57/612/FDIS

FINAL DRAFT INTERNATIONAL STANDARD
PROJET FINAL DE NORME INTERNATIONALE

	Project number / Numéro de projet: **IEC 61850-7-2 Ed.1**	
	IEC/TC or SC CEI/CE ou SC: **TC 57**	Secretariat / Secrétariat: **Germany**
☒ Submitted for parallel voting in CENELEC Soumis au vote parallèle au CENELEC	Distributed on / Diffusé le: **2002-11-15**	Voting terminates on / Vote clos le: **2003-01-24**
Also of interest to the following committees Intéresse également les comités suivants	Supersedes document Remplace le document 57/522/CDV – 57/595/RVC	
Functions concerned Fonctions concernées ☐ Safety / Sécurité ☐ EMC / CEM ☐ Environment / Environnement ☐ Quality assurance / Assurance de la qualité		

INTERNATIONAL ELECTROTECHNICAL COMMISSION **COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE**

THIS DOCUMENT IS A DRAFT DISTRIBUTED FOR APPROVAL. IT MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, FINAL DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

CE DOCUMENT EST UN PROJET DIFFUSÉ POUR APPROBATION. IL NE PEUT ÊTRE CITÉ COMME NORME INTERNATIONALE AVANT SA PUBLICATION EN TANT QUE TELLE.

OUTRE LE FAIT D'ÊTRE EXAMINÉS POUR ÉTABLIR S'ILS SONT ACCEPTABLES À DES FINS INDUSTRIELLES, TECHNOLOGIQUES ET COMMERCIALES, AINSI QUE DU POINT DE VUE DES UTILISATEURS, LES PROJETS FINAUX DE NORMES INTERNATIONALES DOIVENT PARFOIS ÊTRE EXAMINÉS EN VUE DE LEUR POSSIBILITÉ DE DEVENIR DES NORMES POUVANT SERVIR DE RÉFÉRENCE DANS LES RÈGLEMENTATIONS NATIONALES.

Title

Communication networks and systems in substations - Part 7-2: Basic communication structure for substation and feeder equipment - Abstract communication service interface (ACSI)

Titre

ATTENTION
VOTE PARALLÈLE
CEI – CENELEC

L'attention des Comités nationaux de la CEI, membres du CENELEC, est attirée sur le fait que ce projet final de Norme internationale est soumis au vote parallèle. Un bulletin de vote séparé pour le vote CENELEC leur sera envoyé par le Secrétariat Central du CENELEC.

ATTENTION
IEC – CENELEC
PARALLEL VOTING

The attention of IEC National Committees, members of CENELEC, is drawn to the fact that this final Draft International Standard (DIS) is submitted for parallel voting. A separate form for CENELEC voting will be sent to them by the CENELEC Central Secretariat.

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

COMMUNICATION NETWORKS AND SYSTEMS IN SUBSTATIONS –

Part 7-2: Basic communication structure for substation and feeder equipment – Abstract communication service interface (ACSI)

FOREWORD

1) The IEC (International Electrotechnical Commission) is a worldwide organisation for standardisation comprising all national electrotechnical committees (IEC National Committees). The object of the IEC is to promote international co-operation on all questions concerning standardisation in the electrical and electronic fields. To this end and in addition to other activities, the IEC publishes International Standards. Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organisations liaising with the IEC also participate in this preparation. The IEC collaborates closely with the International Organisation for Standardisation (ISO) in accordance with conditions determined by agreement between the two organisations.
2) The formal decisions or agreements of the IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested National Committees.
3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical specifications, technical reports or guides and they are accepted by the National Committees in that sense.
4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.
5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.
6) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61850-7-2 has been prepared by IEC technical committee 57: Power system control and associated communications.

The text of this standard is based on the following documents:

FDIS	Report on voting
57/XXX/FDIS	57/XXX/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

IEC 61850 consists of the following parts, under the general title *Communication networks and systems in substations:*

Part 1: Basic principles [1]

Part 2: Glossary [1]

Part 3: General requirements

Part 4: System and project management

_____

[1] To be published.

Part 5: Communication requirements for functions and device models [2]

Part 6: Substation automation system configuration language [2]

Part 7-1: Basic communication structure for substation and feeder equipment – Principles and models [2]

Part 7-2: Basic communication structure for substation and feeder equipment – Abstract communication service interface (ACSI) [2]

Part 7-3: Basic communication structure for substation and feeder equipment – Common data classes [2]

Part 7-4: Basic communication structure for substation and feeder equipment – Compatible logical node classes and data classes [2]

Part 8-1: Specific communication service mapping (SCSM) – Mappings to MMS (ISO/IEC 9506-1 and ISO/IEC 9506-2) and to ISO/IEC 8802-3 [2]

Part 9-1: Specific communication service mapping (SCSM) – Sampled values over serial unidirectional multidrop point to point link [2]

Part 9-2: Specific communication service mapping (SCSM) – Sampled values over ISO/IEC 8802-3 [2]

Part 10: Conformance testing [2]

The committee has decided that the contents of this publication will remain unchanged until 2005. At this date the publication will be

- reconfirmed;
- withdrawn;
- replaced by a revised edition; or
- amended.

A bilingual version of this publication may be issued at a later date.

---

[2] To be published.

INTRODUCTION

This document is part of a set of specifications which details a layered substation communication architecture. This architecture has been chosen to provide abstract definitions of classes and services such that the specifications are independent of specific protocol stacks, implementations, and operating systems.

The IEC 61850 series is intended to provide interoperability between a variety of substation and feeder devices. Communication between these devices is achieved by the definition of a hierarchical class model (for example, logical device, logical node, data, data set, report control, or log) and services provided by these classes (for example, get, set, report, define, delete) in parts IEC 61850-7-x.

This part of IEC 61850 defines the abstract communication service interface (ACSI) for use in the utility substation domain that require real-time cooperation of intelligent electronic devices. The ACSI has been defined so as to be independent of the underlying communication systems. Specific communication service mappings[3] (SCSM) are specified in part 8-x (station bus) and part 9-x (process bus) of this standard.

