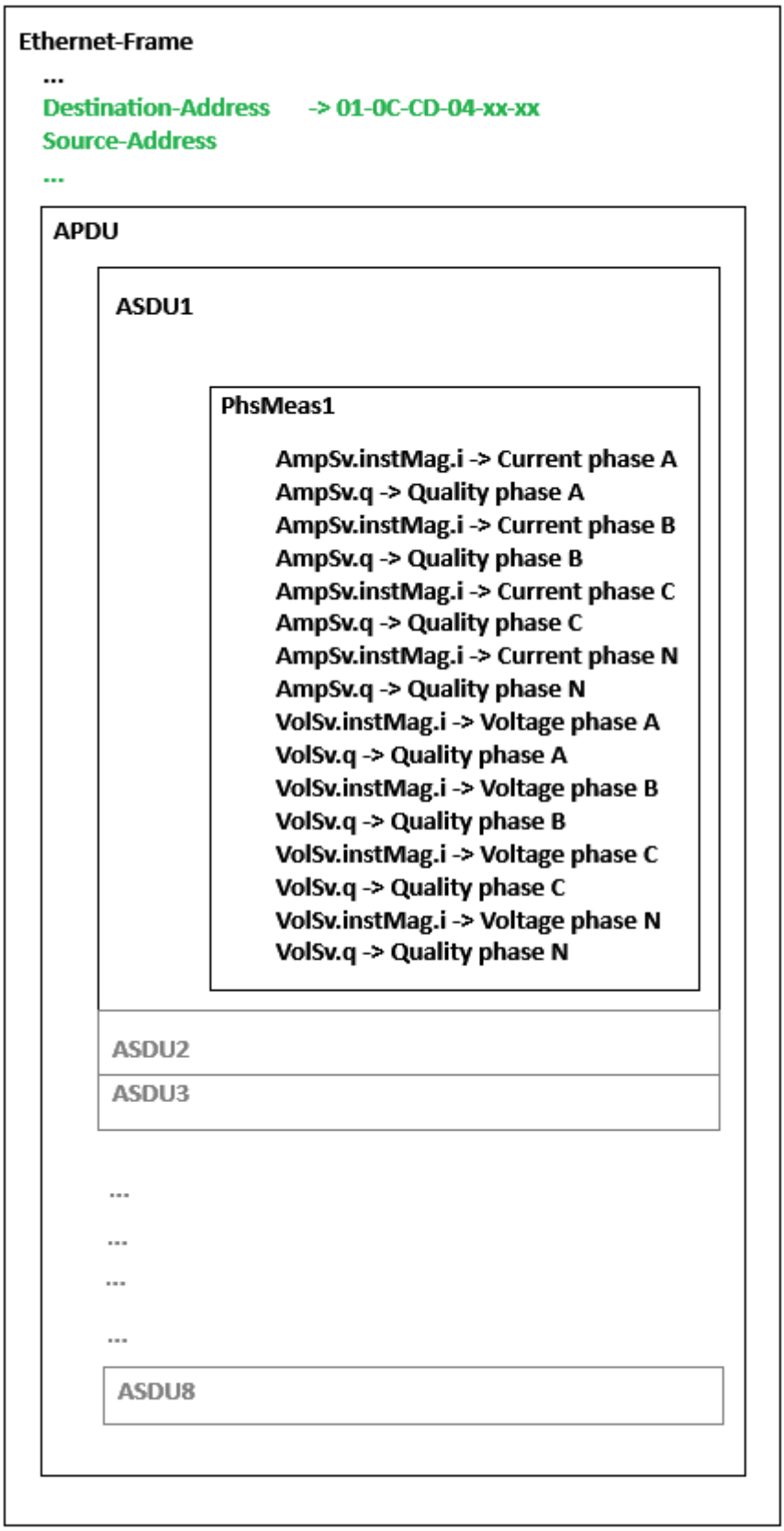
Download TwinCAT XAE Project (*.zip): https://infosys.beckhoff.com/content/1033/TF6510_TC3_IEC61850/Resources/20981055883.zip

References

[1] IEC 61850-9-2LE (Light Edition) Implementation Guideline for Digital Interface to Instrument Transformers using IEC 61850-9-2, UCA International Users Group, www.ucaiug.org.

General information about this sample project

The 9-2LE is a UCA profile in which the data set (ASDU) of the SV message (APDU) is limited to a maximum of 4 voltage values and 4 current values (4 currents and 4 voltages of phase A, B, C and the neutral conductor (N)). Each ASDU contains the 4 three-phase current and voltage measurement values. An SV message (APDU) can contain either 1 or 8 data records (ASDUs). The quality information is also transmitted in the ASDU for each measured current and voltage value. The figure below shows a simplified representation of the SV message within an Ethernet frame.



The sample consists of two separate TwinCAT IEC 61850 projects. After unpacking the zip archive, you will find a TwinCAT IEC 61850 Server (SV Publisher) in the subfolder |*Sample14*|*Publisher* and a TwinCAT IEC 61850 Client (SV Subscriber) project in the subfolder |*Sample14*|*Subscriber*. Both projects are coordinated with each other.

The IP address of the server must be adapted in the code of both projects. The client will establish a connection to the server when you download, activate and start the projects on two TwinCAT target systems connected via LAN. However, the configuration of the IP address and the existence of the client-server connection are not mandatory for SV transmission. SV communication takes place using a publisher-subscriber model. The SV communication stack is mapped directly to the data link layer (layer 2) for maximum speed.

In the project tree under the I/O-Device branch you will find a network adapter instance named "GSE (RT Ethernet adapter)". This adapter instance must be configured accordingly, i.e. the I/O configuration must be adapted to the existing hardware and to the target platform on which the project is to run. A new I/O configuration is also necessary if you change the target platform. This configuration must be done manually in TwinCAT XAE. In addition to the I/O configuration of the network adapter, a link must be established between the network adapter and the PLC function blocks for SV communication. This link allows SV messages received by the network adapter to be forwarded to the instance of the SV Subscriber function block. In the opposite direction, the instance of the SV Publisher function block can forward the SV messages to be sent to the network adapter. You can find more information on the topic here: RT Ethernet adapter Configuration [▶ 35].

The data model of the merging unit

According to 9-2LE, the data model requires 4 instances of the logical node: TCTR (current transformer) and 4 instances of the logical node: TVTR (voltage transformer). The sample project contains the following LNs:

Logical node	Type	Description
InnATCTR1	TCTR	Current transformer, phase A
InnBTCTR2	TCTR	Current transformer, phase B
InnCTCTR3	TCTR	Current transformer, phase C
InnNTCTR4	TCTR	Current transformer, neutral conductor N
UnnATVTR1	TVTR	Voltage transformer, phase A
UnnBTVTR2	TVTR	Voltage transformer, phase B
UnnCTVTR3	TVTR	Voltage transformer, phase C
UnnNTVTR4	TVTR	Voltage transformer, neutral conductor N

The 9-2LE guide defines the data set: **"PhsMeas1"**. This data set fixes which attributes from the data model are transferred as sampled values in the ASDU and then in the SV message (APDU).

PhsMeas1 Dataset member	CDC	FC
InnATCTR1.AmpSv	SAV	MX
InnBTCTR2.AmpSv	SAV	MX
InnCTCTR3.AmpSv	SAV	MX
InnNTCTR4.AmpSv	SAV	MX
InnATVTR1.VolSv	SAV	MX
InnBTVTR2.VolSv	SAV	MX
InnCTVTR3.VolSv	SAV	MX
InnNTVTR4.VolSv	SAV	MX

The dataset members: **"AmpSv"** and **"VolSv"** should have the following mandatory attributes:

Attribute	Type	Description
instMag.i	INT32	Sampled measured value (voltage or current)
q	Quality	Quality flags including the additional Derived flag.

The guide also defines scaling factors for current and voltage. The current and voltage values must be scaled with the respective factor and transmitted as whole numbers. Current factor: 1000 (LSB: 1 mA). Voltage factor: 100 (LSB: 10 mV).

SV control blocks

The 9-2LE document defines two SV control blocks: "**MSVCB01**" (80 samples per period) and "**MSVCB02**" (256 samples per period). At least one of these blocks must be implemented. With "**MSVCB01**" only one ASDU is transmitted in an SV message and with "**MSVCB02**" eight ASDUs. The sample project implements both control blocks. The control block instances can be found in the LLN0 node.

```
MSVCB01: FB_GseMsvCBImplClass := (MsvID:=(sValue:='xxxxMUnn01'), DatSet:=(sValue:='IEDMUnn/
LLN0.PhsMeas1'), ConfRev:=(nValue:=1), SmpRate:=(nValue:=80), OptFlds:=(RefreshTime:=TRUE), DstAddre
ss:=(sAddr:='01-0C-
CD-04-00-00', nPRIORITY:=4, nVID:=16#000, nAPPID:=16#4000), NoASDU:=(nValue:=1), bLinkResult:=THIS^.
AddMulticastSampledValueControlBlockToContainer(ipMulticastSampledValueControlBlock:=MSVCB01));
```

```
MSVCB02: FB_GseMsvCBImplClass := (MsvID:=(sValue:='xxxxMUnn02'), DatSet:=(sValue:='IEDMUnn/
LLN0.PhsMeas1'), ConfRev:=(nValue:=1), SmpRate:=(nValue:=256), OptFlds:=(RefreshTime:=TRUE), DstAddr
ess:=(sAddr:='01-0C-
CD-04-00-01', nPRIORITY:=4, nVID:=16#000, nAPPID:=16#4001), NoASDU:=(nValue:=8), bLinkResult:=THIS^.
AddMulticastSampledValueControlBlockToContainer(ipMulticastSampledValueControlBlock:=MSVCB02));
```

Theoretically, each SV message (APDU) can combine up to 8 ASDUs from different sources. In this case, each ASDU is identified by its unique identifier ("MsvID"). The guide recommends the use of an APPID ("nAPPID") with the value: 0x4000 and the use of multicast addresses ("DstAddress") from the value range: "01:0C:CD:04:xx:xx". The configuration version ("ConfRev") should always have the value: 1.

The SV Subscriber only receives, filters and analyzes messages that are assigned to the respective subscriber.

SV task configuration

The SV messages are processed by two separate PLC tasks in the sample project. Task: **"PlcTaskSV01"** processes the data of the control block: **"MSVCB01"** and Task: **"PlcTaskSV02"** processes the data of the control block: **"MSVCB02"**.

The control blocks send the SV messages at precisely defined time intervals *t*. The time interval *t.* depends on the frequency of the measuring signal (mains frequency) and the number of samples per period (SPP).

Calculation of the time interval for **"MSVCB01"**:
Mains frequency f: 50 Hz and SPP: 80

*t = 1/(f*SPP) = 1/(50*80) = 0.00025 s = 250 µs*

In this case, an SV message (APDU with an ASDU) is sent every 250 µs. This means a total of 4000 frames/second.

Calculation of the time interval for **"MSVCB02"**:
Mains frequency f: 50 Hz and SPP: 256

*t = 1/(f*SPP) = 1/(50*256) = 0.000078125 s = 78.125 µs*

In this case, however, an SV message is only sent every 625 µs because each SV message contains 8 ASDUs (the measured values in the ASDUs have the same sources). This means a total of 1600 frames/second.

In the sample, the TwinCAT base time for all PLC tasks is preconfigured to the value: 125 µs. This enables us to simulate the SV transmission of the sampled values of the three-phase voltages/currents in the 50 Hz grid. The **"Cycle ticks"** multiplier of the PLC task: **"PlcTaskSV01"** has the value: 2 and it is therefore called every 250 µs. The **"Cycle tics"** multiplier of the PLC task: **"PlcTaskSV02"** has the value: 5 and it is therefore called every 625 µs. This corresponds to the 9-2LE requirement for **"MSVCB01"** and **"MSVCB02"** in order to fulfill the SPP values 80 and 256 respectively (50 Hz mains frequency). Each control block periodically sends exactly one SV message in its own task cycle.

The PLC task: **"PlcTask"** with a cycle time of 10 ms is responsible for processing the client-server communication.

Special SV data structures and SV function blocks

In addition to the IED data model, the TwinCAT Telecontrol Configurator also generates the structured data types and function blocks required for SV implementation. The SV data structures are saved in the project tree: **"\DUTs\{IEDName}\Sampled Values"** and the SV function blocks are saved in the project tree:

"\POUs\{IEDName}\Sampled Values". The structured data types and function blocks are required to convert the sampled measured values, which are initially available in TwinCAT PLC data format (SV Publisher), into the data format that is to be transmitted in the SV message. In the opposite direction, the data received in the SV message is converted into the TwinCAT PLC data format (SV Subscriber). The IEC 61850-9-2 standard defines special encoding rules for the IEC 61850-7-2 basic types. In principle, only simple basic types are coded in the SV message. This means that even with the constructed data types (CDCs): **"VolSv"** and **"AmpSv"** in the data set: **"PhsMeas1"**, only the data of the attributes **"i"** and **"q"** are coded in the SV message. The generated SV data structures and SV function blocks are briefly described below. For the sake of simplicity, this is only done using the **"MSVCB01"** control block implementation as an example.

The generated SV data structures have the following name prefixes: **"ST_SVDATA_"**, **"ST_SVASDU_"**, **"ST_SVAPDU_"** and **"ST_SVBUFFER_"** (only in the SV Subscriber project).

The type encoding, byte alignment, and byte order of the member variables in the data structure **“ST_SVDATA_IED_MUnn_LLN0_PhsMeas1”** correspond exactly to the set-up of the data from the data set **“PhsMeas1”** and the format in which this data set is transmitted in the network. This means that the data in this structure have 1-byte alignment and network byte order (big-endian). The data types are already converted to SV data types according to the encoding rules from the IEC 61850-7-2 standard.

```
(* SV dataset values buffer type (IEC 61850 big endian network byte order) *)
{attribute 'pack_mode' := '1'}
TYPE ST_SVDATA_IED_MUnn_LLN0_PhsMeas1 :
STRUCT
    IED_MUnn_InnATCTR1_AmpSv_instMag_i: T_OCTET4;
    IED_MUnn_InnATCTR1_AmpSv_q: T_OCTET4;
    IED_MUnn_InnBTCTR2_AmpSv_instMag_i: T_OCTET4;
    IED_MUnn_InnBTCTR2_AmpSv_q: T_OCTET4;
    IED_MUnn_InnCTCTR3_AmpSv_instMag_i: T_OCTET4;
    IED_MUnn_InnCTCTR3_AmpSv_q: T_OCTET4;
    IED_MUnn_InnNTCTR4_AmpSv_instMag_i: T_OCTET4;
    IED_MUnn_InnNTCTR4_AmpSv_q: T_OCTET4;
    IED_MUnn_UnnATVTR1_VolSv_instMag_i: T_OCTET4;
    IED_MUnn_UnnATVTR1_VolSv_q: T_OCTET4;
    IED_MUnn_UnnBTVTR2_VolSv_instMag_i: T_OCTET4;
    IED_MUnn_UnnBTVTR2_VolSv_q: T_OCTET4;
    IED_MUnn_UnnCTVTR3_VolSv_instMag_i: T_OCTET4;
    IED_MUnn_UnnCTVTR3_VolSv_q: T_OCTET4;
    IED_MUnn_UnnNTVTR4_VolSv_instMag_i: T_OCTET4;
    IED_MUnn_UnnNTVTR4_VolSv_q: T_OCTET4;
END_STRUCT
END_TYPE
```

The member variables in the data structure **“ST_SVASDU_IED_MUnn_LLN0_MSVCB01”** have the encoding type, the byte alignment, and the byte order, which correspond to the type, alignment, and byte order of the data in TwinCAT PLC or the TwinCAT IEC 61850 data format. This means that the data in this structure have 8-byte alignment and host byte order (little-endian). In addition to some additional variables such as **“nSmpCnt”** (sample counter), **“eSmpSynch”** (synchronization type), and **“tRefrTm”** (refresh timestamp), the remaining member variables correspond to the set-up of the data in the data set: **“PhsMeas1”**.

```
(* SV ASDU data type (TwinCAT PLC little endian host byte order) *)
TYPE ST_SVASDU_IED_MUnn_LLN0_MSVCB01 :
STRUCT
    nSmpCnt: UINT; (* Sample counter *)
    eSmpSynch: E_GseSvSmpSynch; (* Synchronization type: 0..2 *)
    tRefrTm: T_UtcTime; (* Sample refresh time stamp (optional) *)

    IED_MUnn_InnATCTR1_AmpSv_instMag_i: DINT;
    IED_MUnn_InnATCTR1_AmpSv_q: ST_AcsiQuality;
    IED_MUnn_InnBTCTR2_AmpSv_instMag_i: DINT;
    IED_MUnn_InnBTCTR2_AmpSv_q: ST_AcsiQuality;
    IED_MUnn_InnCTCTR3_AmpSv_instMag_i: DINT;
    IED_MUnn_InnCTCTR3_AmpSv_q: ST_AcsiQuality;
    IED_MUnn_InnNTCTR4_AmpSv_instMag_i: DINT;
    IED_MUnn_InnNTCTR4_AmpSv_q: ST_AcsiQuality;
    IED_MUnn_UnnATVTR1_VolSv_instMag_i: DINT;
    IED_MUnn_UnnATVTR1_VolSv_q: ST_AcsiQuality;
    IED_MUnn_UnnBTVTR2_VolSv_instMag_i: DINT;
    IED_MUnn_UnnBTVTR2_VolSv_q: ST_AcsiQuality;
    IED_MUnn_UnnCTVTR3_VolSv_instMag_i: DINT;
    IED_MUnn_UnnCTVTR3_VolSv_q: ST_AcsiQuality;
    IED_MUnn_UnnNTVTR4_VolSv_instMag_i: DINT;
```

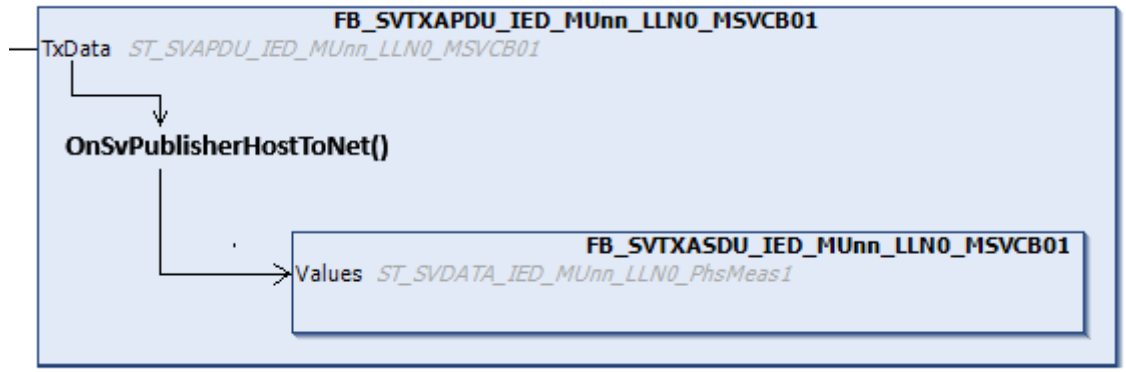
```
    IED_MUnn_UnnNTVTR4_VolSv_q: ST_AcsiQuality;
END_STRUCT
END_TYPE
```

The member variables in the data structure: **“ST_SVAPDU_IED_MUnn_LLN0_MSVCB01”** are instances of the **“ST_SVASDU_”** data structure in TwinCAT PLC data format (little-endian, host byte order).

```
(* SV APDU data type (TwinCAT PLC little endian host byte order) *)
TYPE ST_SVAPDU_IED_MUnn_LLN0_MSVCB01 :
STRUCT
    Asdu1: ST_SVASDU_IED_MUnn_LLN0_MSVCB01; (* SV ASDU *)
END_STRUCT
END_TYPE
```

Type encoding and data mapping

The function blocks with the name prefix: **"FB_SVTXASDU_"**, **"FB_SVTXAPDU_"**, **"FB_SVRXASDU_"** and **"FB_SVRXAPDU_"** are responsible for the type encoding and mapping of the data. Each time the **"Update()"** method is called in the SV Publisher (further information on this can be found further down in this documentation), the method: **"OnSvPublisherHostToNet()"** is called in the function block: **"FB_SVTXAPDU_IED_MUnn_LLN0_MSVCB01"**. In this method, the data is converted from TwinCAT PLC host format to SV network format before it is sent as an SV message.