This part of IEC 61850 defines the abstract communication service interface in terms of

– a hierarchical class model of all information that can be accessed via a communication network,

– services that operate on these classes, and

– parameters associated with each service.

The ACSI description technique abstracts away from all the different approaches to implement the cooperation of the various devices.

NOTE 1 Abstraction in ACSI has two meanings. First, only those aspects of a real device (for example, a breaker) or a real function that are visible and accessible over a communication network are modelled. This abstraction leads to the hierarchical class models and their behaviour defined in IEC 61850-7-2, IEC 61850-7-3, and IEC 61850-7-4. Second, the ACSI abstracts from the aspect of concrete definitions on how the devices exchange information; only a conceptual cooperation is defined. The concrete information exchange is defined in the SCSMs.

NOTE 2 This part of IEC 61850 does not provide comprehensive tutorial material. It is recommended that IEC 61850-5 and IEC 61850-7-1 be read first in conjunction with IEC 61850-7-2 and IEC 61850-7-3.

---

[3] The ACSI is independent of the specific mapping. Mappings to standard application layers or middle ware technologies are possible.

COMMUNICATION NETWORKS AND SYSTEMS IN SUBSTATIONS –

Part 7-2: Basic communication structure for substation and feeder equipment – Abstract communication service interface (ACSI)

1 Scope

This part of IEC 61850 applies to the ACSI communication in substations and feeder applications. The ACSI provides the following abstract interfaces.

a) Abstract interface describing communications between a client and a remote server for

- real-time data access and retrieval,
- device control,
- event reporting and logging,
- publisher/subscriber),
- self-description of devices (device data dictionary),
- data typing and discovery of data types, and
- file transfer.

b) Abstract interface for fast and reliable system-wide event distribution between an application in one device and many remote applications in different devices (publisher/subscriber) and for transmission of sampled measured values (publisher/subscriber).

This part of IEC 61850 may also be applied to describe device models and functions for additional activities, such as:

- substation to substation information exchange,
- substation to control centre information exchange,
- power plant to control centre information exchange,
- information exchange for distributed generation, or
- information exchange for metering.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61850-2, *Communication networks and systems in substations – Part 2: Glossary*

IEC 61850-5, *Communication networks and systems in substations – Part 5: Communication requirements for functions and devices models*

IEC 61850-7-1, *Communication networks and systems in substations – Part 7-1: Basic communication structure for substation and feeder equipment – Principles and models*

IEC 61850-7-3, *Communication networks and systems in substations – Part 7-3*: *Basic communication structure for substation and feeder equipment – Common data classes*

IEC 61850-7-4, *Communication networks and systems in substations – Part 7-4: Basic communication structure for substation and feeder equipment – Compatible logical node classes and data classes*

IEC 61850-8-1: *Communication networks and systems in substations – Part 8-1: Specific communication service mapping (SCSM) – Mappings to MMS (ISO/IEC 9506-1 and ISO/IEC 9506-2) and to ISO/IEC 8802-3*

3 Terms and definitions

For the purpose of this document, the terms and definitions provided in IEC 61850-2 and the following definitions apply.

3.1
class
description of a set of objects that share the same attributes, services, relationships, and semantics

3.2
client
entity that requests a service from a server and that receives unsolicited messages from a server

3.3
device
entity that performs control, actuating and/or sensing functions and interfaces to other such entities within an automation system

NOTE Devices alone do not perform energy transport functions.

3.4
external equipment
entity that is stand-alone, or interfaces to an automation system, and that performs energy transport functions

EXAMPLE Transformer, circuit-breaker, line.

NOTE 1 Equipment can contain devices.

NOTE 2 Equipment cannot have a direct connection to the communication network – only devices can be directly connected to the communication network.

3.5
instance (of a class)
entity that has unique identity, to which a set of services can be applied, and which has a state that stores the effects of the services

NOTE Instance is a synonym for the term object.

3.6
Logical device
entity that represents a set of typical substation functions

3.7
Logical node
entity that represents a typical substation function

3.8
physical device
entity that represents the physical parts of a device (hardware and operating system, etc.)

NOTE Physical devices host logical devices.

3.9
PICS – protocol implementation conformance statements
document containing information regarding the ACSI. This could typically be optional parts, specific restrictions, or add-ons

3.10
PIXIT – protocol implementation extra information
document containing information regarding the physical set-up that is not part of the ACSI. This could be information regarding the hardware, socket, and other information

4 Abbreviated terms

ACSI	abstract communication service Interface
BRCB	BUFFERED-REPORT-CONTROL-BLOCK
CDC	common DATA class (IEC 61850-7-3)
CT	current transformer
DA	data attribute
DAT, DAType	data attribute type
DataRef	data reference
dchg	data change trigger option
DS	DATA-SET
dupd	data update trigger option
FC	functional constraint
FCD	functionally constrained DATA
FCDA	functionally constrained DataAttribute
GI	general interrogation
GoCB	GOOSE-CONTROL-BLOCK
GOOSE	generic object oriented substation events
GSE	generic substation event
GsCB	GSSE-CONTROL-BLOCK
GSSE	generic substation status event
IED	intelligent electronic device
IntgPd	integrity period
LCB	LOG-CONTROL-BLOCK
LD	LOGICAL-DEVICE
LN	LOICAL-NODE
MCAA	multicast application association
MMS	manufacturing message specification
MSVCB	MULTICAST-SAMPLED-VALUE-CONTROL-BLOCK
PDU	protocol data unit

PICS	protocol implementation conformance statement
PIXIT	protocol Implementation extra information
qchg	quality change trigger option
SBO	select before operate
SCL	substation configuration language (IEC 61850-6)
SCSM	specific communication service mapping (defined in IEC 61850-8-x and IEC 61850-9-x)
SG	setting group
SGCB	SETTING-GROUP-CONTROL-BLOCK
SoE	sequence-of-events
SVC	sampled value control
TPAA	TWO-PARTY-APPLICATION-ASSOCIATION
TrgOp	trigger option
TrgOpEna	trigger option enabled
UCA™	Utility Communication Architecture
URCB	UNBUFFERED-REPORT-CONTROL-BLOCK
UTC	coordinated universal time
SV	sampled value
USVCB	UNICAST-SAMPLED-VALUE-CONTROL-BLOCK
VT	voltage transformer

5 ACSI overview and basic concepts

5.1 General

The models of the ACSI provide

– the specification of a basic model for the definition of the substation-specific information models contained in IEC 61850-7-3 (common **DATA** classes) and IEC 61850-7-4 (compatible **LOGICAL-NODE** classes and compatible **DATA** classes) and

– the specification of information exchange service models.

The information models and information exchange services are interwoven. From a descriptive point of view, the two aspects are separated to some degree (see the excerpt shown in Figure 1). The common models (for example, **LOGICAL-NODE** and **DATA** classes including their services) are applied in IEC 61850-7-3 and IEC 61850-7-4 to define many specialized information models – the substation automation models.

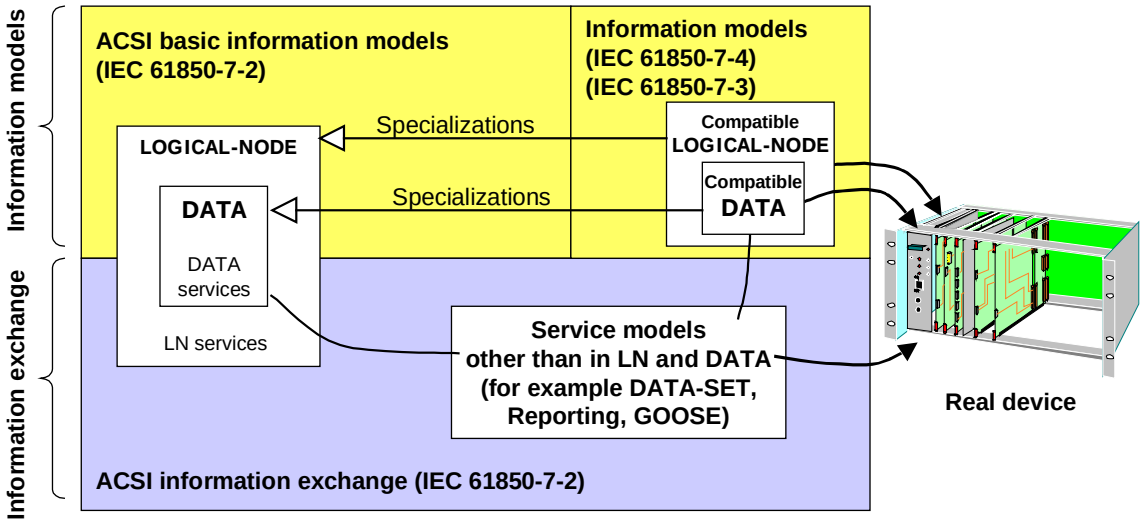


Figure 1 – Excerpt of conceptual model

Other service models required for substation automation systems (for example, **DATA-SET** and reporting provide specific information exchange services) are also defined in this part of the standard; these models are linked to **LOGICAL-NODEs** and **DATA**. The information exchange services are completely defined in the ACSI. The information models defined in IEC 61850-7-4 reference the services defined in the various models of the ACSI.

5.2 Overview of basic information models

The conceptual models to build the domain-specific information models are:

a) **SERVER** – represents the external visible behaviour of a device. All other ACSI models are part of the server.

NOTE 1 A server has two roles: to communicate with a client (most service models in IEC 61850 provide communication with client devices) and to send information to peer devices (for example, for sampled values).

b) **LOGICAL-DEVICE (LD)** – contains the information produced and consumed by a group of domain-specific application functions; functions are defined as **LOGICAL-NODEs**.

c) **LOGICAL-NODE (LN)** – contains the information produced and consumed by a domain-specific application function, for example, overvoltage protection or circuit-breaker.

d) **DATA** – provide means to specify typed information, for example, position of a switch with quality information and timestamp, contained in **LOGICAL-NODEs**.

Each of these information models is defined as a class. The classes comprise attributes and services. The conceptual class diagram of the ACSI is depicted in Figure 2.

NOTE 2 The classes are major building blocks that provide the framework for substation automation device models. Additional details on the modelling and relations between IEC 61850-7-4, IEC 61850-7-3, and this part of IEC 61850 can be found in IEC 61850-7-1.

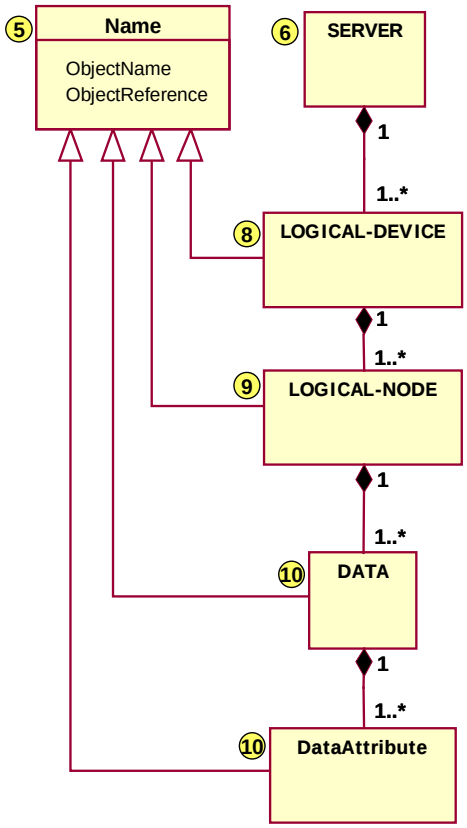


Figure 2 – Basic conceptual class model of the ACSI

NOTE 3 The numbers in the circles indicate the respective clauses in this part of IEC 61850.

The Name class is inherited by the classes **LOGICAL-DEVICE**, **LOGICAL-NODE**, **DATA**, and **DataAttribute**.

EXAMPLE In an implementation the logical device, logical node, data, and data attribute have each an object name (instance name) which is a unique name among classes of the same container to which they belong. In addition, each of the four has an ObjectReference (path name) which is a concatenation of all object names from each container. The four object names (one per column) can be concatenated.