To perform the conversion, the implementation code of the method: **"OnSvPublisherHostToNet()"** uses methods that are already implemented in the base class: **"FB_GseSvApduClass"** (see sample below).

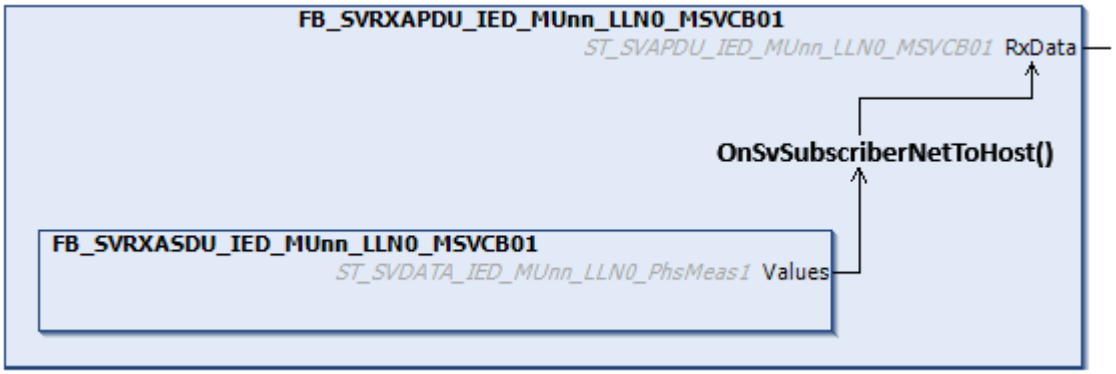
```
(* Converts SV publisher data to be sent from host (little-endian) to network (big-
endian) byte order *)
METHOD FINAL OnSvPublisherHostToNet
VAR_INPUT
    ipAdapter: I_GseAdapterClass; (* Network adapter interface pointer *)
    ipApdu: I_GseSvApduClass; (* APDU object interface pointer *)
END_VAR
```

```
Asdu1.nSmpCnt:= TxData.Asdu1.nSmpCnt; (* Set sample counter *)
Asdu1.eSmpSynch:= TxData.Asdu1.eSmpSynch; (* Set synchronization type: 0..2 *)
Asdu1.tRefrTm:= TxData.Asdu1.tRefrTm; (* Set refresh time stamp (optional) *)

(* Convert SV data to be sent from host byte order to network byte order *)
HostToNetInt32(in:=TxData.Asdu1.IED_MUnn_InnATCTR1_AmpSv_instMag_i, out:=Asdu1.Values.IED_MUnn_InnAT
CTR1_AmpSv_instMag_i);
HostToNetQuality(in:=TxData.Asdu1.IED_MUnn_InnATCTR1_AmpSv_q, out:=Asdu1.Values.IED_MUnn_InnATCTR1_A
mpSv_q);
HostToNetInt32(in:=TxData.Asdu1.IED_MUnn_InnBTCTR2_AmpSv_instMag_i, out:=Asdu1.Values.IED_MUnn_InnBT
CTR2_AmpSv_instMag_i);
HostToNetQuality(in:=TxData.Asdu1.IED_MUnn_InnBTCTR2_AmpSv_q, out:=Asdu1.Values.IED_MUnn_InnBTCTR2_A
mpSv_q);
HostToNetInt32(in:=TxData.Asdu1.IED_MUnn_InnCTCTR3_AmpSv_instMag_i, out:=Asdu1.Values.IED_MUnn_InnCT
CTR3_AmpSv_instMag_i);
HostToNetQuality(in:=TxData.Asdu1.IED_MUnn_InnCTCTR3_AmpSv_q, out:=Asdu1.Values.IED_MUnn_InnCTCTR3_A
mpSv_q);
HostToNetInt32(in:=TxData.Asdu1.IED_MUnn_InnNTCTR4_AmpSv_instMag_i, out:=Asdu1.Values.IED_MUnn_InnNT
CTR4_AmpSv_instMag_i);
HostToNetQuality(in:=TxData.Asdu1.IED_MUnn_InnNTCTR4_AmpSv_q, out:=Asdu1.Values.IED_MUnn_InnNTCTR4_A
mpSv_q);
HostToNetInt32(in:=TxData.Asdu1.IED_MUnn_UnnATVTR1_VolSv_instMag_i, out:=Asdu1.Values.IED_MUnn_UnnAT
VTR1_VolSv_instMag_i);
HostToNetQuality(in:=TxData.Asdu1.IED_MUnn_UnnATVTR1_VolSv_q, out:=Asdu1.Values.IED_MUnn_UnnATVTR1_V
```

```
olSv_q);
HostToNetInt32(in:=TxData.Asdu1.IED_MUnn_UnnBTVTR2_VolSv_instMag_i, out:=Asdu1.Values.IED_MUnn_UnnBT
VTR2_VolSv_instMag_i);
HostToNetQuality(in:=TxData.Asdu1.IED_MUnn_UnnBTVTR2_VolSv_q, out:=Asdu1.Values.IED_MUnn_UnnBTVTR2_V
olSv_q);
HostToNetInt32(in:=TxData.Asdu1.IED_MUnn_UnnCTVTR3_VolSv_instMag_i, out:=Asdu1.Values.IED_MUnn_UnnCT
VTR3_VolSv_instMag_i);
HostToNetQuality(in:=TxData.Asdu1.IED_MUnn_UnnCTVTR3_VolSv_q, out:=Asdu1.Values.IED_MUnn_UnnCTVTR3_V
olSv_q);
HostToNetInt32(in:=TxData.Asdu1.IED_MUnn_UnnNTVTR4_VolSv_instMag_i, out:=Asdu1.Values.IED_MUnn_UnnNT
VTR4_VolSv_instMag_i);
HostToNetQuality(in:=TxData.Asdu1.IED_MUnn_UnnNTVTR4_VolSv_q, out:=Asdu1.Values.IED_MUnn_UnnNTVTR4_V
olSv_q);
```

Each time the SV Subscriber receives a new SV message, the method: **"OnSvSubscriberNetToHost()"** is called in the function block: **"FB_SVRXAPDU_IED_MUnn_LLN0_MSVCB01"**. In this method, the received data is converted from the SV network format to the TwinCAT PLC host or TwinCAT IEC 61850 data format. After the conversion, the event method: **"OnSvUpdate()"** is called in the SV Subscriber (further information on this can be found further down in this documentation).



To perform the conversion, the implementation code of the method: **"OnSvSubscriberNetToHost()"** uses methods that are already implemented in the base class: **"FB_GseSvApduClass"** (see sample code below).

```
(* Converts received SV subscriber data from network (big-endian) to host (little-
endian) byte order *)
METHOD FINAL OnSvSubscriberNetToHost
VAR_INPUT
    ipAdapter: I_GseAdapterClass; (* Network adapter interface pointer *)
    ipApdu: I_GseSvApduClass; (* APDU object interface pointer *)
END_VAR
```

```
RxData.Asdu1.nSmpCnt:= Asdu1.nSmpCnt; (* Get sample counter *)
RxData.Asdu1.eSmpSynch:= Asdu1.eSmpSynch; (* Get synchronization type: 0..2 *)
RxData.Asdu1.tRefrTm:= Asdu1.tRefrTm; (* Get refresh time stamp (optional) *)

(* Convert received SV data from network byte order to host byte order *)
NetToHostInt32(in:=Asdu1.Values.IED_MUnn_InnATCTR1_AmpSv_instMag_i, out:=RxData.Asdu1.IED_MUnn_InnAT
CTR1_AmpSv_instMag_i);
NetToHostQuality(in:=Asdu1.Values.IED_MUnn_InnATCTR1_AmpSv_q, out:=RxData.Asdu1.IED_MUnn_InnATCTR1_A
mpSv_q);
NetToHostInt32(in:=Asdu1.Values.IED_MUnn_InnBTCTR2_AmpSv_instMag_i, out:=RxData.Asdu1.IED_MUnn_InnBT
CTR2_AmpSv_instMag_i);
NetToHostQuality(in:=Asdu1.Values.IED_MUnn_InnBTCTR2_AmpSv_q, out:=RxData.Asdu1.IED_MUnn_InnBTCTR2_A
mpSv_q);
NetToHostInt32(in:=Asdu1.Values.IED_MUnn_InnCTCTR3_AmpSv_instMag_i, out:=RxData.Asdu1.IED_MUnn_InnCT
CTR3_AmpSv_instMag_i);
NetToHostQuality(in:=Asdu1.Values.IED_MUnn_InnCTCTR3_AmpSv_q, out:=RxData.Asdu1.IED_MUnn_InnCTCTR3_A
mpSv_q);
NetToHostInt32(in:=Asdu1.Values.IED_MUnn_InnNTCTR4_AmpSv_instMag_i, out:=RxData.Asdu1.IED_MUnn_InnNT
CTR4_AmpSv_instMag_i);
NetToHostQuality(in:=Asdu1.Values.IED_MUnn_InnNTCTR4_AmpSv_q, out:=RxData.Asdu1.IED_MUnn_InnNTCTR4_A
mpSv_q);
NetToHostInt32(in:=Asdu1.Values.IED_MUnn_UnnATVTR1_VolSv_instMag_i, out:=RxData.Asdu1.IED_MUnn_UnnAT
VTR1_VolSv_instMag_i);
NetToHostQuality(in:=Asdu1.Values.IED_MUnn_UnnATVTR1_VolSv_q, out:=RxData.Asdu1.IED_MUnn_UnnATVTR1_V
olSv_q);
NetToHostInt32(in:=Asdu1.Values.IED_MUnn_UnnBTVTR2_VolSv_instMag_i, out:=RxData.Asdu1.IED_MUnn_UnnBT
VTR2_VolSv_instMag_i);
NetToHostQuality(in:=Asdu1.Values.IED_MUnn_UnnBTVTR2_VolSv_q, out:=RxData.Asdu1.IED_MUnn_UnnBTVTR2_V
olSv_q);
```

```
NetToHostInt32(in:=Asdu1.Values.IED_MUnn_UnnCTVTR3_VolSv_instMag_i, out:=RxData.Asdu1.IED_MUnn_UnnCT
VTR3_VolSv_instMag_i);
NetToHostQuality(in:=Asdu1.Values.IED_MUnn_UnnCTVTR3_VolSv_q, out:=RxData.Asdu1.IED_MUnn_UnnCTVTR3_V
olSv_q);
NetToHostInt32(in:=Asdu1.Values.IED_MUnn_UnnNTVTR4_VolSv_instMag_i, out:=RxData.Asdu1.IED_MUnn_UnnNT
VTR4_VolSv_instMag_i);
NetToHostQuality(in:=Asdu1.Values.IED_MUnn_UnnNTVTR4_VolSv_q, out:=RxData.Asdu1.IED_MUnn_UnnNTVTR4_V
olSv_q);
```