	Logical device	Logical node	Data	Data attribute
Object name	“Atlanta_HV5”	“XCBR1”	“Pos”	“stVal”
Description	High-voltage station 5	Circuit-breaker 1	Position	Status value

5.3 Overview of the other service models

In addition to the models listed above, the ACSI comprise the following models that provide services operating on data, data attributes, and data sets.

a) **DATA-SET** – permits the grouping of data and data attributes. Used for direct access and for reporting and logging.

b) **Substitution** – supports replacement of a process value by another value.

c) **SETTING-GROUP-CONTROL-BLOCK** – defines how to switch from one set of setting values to another one and how to edit setting groups.

d) **REPORT-CONTROL-BLOCK** and **LOG-CONTROL-BLOCK** – describe the conditions for generating reports and logs based on parameters set by the client. Reports may be triggered by changes of process data values (for example, state change or dead band) or by quality changes. Logs can be queried for later retrieval. Reports may be sent immediately or deferred. Reports provide change-of-state and sequence-of-events information exchange.

e) **control blocks for generic substation event** (GSE) – supports a fast and reliable system-wide distribution of input and output data values; peer-to-peer exchange of IED binary status information, for example, a trip signal.

f) **control blocks for transmission of sampled values** – fast and cyclic transfer of samples, for example, of instrument transformers.

g) **control** – describes the services to control, for example, devices.

h) **time and time synchronization** – provides the time base for the device and system.

i) **file transfer** – defines the exchange of large data blocks such as programs.

An overview of the conceptual service model of the ACSI is shown in Figure 3.

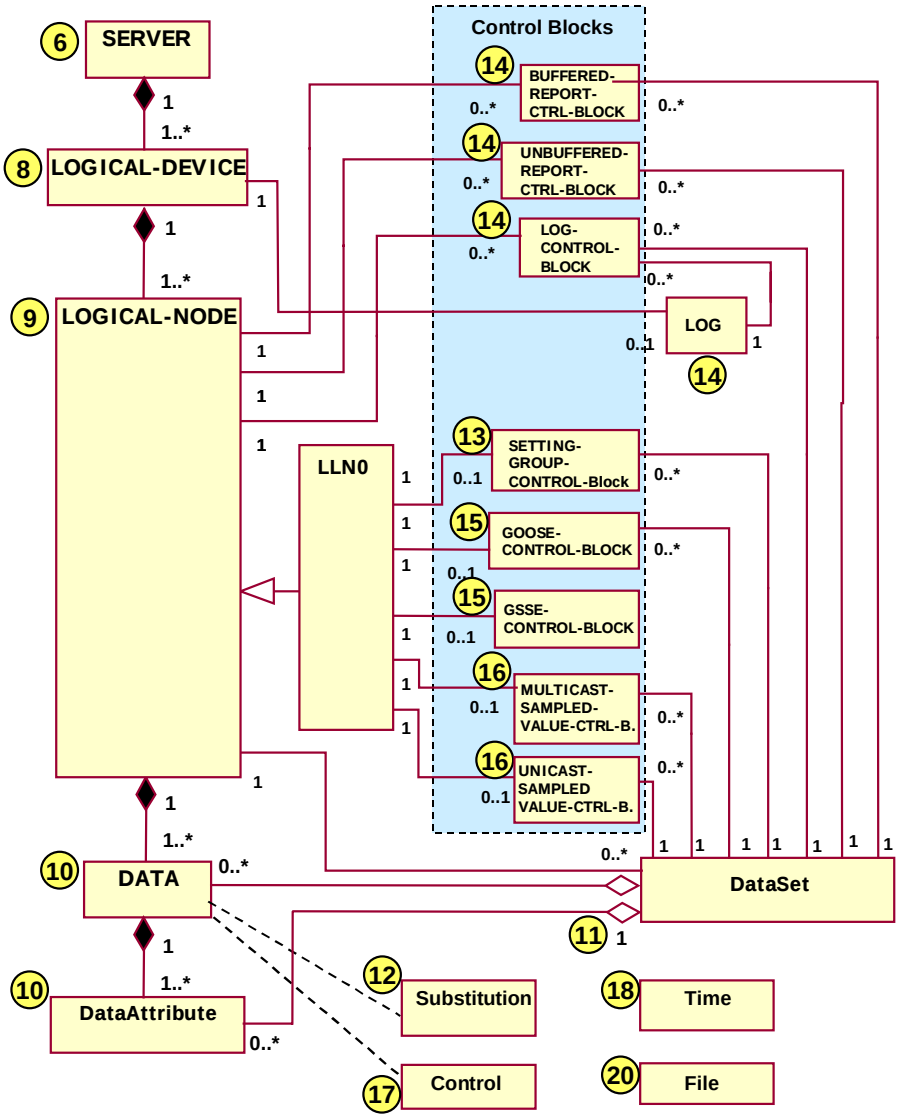


Figure 3 – Conceptual service model of the ACSI

NOTE 1 The numbers in the circles indicate the respective clauses in this part of IEC 61850.

NOTE 2 The class diagrams are conceptual. Details are defined in the respective clauses. Comprehensive diagrams are contained in IEC 61850-7-1. The DATA class may be defined recursively. The operations for substitution and control are restricted to the lowest level in the DATA class. The DataAttributes may be defined recursively as well.

The logical node is one of the major building blocks that has associations to most of the other information exchange models, for example, report control, log control, and setting control.

Any other information exchange service model, for example, report control, log control, and setting control shall inherit the ObjectName and ObjectReference as depicted in Figure 2.

NOTE 3 The class models and services are defined using an object-oriented approach allowing for the mapping of class models and services to different application layer and middle ware solutions.

5.4 Overview of ACSI services

The complete list of ACSI classes and their services is shown in Table 1.

Table 1 – ACSI classes

SERVER model (Clause 6) GetServerDirectory **ASSOCIATION model (Clause 7)** Associate Abort Release **LOGICAL-DEVICE model (Clause 8)** GetLogicalDeviceDirectory **LOGICAL-NODE model (Clause 9)** GetLogicalNodeDirectory GetAllDataValues **DATA model (Clause 10)** GetDataValues SetDataValues GetDataDirectory GetDataDefinition **DATA-SET model (Clause 11)** GetDataSetValues SetDataSetValues CreateDataSet DeleteDataSet GetDataSetDirectory **Substitution model (Clause 12)** SetDataValues GetDataValues **SETTING-GROUP-CONTROL-BLOCK model (Clause 13)** SelectActiveSG SelectEditSG SetSGValues ConfirmEditSGValues GetSGValues GetSGCBValues **REPORT-CONTROL-BLOCK and LCB-BLOCK model (Clause 14)** **BUFFERED-REPORT-CONTROL-BLOCK**: Report GetBRCBValues SetBRCBValues **UNBUFFERED-REPORT-CONTROL-BLOCK:** Report GetURCBValues SetURCBValues	**LOG-CONTROL-BLOCK model:** GetLCBValues SetLCBValues QueryLogByTime QueryLogAfter GetLogStatusValues **Generic substation event model – GSE (Clause 15)** **GOOSE** SendGOOSEMessage GetReference GetGOOSEElementNumber GetGoCBValues SetGoCBValues **GSSE** SendGSSEMessage GetReference GetGSSEDataOffset GetGsCBValues SetGsCBValues **Transmission of sampled values model (Clause 16)** **MULTICAST-SAMPLE-VALUE-CONTROL-BLOCK:** SendMSVMessage GetMSVCBValues SetMSVCBValues **UNICAST-SAMPLE-VALUE-CONTROL-BLOCK:** SendUSVMessage GetUSVCBValues SetUSVCBValues **Control model (Clause 17)** Select SelectWithValue Cancel Operate CommandTermination TimeActivatedOperate **Time and time synchronization (Clause 18)** TimeSynchronization **FILE transfer model (Clause 20)** GetFile SetFile DeleteFile GetFileAttributeValues

5.5 Type definitions

5.5.1 Data attribute types

IEC 61850-7-2 and IEC 61850-7-3 shall use the types that are defined in the following subclauses in order to define the specific data for the application models in IEC 61850-7-4 and the control blocks in this part of IEC 61850 (for example, report control blocks).

The data attribute type concept is depicted in Figure 4. The data attribute type (**DAType**) is a class that has a Name, an indication (Presence) if the attribute is mandatory (present) or optional (possibly not-present), and BasicTypes.

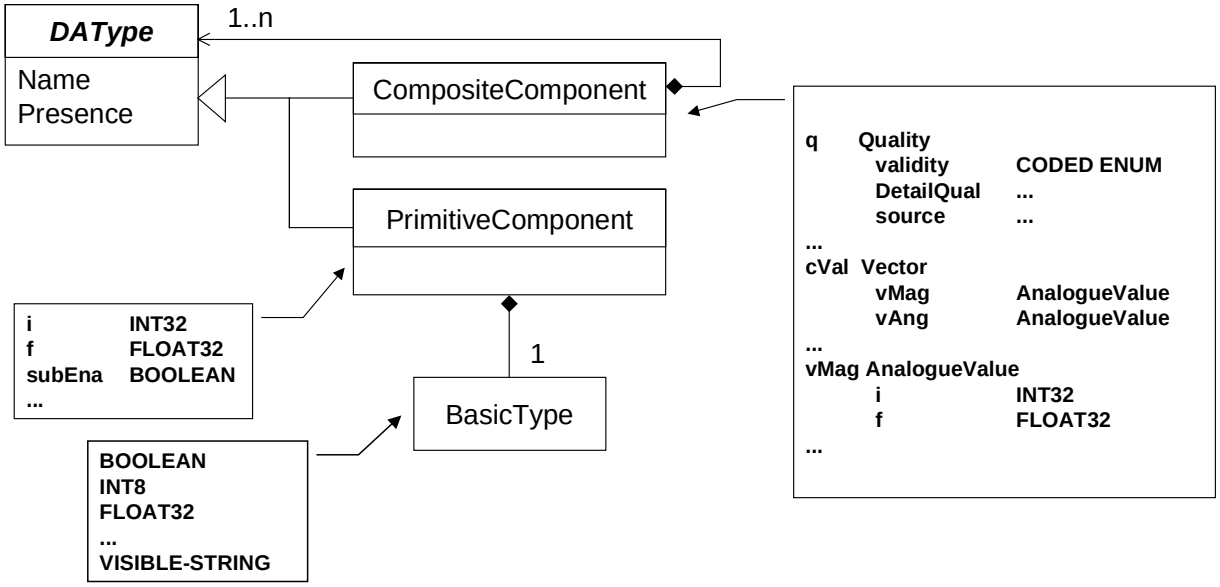
NOTE 1 The **DAType** class is an abstract class that is an auxiliary means to construct the primitive and composite components.

NOTE 2 The formal specification of the **DAType** class and the use of **DATypes** to specify the types of data attributes can be found in Clause 10. The class diagram has been introduced in this subclause to depict the context in which the basic types are used.

NOTE 3 A comprehensive example is provided in IEC 61850-7-1.

The **BasicTypes** (for example, BOOLEAN and INT8) are used to build **PrimitiveComponents** and **CompositeComponents**. **PrimitiveComponents** shall have a Name, a Presence, and a BasicType (for example, Name = **i**, Presence = Mandatory, and BasicType = INT32). The **Composite Component** is constructed by one or more **PrimitiveComponents** each of **BasicType** (for example, Name = **vMag** of type AnalogueValue comprising two **PrimitiveComponents i** (of INT32) and **f** (of FLOAT32)).

Common CompositComponents and PrimitiveComponents are defined in the various common **DATA** classes of IEC 61850-7-3.



NOTE Attribute *Presence* not shown in examples.

Figure 4 – Data attribute type concept

5.5.2 BasicTypes

The **BasicTypes** shall be as listed in Table 2.