SV Publisher function block

In the global variable list: TcTelecontrol you will find an IED data model function block instance: **"fbIED"** and two SV Publisher function block instances. The program: **"P_IEC61850SV01"** is called by **"PlcTaskSV01"**. **"P_IEC61850SV01"** in turn calls the SV Publisher function block: **"fbIED_MUnn_MSVCB01SvPublisher"**, which is responsible for sending the SV messages. The second SV Publisher function block: **"fbIED_MUnn_MSVCB02SvPublisher"** is called by **"PlcTaskSV02"** and the program: **"P_IEC61850SV02"**.

```
VAR_GLOBAL
    …
    (* IED data model *)
    fbIED: FB_IED_IED;
    (* Sampled values publisher *)
    fbIED_MUnn_MSVCB01SvPublisher: FB_IED_MUnn_MSVCB01SvPublisher := (fbAdapter:=(ipIED:=fbIED, sett
ings:=(eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
    fbIED_MUnn_MSVCB02SvPublisher: FB_IED_MUnn_MSVCB02SvPublisher := (fbAdapter:=(ipIED:=fbIED, sett
ings:=(eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
    …
END_VAR
```

In the sample implementation, the SV publishing process is started automatically for both SV control blocks after the PLC is started. A rising edge at the **“bStop”** variable stops the publishing process. Simulated current and voltage measured values are generated by the **“FB_GseSvPhsMeasGeneratorClass”** function block for SV transmission. If the SV transmission was started via a rising edge at the **“bStart”** variable (this happens automatically after the PLC is started) and the **“bUpdate”** variable has the value TRUE, new measured values for the current and voltage of the three phases A, B, and C are generated in the method: **SvUpdate()** and assigned to **“fbApdu1.TxData”**. The current and voltage values of the neutral conductor (N) are calculated from the sum of the measured values of phases A, B and C. Alternatively, analog input signals for current and voltage from physical measurement terminals (I/O terminals) can also be used as a signal source for SV transmission. See the sample code implementation of the MSVCB01 publisher below.

```
(* Sampled values publisher *)
FUNCTION_BLOCK FB_IED_MUnn_MSVCB01SvPublisher IMPLEMENTS I_GseLinkStatusEventSink
VAR_INPUT
    fbAdapter: FB_GseAdapterClass := (ipLinkStatus:=THIS^); (* GSE network adapter *)
    fbApdu1: FB_SVTXAPDU_IED_MUnn_LLN0_MSVCB01; (* SV publisher APDU *)
END_VAR
VAR
    eLinkStatus: E_GseLinkStatus; (* Network adapter link status *)
    bSuccess: BOOL;
    ipError: I_ServiceErrorClass;
    bStart: BOOL := TRUE; (* Rising edge starts publishing of SV data *)
    bStop: BOOL; (* Rising edge stops publishing of SV data *)
    bUpdate: BOOL := TRUE; (* Enable/
disable cyclic update of published SV data. Default: TRUE (enabled) *)
    bUpdateOnce: BOOL; (* Rising edge updates published SV data only once *)

    bSynch: BOOL := TRUE; (* Rising edge at this input resets the sample counter to 0 *)
    nSynch: UDINT; (* Increments every time sampling is synchronized *)
    nSmpCnt: UINT; (* Sample counter *)
    eSmpSynch: E_GseSvSmpSynch := E_GseSvSmpSynch.Local; (* Synchronization type: 0..2 *)
    tRefrTm: T_UtcTime; (* Sample refresh time stamp (optional) *)
    qMeasure: ST_AcsiQuality := (Derived:=TRUE); (* SV quality bits *)
    bSimulation: BOOL := TRUE; (* Enable/
disable SV test signal function generator. Default: TRUE (enabled) *)

    fbGenerator1: FB_GseSvPhsMeasGeneratorClass := (
        nOversampleFactor:=1, (* Sets oversampling factor. Default: 1 (no oversampling) *)
        AmpA:=(fFrequency:=50.0, fAmplitude:=100, fPhi:=0),
        AmpB:=(fFrequency:=50.0, fAmplitude:=100, fPhi:=-120),
        AmpC:=(fFrequency:=50.0, fAmplitude:=100, fPhi:=-240),
        VolA:=(fFrequency:=50.0, fAmplitude:=230 * SQRT(2), fPhi:=0),
        VolB:=(fFrequency:=50.0, fAmplitude:=230 * SQRT(2), fPhi:=-120),
        VolC:=(fFrequency:=50.0, fAmplitude:=230 * SQRT(2), fPhi:=-240)); (* SV test signal function
 generator for PhsMeas1 dataset according to UCA 61850-9-2LE specification *)

    IED_MUnn_InnATCTR1_AmpSv_instMag_i AT %I*: DINT;
```

```
    IED_MUnn_InnBTCTR2_AmpSv_instMag_i AT %I*: DINT;
    IED_MUnn_InnCTCTR3_AmpSv_instMag_i AT %I*: DINT;
    IED_MUnn_InnNTCTR4_AmpSv_instMag_i AT %I*: DINT;
    IED_MUnn_UnnATVTR1_VolSv_instMag_i AT %I*: DINT;
    IED_MUnn_UnnBTVTR2_VolSv_instMag_i AT %I*: DINT;
    IED_MUnn_UnnCTVTR3_VolSv_instMag_i AT %I*: DINT;
    IED_MUnn_UnnNTVTR4_VolSv_instMag_i AT %I*: DINT;
END_VAR
```

```
bSuccess:= fbAdapter.Execute(ipError=>ipError); (* Execute network adapter *)

IF bStart THEN (* Start publishing of SV data *)
    bStart:= FALSE;
    fbGenerator1.Config(); (* Configure SV test signal function generator *)
    bSuccess:= fbIED.IEDMUnn.LLN0.MSVCB01.Publisher.Start(ipAdapter:=fbAdapter, ipApdu:=fbApdu1, ipE
rror=>ipError);
ELSIF bStop THEN (* Stop publishing of SV data *)
    bStop:= FALSE;
    bSuccess:= fbIED.IEDMUnn.LLN0.MSVCB01.Publisher.Stop(ipError=>ipError);
ELSIF bUpdate OR_ELSE bUpdateOnce THEN (* Update published SV data *)
    bUpdateOnce:= FALSE;
    SvUpdate(); (* Update SV data (test signal | I/O input terminal) *)
    bSuccess:= fbIED.IEDMUnn.LLN0.MSVCB01.Publisher.Update(ipError=>ipError);
ELSE (* Execute publisher SvCB's *)
    bSuccess:= fbIED.IEDMUnn.LLN0.MSVCB01.Publisher.Execute(ipError=>ipError);
END_IF
```

SV-Publisher SvUpdate()-Methode

```
(* Update SV data. The data will be sent the next time when the publisher's Update() method is calle
d *)
METHOD PRIVATE SvUpdate
VAR
END_VAR
```

```
fbApdu1.bSimulation:= bSimulation; (* Example => Sets SV ethernet header simulation flag *)
qMeasure.Test:= bSimulation; (* Example => Sets "Quality.Test" bit to TRUE if simulation enabled *)
qMeasure.Validity1:= NOT qMeasure.Validity1; (* Example => Toggle some "Quality.Validity" bits *)

IF bSynch THEN
    bSynch:= FALSE;
    nSmpCnt:= 0; (* Example => Set sample counter to 0 *)
    fbGenerator1.Reset(); (* Reset SV test signal function generator *)
    nSynch:= nSynch + 1;
ELSE
    nSmpCnt:= nSmpCnt + 1; (* Increment sample counter *)
END_IF

IF bSimulation THEN
    fbGenerator1.Call(); (* Call SV test signal function generator for new data *)
END_IF

fbApdu1.TxData.Asdu1.nSmpCnt:= nSmpCnt; (* Set sample counter *)
fbApdu1.TxData.Asdu1.eSmpSynch:= eSmpSynch; (* Set synchronization type: 0..2 *)
(* Hint: The time stamp will only be sent if "RefreshTime" parameter in the "OptFlds" of the SV publ
isher control block is set to TRUE *)
fbApdu1.TxData.Asdu1.tRefrTm:= tRefrTm; (* Set sample refresh time stamp (optional) *)

{warning 'Supplement the following code to acquire values from physical I/
O input terminal or simulation. Note: Data model must be IEC 61850 9-2LE compliant.'}
fbApdu1.TxData.Asdu1.IED_MUnn_InnATCTR1_AmpSv_instMag_i:= SEL(bSimulation, IED_MUnn_InnATCTR1_AmpSv_
instMag_i, TO_DINT(fbGenerator1.AmpA.fSignal));
fbApdu1.TxData.Asdu1.IED_MUnn_InnATCTR1_AmpSv_q:= qMeasure;
fbApdu1.TxData.Asdu1.IED_MUnn_InnBTCTR2_AmpSv_instMag_i:= SEL(bSimulation, IED_MUnn_InnBTCTR2_AmpSv_
instMag_i, TO_DINT(fbGenerator1.AmpB.fSignal));
fbApdu1.TxData.Asdu1.IED_MUnn_InnBTCTR2_AmpSv_q:= qMeasure;
fbApdu1.TxData.Asdu1.IED_MUnn_InnCTCTR3_AmpSv_instMag_i:= SEL(bSimulation, IED_MUnn_InnCTCTR3_AmpSv_
instMag_i, TO_DINT(fbGenerator1.AmpC.fSignal));
fbApdu1.TxData.Asdu1.IED_MUnn_InnCTCTR3_AmpSv_q:= qMeasure;
fbApdu1.TxData.Asdu1.IED_MUnn_InnNTCTR4_AmpSv_instMag_i:= SEL(bSimulation, IED_MUnn_InnNTCTR4_AmpSv_
instMag_i, TO_DINT(fbGenerator1.AmpA.fSignal + fbGenerator1.AmpB.fSignal + fbGenerator1.AmpC.fSignal
));
fbApdu1.TxData.Asdu1.IED_MUnn_InnNTCTR4_AmpSv_q:= qMeasure;
fbApdu1.TxData.Asdu1.IED_MUnn_UnnATVTR1_VolSv_instMag_i:= SEL(bSimulation, IED_MUnn_UnnATVTR1_VolSv_
instMag_i, TO_DINT(fbGenerator1.VolA.fSignal));
fbApdu1.TxData.Asdu1.IED_MUnn_UnnATVTR1_VolSv_q:= qMeasure;
fbApdu1.TxData.Asdu1.IED_MUnn_UnnBTVTR2_VolSv_instMag_i:= SEL(bSimulation, IED_MUnn_UnnBTVTR2_VolSv_
instMag_i, TO_DINT(fbGenerator1.VolB.fSignal));
fbApdu1.TxData.Asdu1.IED_MUnn_UnnBTVTR2_VolSv_q:= qMeasure;
fbApdu1.TxData.Asdu1.IED_MUnn_UnnCTVTR3_VolSv_instMag_i:= SEL(bSimulation, IED_MUnn_UnnCTVTR3_VolSv_
instMag_i, TO_DINT(fbGenerator1.VolC.fSignal));
```

```
fbApdu1.TxData.Asdu1.IED_MUnn_UnnCTVTR3_VolSv_q:= qMeasure;
fbApdu1.TxData.Asdu1.IED_MUnn_UnnNTVTR4_VolSv_instMag_i:= SEL(bSimulation, IED_MUnn_UnnNTVTR4_VolSv_
instMag_i, TO_DINT(fbGenerator1.VolA.fSignal + fbGenerator1.VolB.fSignal + fbGenerator1.VolC.fSignal
));
fbApdu1.TxData.Asdu1.IED_MUnn_UnnNTVTR4_VolSv_q:= qMeasure;
```