Table 2 – BasicTypes

BasicTypes			
Name	**Value range**	**Remark**	**Used by**
BOOLEAN			IEC 61850-7-3 IEC 61850-7-2
INT8	-128 to 127		IEC 61850-7-3 IEC 61850-7-2
INT16	-32,768 to 32,767		IEC 61850-7-3 IEC 61850-7-2
INT24	-8388608 to 8388607	for TimeStamp type	IEC 61850-7-2
INT32	-2,147,483,648 to 2,147,483,647		IEC 61850-7-3 IEC 61850-7-2
INT128	-2**127 to (2**127)-1	Required for counters	IEC 61850-7-3
INT8U	Unsigned integer, 0 to 255		IEC 61850-7-3 IEC 61850-7-2
INT16U	Unsigned integer, 0 to 65,535		IEC 61850-7-3 IEC 61850-7-2
INT32U	Unsigned integer, 0 to 4,294,967,295		IEC 61850-7-3 IEC 61850-7-2
FLOAT32	Range of values and precision as specified by IEEE 754 single-precision floating point		IEC 61850-7-3
FLOAT64	Range of values and precision as specified by IEEE 754 double-precision floating point		IEC 61850-7-3
ENUMERATED	Ordered set of values, defined where type is used	Custom extensions are allowed	IEC 61850-7-3 IEC 61850-7-2
CODED ENUM	Ordered set of values, defined where type is used	Custom extensions shall not be allowed. Type shall be mapped to an efficient encoding in a SCSM	IEC 61850-7-3 IEC 61850-7-2
OCTET STRING	Max. length shall be defined where type is used		IEC 61850-7-3 IEC 61850-7-2
VISIBLE STRING	Max. length shall be defined where type is used		IEC 61850-7-3 IEC 61850-7-2

5.5.3 Common ACSI types

5.5.3.1 General

The common ACSI types shall be used for the attribute definitions of the classes (for example, report control blocks) defined in this part of IEC 61850. The common ACSI types may also be used in the application models defined in IEC 61850-7-3 and IEC 61850-7-4.

5.5.3.2 ObjectName

The ObjectName shall specify a unique instance name among instances of a class owned by the same parent class with a type as specified in Table 3.

Table 3 – ObjectName type

ObjectName type			
Attribute name	**Attribute type**	**Value/value range/explanation**	**Used by**
ObjectName	VISIBLE STRING32	Name of an instance of a class of a single hierarchy level	IEC 61850-7-4 IEC 61850-7-3 IEC 61850-7-2
NOTE Clause 19 specifies constraints on the use of the type ObjectName.			

5.5.3.3 ObjectReference

Instances of classes in the hierarchical information model (ACSI class hierarchy of logical device, logical node, data, data attributes) shall be constructed by the concatenation of all instance names comprising the whole path-name of an instance of a class that identifies the instance uniquely. The type of the ObjectReference shall be as specified in Table 4.

Table 4 – ObjectReference type

ObjectReference type			
Attribute name	**Attribute type**	**Value/value range/explanation**	**Used by**
ObjectReference	VISIBLE STRING255	ObjectReference comprises the whole path-name of an instance of a class that identifies the instance uniquely	IEC 61850-7-2

The ObjectReference syntax shall be:

LDName/LNName[.Name[. ...]]

The "/" shall separate the instance name of a logical device (LDName) from the name of an instance of a logical node (LNName). The "." shall separate the further names in the hierarchy. The "[ ]" shall indicate an option. The inner square bracket "[. ...]" shall indicate further names of recursively nested definitions.

NOTE 1 In any case where the context of the text provides sufficient information that an instance of a class is meant, the term "instance of" is not used.

NOTE 2 Clause 19 specifies constraints on the use of the type ObjectReference.

5.5.3.4 ServiceError type

The service error code for negative service responses (originated within the server) shall be as specified in Table 5.

Table 5 – ServiceError type

ServiceError type definition			
Attribute name	**Attribute type**	**Value /value range/explanation**	**Used by**
ServiceError	ENUMERATED	instance-not-available | instance-in-use | access-violation | access-not-allowed-in-current-state | parameter-value-inappropriate parameter-value-inconsistent | class-not-supported | instance-locked-by-other-client | control-must-be-selected | type-conflict | failed-due-to-communications-constraint | failed-due-to-server-constraint	IEC 61850-7-2

Additional ServiceError values for negative service responses (originated in the application, for example, additional cause diagnosis for control-related services) shall be as specified in the appropriate service models.

NOTE The ServiceError may be extended by an SCSM and the application layer referenced by an SCSM.

5.5.3.5 EntryID type

The type **EntryID** shall represent an arbitrary OCTET STRING used to identify an entry in a sequence of events such as a log or a buffered report as specified by an SCSM.

NOTE 1 The EntryID (handle) allows a client to re-synchronize, for example, with the sequence of the events stored in the IED. The syntax and semantic of the EntryID are outside the scope of this standard.

NOTE 2 The EntryID is used in this part of IEC 61850.

5.5.3.6 Packed list type

The PACKED LIST type shall be as defined in Table 6.

Table 6 – PACKED-LIST type

PACKED-LIST type definition			
Name	**Value range**	**Remark**	**Used by**
PACKED LIST	Ordered list of types; defined where type is used	Any value inside a PACKED LIST shall be mapped to an efficient encoding in a SCSM. No access to individual members of the list is required	IEC 61850-7-3 IEC 61850-7-2

5.5.3.7 TimeStamp type

5.5.3.7.1 General

The relation between a time stamp value, the synchronization of an internal time with an external time source (for example, UTC time), and other time-model-related information are defined in Clause 18.

NOTE 1 The TimeStamp type relies on requirements specified in Clause 18. The reader should first read that clause. The presentation of the TimeStamp is defined in the SCSMs.

NOTE 2 The TimeStamp is used in this part of IEC 61850 and in IEC 61850-7-3.

5.5.3.7.2 TimeStamp syntax

The TimeStamp type shall represent a UTC time with the epoch of midnight (00:00:00) of 1970-01-01 specified in Table 7.

Table 7 – TimeStamp type

TimeStamp type definition			
Attribute name	**Attribute type**	**Value/value range/explanation**	**M/O**
SecondSinceEpoch	INT32	(0...MAX)	M
FractionOfSecond	INT24U	Value = SUM from i=0 to 23 of bi*2**–(i+1); Order = b0, b1, b2, b3, ...	M
TimeQuality	TimeQuality		M

5.5.3.7.3 TimeStamp attributes

5.5.3.7.3.1 SecondSinceEpoch

The SecondSinceEpoch shall be the interval in seconds continuously counted from the epoch 1970-01-01 00:00:00 UTC.

NOTE SecondSinceEpoch corresponds with the Unix epoch.

5.5.3.7.3.2 FractionOfSecond

The attribute FractionOfSeconds shall be the fraction of the current second when the value of the TimeStamp has been determined. The fraction of second shall be calculated as (SUM from I = 0 to 23 of bi*2**–(I+1) s).

NOTE 1 The resolution is the smallest unit by which the time stamp is updated. The 24 bits of the integer provides 1 out of 16777216 counts as the smallest unit; calculated by 1/2**24 which equals approximately 60 ns.

NOTE 2 The resolution of a time stamp may be 1/2**1 (= 0,5 s) if only the first bit is used; or may be 1/2**2 (= 0,25 s) if the first two bits are used; or may be approximately 60 ns if all 24 bits are used. The resolution provided by an IED is outside the scope of this standard.

5.5.3.7.3.3 TimeQuality

The TimeQuality shall provide information about the time source of the sending IED as listed in Table 8.

Table 8 – TimeQuality definition

TimeQuality definition				
Attribute name		**Attribute type**	**Value/Value range/explanation**	**M/O**
		PACKED LIST		
	LeapSecondsKnown	BOOLEAN		M
	ClockFailure	BOOLEAN		M
	ClockNotSynchronized	BOOLEAN		O
	TimeAccuracy	CODED ENUM	Number of significant bits in the FractionOfSecond: Minimum time interval shall be: 2**–n	M

LeapSecondsKnown: The value TRUE of the attribute **LeapSecondsKnown** shall indicate that the value for SecondSinceEpoch takes into account all leap seconds occurred. If it is FALSE then the value does not take into account the leap seconds that occurred before the initialization of the time source of the device.

ClockFailure: The attribute **clockFailure** shall indicate that the time source of the sending device is unreliable. The value of the TimeStamp shall be ignored.

ClockNotSynchronized: The attribute **clockNotSynchronized** shall indicate that the time source of the sending device is not synchronized with the external UTC time.

TimeAccuracy: The attribute **TimeAccuracy** shall represent the time accuracy class of the time source of the sending device relative to the external UTC time. The timeAccuracy classes shall represent the number of significant bits in the FractionOfSecond.

The values of n shall be as listed in Table 9.

NOTE 1 The TimeAccuracy meets the requirements specified in IEC 61850-5 for the selected values of n.

Table 9 – TimeAccuracy

n	Resulting TimeAccuracy (2**–n)	Corresponding time performance class defined in IEC 61850-5
31	–	– unspecified
7	approx. 7,8 ms	10 ms (performance class T0)
10	approx. 0,9 ms	1 ms (performance class T1)
14	approx. 61 µs	100 µs (performance class T2)
16	approx. 15 µs	25 µs (performance class T3)
18	approx. 3,8 µs	4 µs (performance class T4)
20	approx. 0,9 µs	1 µs (performance class T5)

5.5.3.8 EntryTime type

The type **EntryTime** shall represent the time and date as applied internally for the communication, reporting, logging, and subsystem as specified by a SCSM.

NOTE 1 The TimeStamp type is used for common DATA classes in IEC 61850-7-3 and definition of compatible DATA classes in IEC 61850-7-4. The EntryTime type is used for all IEC 61850-7-2 class definitions. The EntryTime type may or may not be the same as TimeStamp in a SCSM.

NOTE 2 The EntryTime is used in this part of IEC 61850.

5.5.3.9 TriggerConditions type

The **TriggerConditions** type shall represent the trigger conditions used to trigger processing reports and logs (see Table 10).

NOTE 1 The TriggerConditions are used in this part of IEC 61850 and in IEC 61850-7-3.

Table 10 – TriggerConditions type

TriggerConditions type			
Attribute name	**Attribute type**	**TriggerOption (TrgOp) for use in DataAttributes**	**Value / value range / explanation**
	PACKED LIST		
data-change	BOOLEAN	dchg	Trigger used in **DATA-Attributes** determined by common **DATA** classes of IEC 61850-7-3
quality-change	BOOLEAN	qchg	Trigger used in **DATA-Attributes** determined by common **DATA** classes of IEC 61850-7-3
data-update	BOOLEAN	dupd	Trigger used in **DATA-Attributes** determined by common **DATA** classes of IEC 61850-7-3
integrity	BOOLEAN	–	Trigger whose value (time) can be set by a service or by configuration; independent of an instance of **DATA**
general-interrogation	BOOLEAN	–	Trigger whose value (initiate general interrogation) can be set by a service or by configuration; independent of an instance of **DATA**

The TriggerOption (**TrgOp**) shall be used in the specification of **DataAttributes** to indicate on which change/update the value of an instance of a **DataAttribute** may be reported or logged.

NOTE 2 Details on the use of **TriggerConditions** are defined in 10.2.2.4.3 and Clause 14.

6 SERVER class model

6.1 SERVER class definition

6.1.1 SERVER class syntax

The class **SERVER** shall represent the externally visible behaviour of a device. The **SERVER** shall be a composition as defined in Table 11.

NOTE 1 For simple devices the server may comprise just one logical device with the GOOSE control model with no other service.

Table 11 – SERVER class definition

SERVER class		
Attribute name	**Attribute type**	**Value/value range/explanation**
ServiceAccessPoint [1..n]	(*)	(*) Type is SCSM specific
LogicalDevice [1..n]	LOGICAL-DEVICE	
File [0..n]	FILE	
TPAppAssociation [0..n]	TWO-PARTY-APPLICATION-ASSOCIATION	
MCAppAssociation [0..n]	MULTICAST-APPLICATION-ASSOCIATION	
Services GetServerDirectory		

NOTE 2 The server's relationship to the underlying communication system and the concrete implementation depend on the SCSM (specific communication service mapping, see IEC 61850-8-x and IEC 61850-9-x) used. Network management (as part of an SCSM), device management, and system management are outside the scope of IEC 61850-7-2.