SV Subscriber function block

In the global variable list: TcTelecontrol you will find an IED data model function block instance: **"fbIED"** and two SV Subscriber function block instances. The program: **"P_IEC61850SV01"** is called by **"PlcTaskSV01"**. **"P_IEC61850SV01"** in turn calls the SV Subscriber function block: **"fbIED_MUnn_MSVCB01SvSubscriber"**, which is responsible for receiving the SV messages. The second SV Subscriber function block: **"fbIED_MUnn_MSVCB02SvSubscriber"** is called by **"PlcTaskSV02"** and the program: **"P_IEC61850SV02"**.

```
VAR_GLOBAL
    …
    (* IED data model *)
    fbIED: FB_IED_IED;
    (* Sampled values subscriber *)
    fbIED_MUnn_MSVCB01SvSubscriber: FB_IED_MUnn_MSVCB01SvSubscriber := (fbAdapter:=(ipIED:=fbIED, se
ttings:=(eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
    fbIED_MUnn_MSVCB02SvSubscriber: FB_IED_MUnn_MSVCB02SvSubscriber := (fbAdapter:=(ipIED:=fbIED, se
ttings:=(eDispatchMode:=E_GseDispatchMode.NonPromiscuous)));
    …
END_VAR
```

The receipt of SV messages is activated by a rising edge on the **“bSubscribe”** variable. A rising edge on the **“bUnsubscribe”** variable stops the SV message reception. In the sample project, both SV subscriber instances are activated when the PLC program starts.

Each time the SV subscriber receives a new SV message, the event method **“OnSvUpdate()”** is called in the SV Subscriber function block. In our sample implementation, this method copies the received SV measured values from **"fbApdu1.RxData"** to a larger data buffer for later evaluation. What happens with the received data is left to the application. However, the data must be saved in some way because it is overwritten with newly received SV data.

```
(* Sampled values subscriber *)
FUNCTION_BLOCK FB_IED_MUnn_MSVCB01SvSubscriber IMPLEMENTS I_GseLinkStatusEventSink, I_GseSvUpdateEve
ntSink
VAR_INPUT
    fbAdapter: FB_GseAdapterClass := (ipLinkStatus:=THIS^); (* GSE network adapter *)
END_VAR
VAR_OUTPUT
    fbApdu1: FB_SVRXAPDU_IED_MUnn_LLN0_MSVCB01 := (ipUpdate:=THIS^); (* SV subscriber APDU *)
    RxBuffer1: ST_SVBUFFER_IED_MUnn_LLN0_MSVCB01; (* SV data buffer *)
END_VAR
VAR
    eLinkStatus: E_GseLinkStatus; (* Network adapter link status *)
    bSuccess: BOOL;
    ipError: I_ServiceErrorClass;
    bSubscribe: BOOL := TRUE; (* Rising edge enables subscriber SvCB's *)
    bUnsubscribe: BOOL; (* Rising edge disables subscriber SvCB's *)

    bSynch: BOOL; (* If TRUE => sample counter is 0 *)
    nSynch: UDINT; (* Increments every time sample counter is 0 *)
    nSmpCnt: UINT; (* Sample counter *)
    eSmpSynch: E_GseSvSmpSynch := E_GseSvSmpSynch.None; (* Synchronization type: 0..2 *)
    tRefrTm: T_UtcTime; (* Sample refresh time stamp (optional) *)
    bSimulation: BOOL; (* SV ethernet header simulation flag *)
END_VAR
```

```
bSuccess:= fbAdapter.Execute(ipError=>ipError); (* Execute network adapter *)

IF bSubscribe THEN (* Enable subscriber SvCB's *)
    bSubscribe:= FALSE;
    bSuccess:= fbIED.IEDMUnn.LLN0.MSVCB01.Subscriber.Enable(ipAdapter:=fbAdapter, ipApdu:=fbApdu1, i
pError=>ipError);
ELSIF bUnsubscribe THEN (* Disable subscriber SvCB's *)
    bUnsubscribe:= FALSE;
    bSuccess:= fbIED.IEDMUnn.LLN0.MSVCB01.Subscriber.Disable(ipError=>ipError);
ELSE (* Execute subscriber SvCB's *)
    bSuccess:= fbIED.IEDMUnn.LLN0.MSVCB01.Subscriber.Execute(ipError=>ipError);
END_IF
```

SV Subscriber OnSvUpdate() event method

```
(* This method receives SV APDU update event notification *)
METHOD FINAL OnSvUpdate
VAR_INPUT
    ipAdapter: I_GseAdapterClass; (* Network adapter interface pointer *)
    ipApdu: I_GseSvApduClass; (* APDU object interface pointer *)
END_VAR
```

```
(* Hint: The time stamp will only be sent if "RefreshTime" parameter in the "OptFlds" of the SV publ
isher control block is set to TRUE *)

(* Find the updated APDU object and process the received data *)
IF fbApdu1.Equals(ipOther:=ipApdu) THEN

    bSimulation:= fbApdu1.bSimulation; (* Get SV ethernet header simulation flag *)
    nSmpCnt:= fbApdu1.RxData.Asdu1.nSmpCnt; (* Get sample counter *)
    eSmpSynch:= fbApdu1.RxData.Asdu1.eSmpSynch; (* Get synchronization type: 0..2 *)
    tRefrTm:= fbApdu1.RxData.Asdu1.tRefrTm; (* Get sample refresh time stamp (optional) *)
    IF nSmpCnt = 0 THEN
        bSynch:= TRUE;
        nSynch:= nSynch + 1;
    END_IF

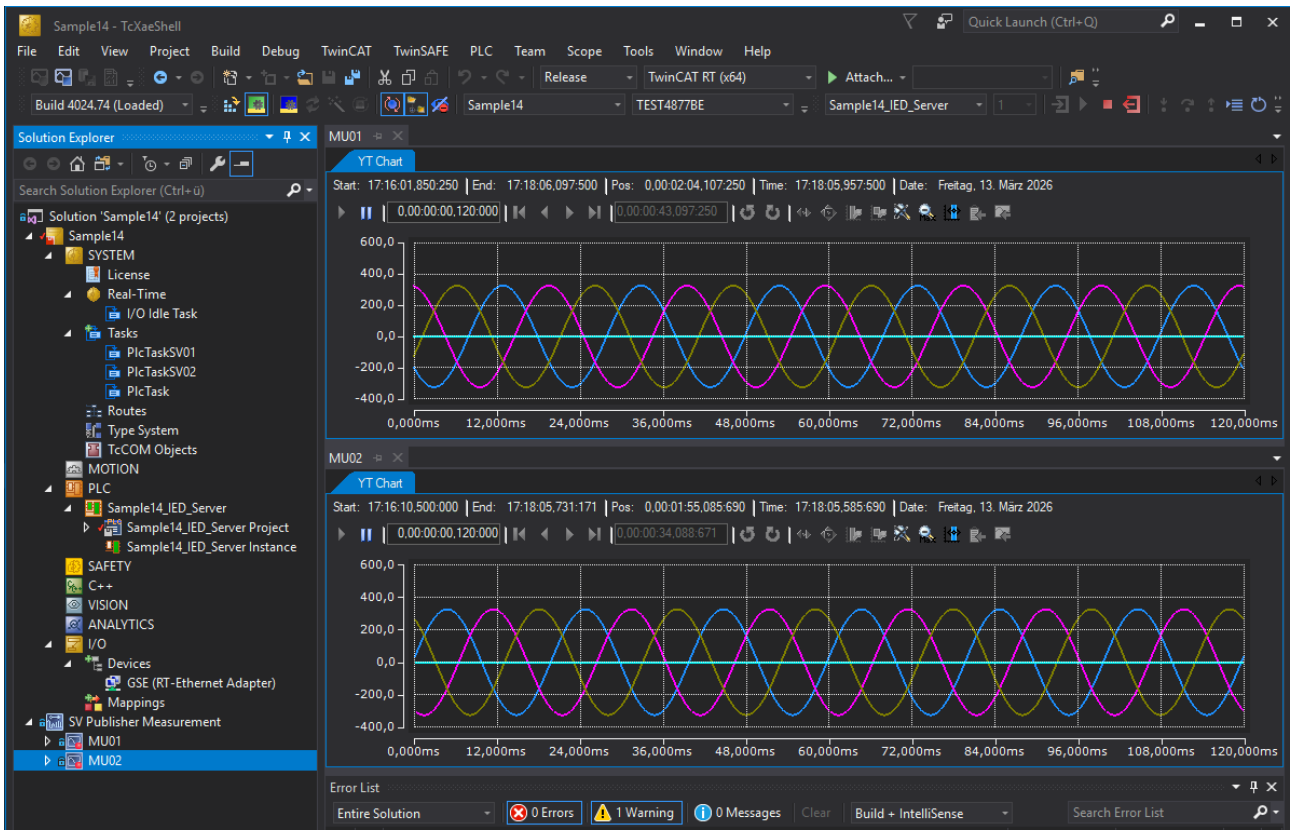
    RxBuffer1.IED_MUnn_InnATCTR1_AmpSv_instMag_i[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_In
nATCTR1_AmpSv_instMag_i;
    RxBuffer1.IED_MUnn_InnATCTR1_AmpSv_q[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_InnATCTR1_
AmpSv_q;
    RxBuffer1.IED_MUnn_InnBTCTR2_AmpSv_instMag_i[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_In
nBTCTR2_AmpSv_instMag_i;
    RxBuffer1.IED_MUnn_InnBTCTR2_AmpSv_q[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_InnBTCTR2_
AmpSv_q;
    RxBuffer1.IED_MUnn_InnCTCTR3_AmpSv_instMag_i[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_In
nCTCTR3_AmpSv_instMag_i;
    RxBuffer1.IED_MUnn_InnCTCTR3_AmpSv_q[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_InnCTCTR3_
AmpSv_q;
    RxBuffer1.IED_MUnn_InnNTCTR4_AmpSv_instMag_i[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_In
nNTCTR4_AmpSv_instMag_i;
    RxBuffer1.IED_MUnn_InnNTCTR4_AmpSv_q[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_InnNTCTR4_
AmpSv_q;
    RxBuffer1.IED_MUnn_UnnATVTR1_VolSv_instMag_i[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_Un
nATVTR1_VolSv_instMag_i;
    RxBuffer1.IED_MUnn_UnnATVTR1_VolSv_q[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_UnnATVTR1_
VolSv_q;
    RxBuffer1.IED_MUnn_UnnBTVTR2_VolSv_instMag_i[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_Un
nBTVTR2_VolSv_instMag_i;
    RxBuffer1.IED_MUnn_UnnBTVTR2_VolSv_q[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_UnnBTVTR2_
VolSv_q;
    RxBuffer1.IED_MUnn_UnnCTVTR3_VolSv_instMag_i[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_Un
nCTVTR3_VolSv_instMag_i;
    RxBuffer1.IED_MUnn_UnnCTVTR3_VolSv_q[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_UnnCTVTR3_
VolSv_q;
    RxBuffer1.IED_MUnn_UnnNTVTR4_VolSv_instMag_i[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_Un
nNTVTR4_VolSv_instMag_i;
    RxBuffer1.IED_MUnn_UnnNTVTR4_VolSv_q[RxBuffer1.index]:= fbApdu1.RxData.Asdu1.IED_MUnn_UnnNTVTR4_
VolSv_q;

    RxBuffer1.index:= SEL(RxBuffer1.index >= cArrayBufferSize_IED_MUnn_LLN0_MSVCB01-1, RxBuffer1.ind
ex + 1, 0);
END_IF
```

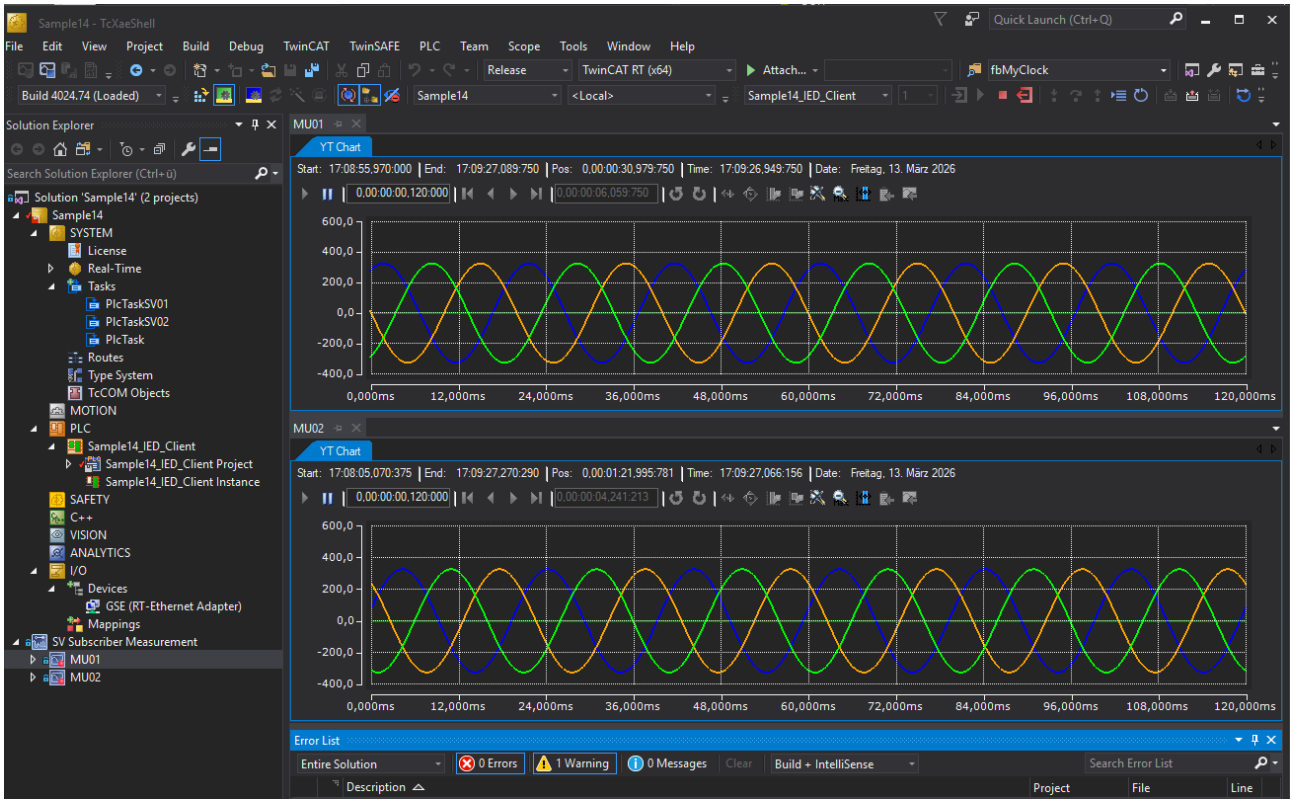
Test

Both sample projects: The SV Publisher and the SV Subscriber also contain a TwinCAT Measurement Scope project. These projects are not automatically created by TwinCAT Telecontrol Configurator. They were manually added to the TwinCAT SV Publisher/Subscriber PLC project. As soon as you have started both projects and the network adapter has been configured correctly beforehand, the data sent and received can be recorded in the measurement scope project. However, you may need to adapt the TwinCAT network addresses (NetIDs) to your actual configuration in the Measurement project beforehand. Right-click on **"SV Subscriber Measurement/MU01"** in the project tree and select **"Change Target System"** from the context menu. Repeat the same for **"SV Subscriber Measurement/MU02"**. Proceed similarly in the TwinCAT SV Publisher project.

If the recording in the sample project on the SV Publisher page is active, you should see an image similar to the one below (zoom in until you see sine waves).



If the recording in the sample project on the SV Subscriber page is active, you should see an image similar to the one below (zoom in until you see sine waves).



8 Appendix

8.1 Binary-Time LWORD format specification

The TwinCAT Binary-Time components are coded in LWORD in the following way:

Bit	Compo-nent	Type	Value range	Description
0..31	timeOfDay	TOD	0..16#05265BFF	Number of milliseconds since midnight of the current day. "16#05265BFF"=23:59:59 and 999 ms=86399999 ms.
32..47	day	WORD	0..16#FFFF	Number of days since January 1, 1984.
48..63	reserved	WORD	0	These bits are not used. Default:

Samples

The table below contains some LWORD numerical values and the values after conversion to the formatted Binary-Time string and the structured type T_BinaryTime [▶ 556].

LWORD	STRING(26)	T_BinaryTime
16#000000000000	'BT#1984-01-01-00:00:00'	STRUCT(timeOfDay:=TOD#00:00:00, day:=0)
16#000000000001	'BT#1984-01-01-00:00:00.001'	STRUCT(timeOfDay:=TOD#00:00:00.001, day:=0)
16#2F650270E660	'BT#2017-03-21-11:22:33.440‘	STRUCT(timeOfDay:=TOD#11:22:33.440, day:=16#2F65)
16#30E805265BFF	'BT#2018-04-12-23:59:59.999'	STRUCT(timeOfDay:=TOD#23:59:59.999, day:=16#30E8)

8.2 Binary-Time string format specification

A TwinCAT Binary-Time string has a max. length of 26 characters and the following structure: ‘BT#[YYYY-MM-DD-]hh:mm:ss[.n[n[n]]]‘.

Using the conversion function String_To_BinaryTime6 [▶ 339], the Binary-Time information, which is formatted as a string, can easily be converted into the components of the structured type T_BinaryTime [▶ 556]. The date is converted into the "day" component (2 bytes) and the time into the "timeOfDay" component (4 bytes).

The date in the Binary-Time string is optional and need not be specified. If the date was not specified, then the string is a 4-byte Binary-Time format (only the "timeOfDay" component). In this case the value 0 is implicitly assigned during the conversion of the "day" component (this corresponds to the date 1984-01-01). A string with date is a 6-byte Binary-Time format ("timeOfDay" and "day" component).

Variable	Representation	Value range	Description
Prefix	BT#	‘BT#‘	Constant prefix.
Year	YYYY	‘1984‘..‘2106‘	Year (4 decimal places, optional).
Month	MM	‘01‘..‘12‘	Month (2 decimal places, optional).
Day	DD	‘01‘..‘31‘	Day (2 decimal places, optional).
Hour	hh	‘00‘..‘23‘	Hour (2 decimal places).
Minute	mm	‘00‘..‘59‘	Minute (2 decimal places).
Second	ss	‘00‘..‘59‘	Second (2 decimal places).
Millisecond	n[n[n]]]	‘‘,‘0‘..‘999‘	Millisecond (0 to 3 decimal places, optional).

Samples

The table below contains some strings and the values of the components after conversion to the structured type T_BinaryTime [▶ 556].

STRING(26)	T_BinaryTime
‘BT#1984-01-01-00:00:00‘	STRUCT(timeOfDay:=TOD#00:00:00, day:=0)
‘BT#2018-03-04-11:12:55.6‘	STRUCT(timeOfDay:=TOD#11:12:55.600, day:=16#30C1)
‘BT#2018-03-04-11:12:55.678‘	STRUCT(timeOfDay:=TOD#11:12:55.678, day:=16#30C1)
‘BT#01:02:03‘	STRUCT(timeOfDay:=TOD#01:02:03, day:=0)
‘BT#01:02:03.5‘	STRUCT(timeOfDay:=TOD#01:02:03.500, day:=0)
‘BT#01:02:03.567‘	STRUCT(timeOfDay:=TOD#01:02:03.567, day:=0)

8.3 UTC-Time LWORD format specification

The TwinCAT UTC-Time components are coded in LWORD in the following way:

Bit	Component	Type	Value range	Description
0..31	secondSinceEpoch	DT	0..16#FFFFFFFF	Number of whole seconds since January 1, 1970.
32..55	fractionOfSecond	T_UINT24 [▶ 559]	0..16#FFFFFF	Seconds fraction encoded as 24-bit number.
56	Quality.LeapSeconds Known	BIT	0..1	Quality of the leap second (0:=unknown, 1:=known).
57	Quality.ClockFailure	BIT	0..1	Quality of the timer (0:=good, 1:=erroneous).
58	Quality.ClockNotSync hronized	BIT	0..1	Quality of the time synchronization (0:=synchronized, 1:=not synchronized).
59	Quality.Accuracy0	BIT	0..1	Accuracy of the time information (precision of the fraction of a second). The accuracy bits determine the number of significant bits of the T_UtcTime [▶ 560].fractionOfSecond component.
60	Quality.Accuracy1	BIT	0..1	
61	Quality.Accuracy2	BIT	0..1	
62	Quality.Accuracy3	BIT	0..1	
63	Quality.Accuracy4	BIT	0..1	

The UTC-Time.fractionOfSecond component is coded in the 24 bits of the type T_UINT24 in the following way:

Bit	Component	Value
0	fractionOfSecond[0].0	0.5 s
1	fractionOfSecond[0].1	0.25 s
2	fractionOfSecond[0].2	0.125 s
3	fractionOfSecond[0].3	0.0625 s
4	fractionOfSecond[0].4	0.03125 s
5	fractionOfSecond[0].5	0.015625 s
6	fractionOfSecond[0].6	0.0078125 s
7	fractionOfSecond[0].7	0.00390625 s
8	fractionOfSecond[1].0	0.001953125 s
9..15	fractionOfSecond[1].1..7	etc.
16..23	fractionOfSecond[2].0..7	-/-

Samples

The table below contains some fraction-of-a-second values and the values of the fractionOfSecond array elements after the conversion.

Fraction of a second	T_UINT24	Description
0 s	[0,0,0]	All bits have the value: 0.
0.75 s	[3,0,0]	Bits 0 and 1 are set. Value calculation: 0.5 s + 0.25 s = 0.75 s.
0.875 s	[7,0,0]	Bits 0, 1 and 2 are set. Value calculation: 0.5 s + 0.25 s + 0.125 s = 0.875 s.
0.5625 s	[9,0,0]	Bits 0 and 3 are set. Value calculation: 0.5 s + 0.0625 s = 0.5625 s.

Coding of the precision in the UTC-Time.Quality.Accuracy0..4 bits.

Precision	Accuracy0	Accuracy1	Accuracy2	Accuracy3	Accuracy4	E_UtcTimeAccuracy
0-bit	0	0	0	0	0	Zero
1-bit	0	0	0	0	1	_01
2-bit	0	0	0	1	0	_02
3-bit	0	0	0	1	1	_03
4-bit	0	0	1	0	0	_04
5-bit	0	0	1	0	1	_05
6-bit	0	0	1	1	0	_06
7-bit	0	0	1	1	1	_07
8-bit	0	1	0	0	0	_08
9-bit	0	1	0	0	1	_09
10-bit	0	1	0	1	0	_10
11-bit	0	1	0	1	1	_11
12-bit	0	1	1	0	0	_12
13-bit	0	1	1	0	1	_13
14-bit	0	1	1	1	0	_14
15-bit	0	1	1	1	1	_15
16-bit	1	0	0	0	0	_16
17-bit	1	0	0	0	1	_17
18-bit	1	0	0	1	0	_18
19-bit	1	0	0	1	1	_19
20-bit	1	0	1	0	0	_20
21-bit	1	0	1	0	1	_21
22-bit	1	0	1	1	0	_22
23-bit	1	0	1	1	1	_23
24-bit	1	1	0	0	0	_24
Impermissible	1	1	n	n	n	_25_invalid.._30_invalid
Not specified	1	1	1	1	1	Unspecified

Samples

The table below contains some LWORD values and the values after conversion to a formatted Binary-Time string and the structured type: T_UtcTime [▶ 560].

LWORD	STRING(39)	T_UtcTime	Description
16#0000000000000000	'UT#1970-01-01-00:00:00.000000000|000|0'	STRUCT(0)	All components have the value 0. Date: January 1, 1970. Time: 00:00:00. Fraction of a second: 0 ms. Precision: 0 bit.
16#C4000000386D4380	'UT#2000-01-01-00:00:00.000000000|001|3'	STRUCT(secondSinceEpoch:=DT#2000-01-01-00:00:00, fractionOfSecond:=[0,0,0], quality:=(ClockNotSynchronized:=1, Accuracy3:=1, Accuracy4:=1))	Date: January 1, 2000. Time: 00:00:00. Fraction of a second: 0 ms. Timer is not synchronized. Precision: 3-bit.
16#A4000001386D4380	'UT#2000-01-01-00:00:00.500000000|001|5'	STRUCT(secondSinceEpoch:=DT#2000-01-01-00:00:00, fractionOfSecond:=[1,0,0], quality:=(ClockNotSynchronized:=1, Accuracy2:=1, Accuracy4:=1))	Date: January 1, 2000. Time: 00:00:00. Fraction of a second: 500 ms. Timer is not synchronized. Precision: 5-bit.
16#810000000FFFFFFF	'UT#2106-02-07-06:28:15.000000000|100|1'	STRUCT(secondSinceEpoch:=DT#2106-02-07-06:28:15, fractionOfSecond:=[0,0,0], quality:=(LeapSecondsKnown:=1, Accuracy4:=1))	Date: February 7, 2106. Time: 06:28:15. Fraction of a second: 0 ms. Leap seconds are unknown. Precision: 1-bit.
16#C20000007FFFFFFF	'UT#2106-02-07-06:28:15.875000000|010|3'	STRUCT(secondSinceEpoch:=DT#2106-02-07-06:28:15, fractionOfSecond:=[7,0,0], quality:=(ClockFailure:=1, Accuracy3:=1, Accuracy4:=1))	Date: February 7, 2106. Time: 06:28:15. Fraction of a second: 875 ms. Timer signals an error. Precision: 3-bit.
16#1C000007FFFFFFFF	'UT#2106-02-07-06:28:15.875000000|001|24'	STRUCT(secondSinceEpoch:=DT#2106-02-07-06:28:15, fractionOfSecond:=[7,0,0], quality:=(ClockNotSynchronized:=1, Accuracy0:=1, Accuracy1:=1))	Date: February 7, 2106. Time: 06:28:15. Fraction of a second: 875 ms. Timer is not synchronized. Precision: 24-bit.

8.4 UTC-Time-Quality BYTE format specification

The TwinCAT UTC-Time-Quality components are coded in BYTE in the following way:

Bit	Component	Type	Value range	Description
0	LeapSecondsKnown	BIT	0..1	Quality of the leap second (0:=unknown, 1:=known).
1	ClockFailure	BIT	0..1	Quality of the timer (0:=good, 1:=erroneous).
2	ClockNotSynchronized	BIT	0..1	Quality of the time synchronization (0:=synchronized, 1:=not synchronized).
3	Accuracy0	BIT	0..1 (MSB)	Accuracy of the time information (precision of the fraction of a second). Max. number of significant bits in the T_UtcTime [▶ 560].fractionOfSecond component. In the precision, Accuracy0 is the most significant bit and Accuracy4 the least significant bit.
4	Accuracy1	BIT	0..1	
5	Accuracy2	BIT	0..1	
6	Accuracy3	BIT	0..1	
7	Accuracy4	BIT	0..1 (LSB)	

Coding of the precision in the UTC-Time.Quality.Accuracy0..4 bits.

Precision	Accuracy0	Accuracy1	Accuracy2	Accuracy3	Accuracy4	E_UtcTimeAccuracy
0-bit	0	0	0	0	0	Zero
1-bit	0	0	0	0	1	_01
2-bit	0	0	0	1	0	_02
3-bit	0	0	0	1	1	_03
4-bit	0	0	1	0	0	_04
5-bit	0	0	1	0	1	_05
6-bit	0	0	1	1	0	_06
7-bit	0	0	1	1	1	_07
8-bit	0	1	0	0	0	_08
9-bit	0	1	0	0	1	_09
10-bit	0	1	0	1	0	_10
11-bit	0	1	0	1	1	_11
12-bit	0	1	1	0	0	_12
13-bit	0	1	1	0	1	_13
14-bit	0	1	1	1	0	_14
15-bit	0	1	1	1	1	_15
16-bit	1	0	0	0	0	_16
17-bit	1	0	0	0	1	_17
18-bit	1	0	0	1	0	_18
19-bit	1	0	0	1	1	_19
20-bit	1	0	1	0	0	_20
21-bit	1	0	1	0	1	_21
22-bit	1	0	1	1	0	_22
23-bit	1	0	1	1	1	_23
24-bit	1	1	0	0	0	_24
Impermissible	1	1	n	n	n	_25_invalid.._30_invalid
Not specified	1	1	1	1	1	Unspecified

Samples

The table below contains some BYTE numerical values and the values after conversion to the formatted UTC Time-Quality string and the structured type: T_UtcTimeQuality [▶ 560].

BYTE	STRING(9)	T_UtcTimeQuality	Description
2#00000000	'UQ#000|0'	STRUCT(0)	All components have the value: 0. Precision: 0-bit.
2#11000000	'UQ#000|3'	STRUCT(Accuracy3:=1, Accuracy4:=1)	Precision: 3-bit. All other components have the value 0.
2#10100100	'UQ#001|5'	STRUCT(ClockNotSynchronized:=1, Accuracy2:=1, Accuracy4:=1)	Timer is not synchronized. Precision: 5-bit. All other components have the value 0.
2#11111010	'UQ#010|31'	STRUCT(ClockFailure:=1, Accuracy0:=1, Accuracy1:=1, Accuracy2:=1, Accuracy3:=1, Accuracy4:=1)	Timer signals an error. Precision: Not specified. All other components have the value 0.

8.5 UTC-Time-Quality string format specification

A TwinCAT UTC-Time-Quality string has the following structure: 'UQ#LFC|A[A]'.

Variable	Representation	Value range	Description
Prefix	UQ#	‘UQ#‘	Constant prefix.
LeapSecondsKnown	L	‘0‘..‘1‘	Quality of the leap second ('0':=unknown, '1':=known).
ClockFailure	F	‘0‘..‘1‘	Quality of the timer ('0':=good, '1':=erroneous).
ClockNotSynchronized	C	‘0‘..‘1‘	Quality of the time synchronization ('0':=synchronized, '1':=not synchronized).
Accuracy	A[A]	‘0‘..‘31‘	Accuracy of the time information (precision of the fraction of a second). Max. number of significant bits in the T_UtcTime [▶ 560].fractionOfSecond component (1 or 2 decimal places).

Samples

The table below contains some strings and the values of the components after conversion to the structured type T_UtcTimeQuality [▶ 560].

STRING(9)	T_UtcTimeQuality	Description
'UQ#000|0'	STRUCT(0)	All components have the value: 0.
'UQ#001|3'	STRUCT(ClockNotSynchronized:=1, Accuracy3:=1, Accuracy4:=1)	Timer not synchronized. Precision: 3-bit. All other components have the value: 0.
'UQ#000|24'	STRUCT(Accuracy0:=1, Accuracy1:=1)	Precision: 24-bit. All other components have the value: 0.

8.6 UTC-Time string format specification

A TwinCAT UTC-Time string has the following structure: ‘UT#YYYY-MM-DD-hh:mm:ss.nnnnnnnnn|LFC|A[A]'.

Variable	Representation	Value range	Description
Prefix	UT#	‘UT#‘	Constant prefix.
Year	YYYY	‘1970‘..‘2106‘	Year (4 decimal places).
Month	MM	‘01‘..‘12‘	Month (2 decimal places).
Day	DD	‘01‘..‘31‘	Day (2 decimal places).
Hour	hh	‘00‘..‘23‘	Hour (2 decimal places).
Minute	mm	‘00‘..‘59‘	Minute (2 decimal places).
Second	ss	‘00‘..‘59‘	Second (2 decimal places).
Nanosecond	nnnnnnnnn	‘000000000‘..‘999999999‘	Nanoseconds: (9 decimal places).
LeapSecondsKnown	L	‘0‘..‘1‘	Quality of the leap second ('0':=unknown, '1':=known).
ClockFailure	F	‘0‘..‘1‘	Quality of the timer ('0':=good, '1':=erroneous).
ClockNotSynchronized	C	‘0‘..‘1‘	Quality of the time synchronization ('0':=synchronized, '1':=not synchronized).
Accuracy	A[A]	‘0‘..‘31‘	Accuracy of the time information (precision of the fraction of a second). Max. number of significant bits in the T_UtcTime [▶ 560].fractionOfSecond component (1 or 2 decimal places).

Samples

The table below contains strings and the values of the components after conversion to the structured type: T_UtcTime [▶ 560].

STRING(39)	T_UtcTime	Description
'UT#1970-01-01-00:00:00.000000000|000|0'	STRUCT(0)	All components have the value: 0. Date: January 1, 1970. Time: 00:00:00. Fraction of a second: 0 ms. Precision: 0-bit.
'UT#2018-01-02-03:04:05.125000000|010|3'	STRUCT(secondSinceEpoch:=DT#2018-01-02-03:04:05, fractionOfSecond:=[4,0,0], quality:=(ClockFailure:=1, Accuracy3:=1, Accuracy4:=1))	Date: January 2, 2018. Time: 03:04:05. Fraction of a second: 125000000 ns := 125 ms. Timer signals an error. Precision: 3-bit. All other components have the value: 0.
'UT#2018-01-02-03:04:05.750000000|001|24'	STRUCT(secondSinceEpoch:=DT#2018-01-02-03:04:05, fractionOfSecond:=[3,0,0], quality:=(ClockNotSynchronized:=1, Accuracy0:=1, Accuracy1:=1))	Date: January 2, 2018. Time: 03:04:05. Fraction of a second: 750000000 ns := 750 ms. Timer is not synchronized. Precision: 24-bit. All other components have the value: 0.

8.7 Quality-String format specification

A TwinCAT Quality string has the following structure: 'Q#vv|ORBSFDCA|s|TB'.

Variable	Representation	Value range	Description
Prefix	Q#	‘Q#‘	Constant prefix.
Validity [▶ 778]	vv	‘00‘,‘01‘..‘11‘	Validity.
Overflow	O	'0'..'1'	Overflow.
OutOfRange	R	'0'..'1'	Not in permissible range.
BadReference	B	'0'..'1'	Incorrect reference.
Oscilatory	S	'0'..'1'	Oscillating.
Failure	F	'0'..'1'	Failure.
OldData	D	'0'..'1'	Data not up to date.
Inconsistent	C	'0'..'1'	Inconsistent.
Inaccurate	A	'0'..'1'	Inaccurate.
Source [▶ 778]	s	'0'..'1'	Source (‘0‘:=process, ‘1‘:=replaced).
Test	T	'0'..'1'	Test.
OperatorBlocked	B	'0'..'1'	Operator blocked.

Coding of the Quality.Validity components:

Value	Validity0	Validity1	STRING(2)	Description
Good	0	0	‘00‘	Good (default value).
Invalid	0	1	‘01‘	Invalid.
Reserved	1	0	‘10‘	Reserved.
Questionable	1	1	‘11‘	Questionable.

Coding of the Quality.Source components:

Value	Source	STRING(1)	Description
Process	0	‘0‘	Value comes from the process (default value).
Substituted	1	‘1‘	Value was replaced.

Samples

The table below contains strings and the values of the components after conversion to the structured type: ST_AcsiQuality [▶ 541].

STRING(18)	ST_AcsiQuality	Description
'Q#00|00000000|0|00'	STRUCT(0)	All components have the value: 0.
'Q#01|00000000|1|10'	STRUCT(Validity0:=0, Validity1:=1, Source:=1, Test:=1)	Validity: Invalid. Source: Replaced.Test is set. All other components have the value: 0.
'Q#11|11000001|0|01'	STRUCT(Validity0:=1, Validity1:=1, Overflow:=1, OutOfRange:=1, Inaccurate:=1, OperatorBlocked:=1)	Validity: Questionable.Overflow, OutOfRange, Inaccurate and OperatorBlocked are set. All other components have the value: 0.

8.8 Quality-WORD format specification

The TwinCAT Quality components are coded in WORD in the following way:

Bit	Component	Type	Value range	Description
0	Validity0 [▶ 779]	BIT	0..1	Validity MSB.
1	Validity1 [▶ 779]	BIT	0..1	Validity LSB.
2	Overflow	BIT	0..1	Overflow.
3	OutOfRange	BIT	0..1	Not in permissible range.
4	BadReference	BIT	0..1	Incorrect reference.
5	Oscillatory	BIT	0..1	Oscillating.
6	Failure	BIT	0..1	Failure.
7	OldData	BIT	0..1	Data not up to date.
8	Inconsistent	BIT	0..1	Inconsistent.
9	Inaccurate	BIT	0..1	Inaccurate.
10	Source [▶ 779]	BIT	0..1	Source (‘0‘:=process, ‘1‘:=replaced).
11	Test	BIT	0..1	Test.
12	OperatorBlocked	BIT	0..1	Operator blocked.
13..15	Reserved	BIT	0	Are not used.

Coding of the Quality.Validity components:

Value	Validity0	Validity1	STRING(2)	Description
Good	0	0	‘00‘	Good (default value).
Invalid	0	1	‘01‘	Invalid.
Reserved	1	0	‘10‘	Reserved.
Questionable	1	1	‘11‘	Questionable.

Coding of the Quality.Source components:

Value	Source	STRING(1)	Description
Process	0	‘0‘	Value comes from the process (default value).
Substituted	1	‘1‘	Value was replaced.

Samples

The table below contains strings and the values of the components after conversion to the structured type: ST_AcsiQuality [▶ 541].

WORD	STRING(18)	ST_AcsiQuality	Description
2#0000000000000000	'Q#00|00000000|0|00'	STRUCT(0)	All components have the value: 0.
2#0001001100110010	'Q#01|00110011|0|01'	STRUCT(Validity0:=0, Validity1:=1, BadReference:=1, Oscillatory:=1, Inconsistent:=1, Inaccurate:=1, Source:=0, OperatorBlocked:=1)	Validity: Invalid. Source: Process. BadReference, Oscillatory, Inconsistent, Inaccurate, OperatorBlocked are set. All other components have the value: 0.
2#0000110011001101	'Q#10|11001100|1|10'	STRUCT(Validity0:=1, Validity1:=0, Overflow:=1, OutOfRange:=1, Failure:=1, OldData:=1, Source:=1, Test:=1)	Validity: Reserved. Source: Replaced. Overflow, OutOfRange, Failure, OldData, Test are set. All other components have the value:

8.9 FAQ - frequently asked questions and answers

Frequently asked questions are answered in this section in order to make it easier for you to work with TwinCAT IEC 61850.
If you have any further questions, contact please our Support.

Can a connection be established with several server devices at the same time with the TwinCAT IEC 61850 client? [▶ 780]

? Can a connection be established with several server devices at the same time with the TwinCAT IEC 61850 client?

! Yes, this is possible. One instance of FB_IEC61850ClientClass must be available per connection.

Trademark statements

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.

Third-party trademark statements

Arm, Arm9 and Cortex are trademarks or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

Excel, IntelliSense, Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

MAX®, Stratix®, Cyclone®, Altera®, Agilex™, Arria®, Intel, the Intel logo, Intel Core, Xeon, Intel Atom, Celeron and Pentium are trademarks of Intel Corporation or its subsidiaries.

More Information:
www.beckhoff.com/tf6510

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com