6.1.2 SERVER class attributes

6.1.2.1 ServiceAccessPoint

The attribute **ServiceAccessPoint** shall identify a **SERVER** within the scope of a system.

NOTE The ServiceAccessPoint is an abstraction of an address used to identify the server in the underlying SCSM. The type depends on the SCSM and should be defined there. A specific ServiceAccessPoint is required by most services to address a server. Nevertheless, it has not been included explicitly in the service parameter tables throughout this part of IEC 61850.

6.1.2.2 LogicalDevice [1..n]

The attribute **LogicalDevice** shall identify a **LogicalDevice** that is contained in a **SERVER**.

6.1.2.3 File [0..n]

The attribute **File** shall identify a **File** contained in a **SERVER**.

6.1.2.4 TPAppAssociation [0..n] – two-party application association

The attribute **TPAppAssociation** shall identify a client with which a **SERVER** maintains a two-party application association.

NOTE Details can be found in Clause 7.

6.1.2.5 MCAppAssociation [0..n] – multicast application association

The attribute **MCAppAssociation** shall identify a subscriber with which a **SERVER** (publisher) maintains a multicast application association.

NOTE Details can be found in Clause 7.

6.2 Server class services

6.2.1 Overview of directory and GetDefinition services

To support self-description of a device several GetXXDirectory and GetXXDefinition services as shown in Figure 5 are specified in this part of IEC 61850.

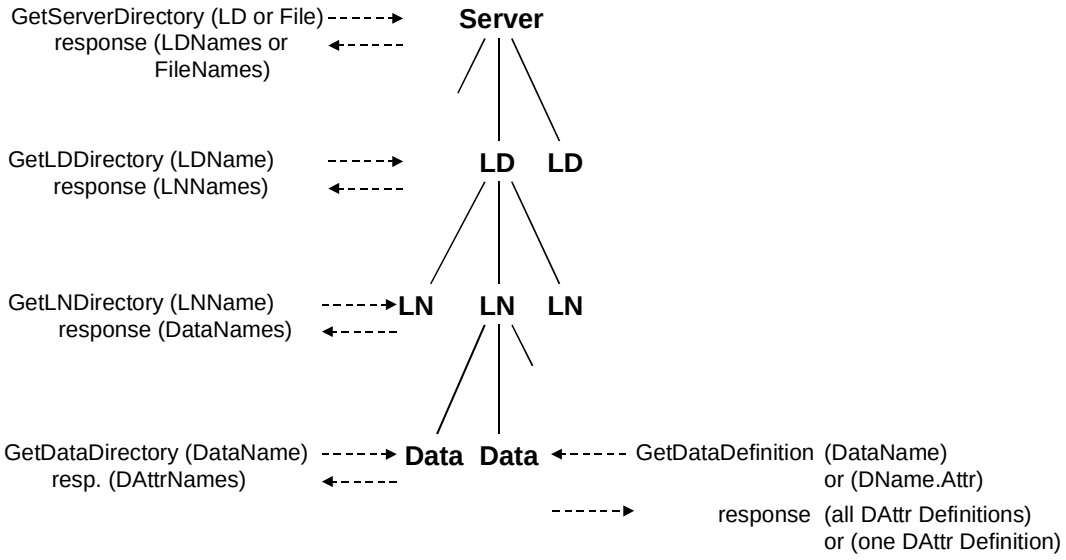


Figure 5 – Overview about GetDirectory and GetDefinition services

A client shall use these services to retrieve the definition of the complete hierarchy – as well as the definition of all accessible information – and of all instances of all underlying classes in a given server.

6.2.2 GetServerDirectory

6.2.2.1 GetServerDirectory parameter table

A client shall use the **GetServerDirectory** service to retrieve a list of the names of all **LOGICAL-DEVICEs** or **Files** made visible and thus accessible to the requesting client by the addressed **SERVER**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
ObjectClass
Response+
Reference [0..n]
Response–
ServiceError

6.2.2.2 Request

6.2.2.2.1 ObjectClass

The parameter **ObjectClass** shall contain the selected class. The client shall select one of the following classes:

– **LOGICAL-DEVICE**

– **FILE**

6.2.2.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. A successful result shall return the following parameter.

6.2.2.3.1 Reference [0..n]

The parameter **Reference** shall contain the ObjectReference of the **LOGICAL-DEVICE** or the **FileName**.

NOTE The FileName type is VISIBLE STRING255.

6.2.2.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

7 Application association model

7.1 Introduction

The application association model consists of provisions on how the communication between the various types of devices is achieved. The model comprises

– class definitions of associations (two-party and multicast); and

– access control concepts (how to restrict access to instances in a server).

The security requirements for the restriction of access to the data in a server is defined in IEC 61850-5.

NOTE Security requirements are implemented by the SCSMs.

7.2 Concept of application associations

The application association model defines

– the services provided for managing associations between client and server (two-party application association); and

– the services provided for managing associations for multicast messaging (for example, GOOSE and transmission of sampled values).

The **two-party application association** class shall convey service requests and responses (thus transferring unconfirmed and confirmed services). The **multicast application association** class shall be capable of conveying unconfirmed services (in one direction only).

Application associations provide a mechanism for controlling the access to the instances of a device (access control).

NOTE The details of an application association model are defined in the SCSMs. The following descriptions provide a conceptual model of the application associations between devices.

7.3 Access control

The access control model provides the capability to restrict the access of a specific client to class instances, class instance attributes, and ACSI services acting upon class instances of a specific server. The ACSI server contains a set of, for example, **LOGICAL-DEVICEs**, **LOGICAL-NODEs**, **DATA**, or report controls. The set of instances visible (and therefore accessible) to a client is restricted on the basis of the identification of the client and the access control specification of the server. This restricted set is called a virtual access view. A virtual access view may not only restrict the visibility of instances or attributes but also the supported service. The concept of a virtual access view is illustrated in Figure 6.

NOTE 1 The virtual access view is the authentication's view of the IED's data model.

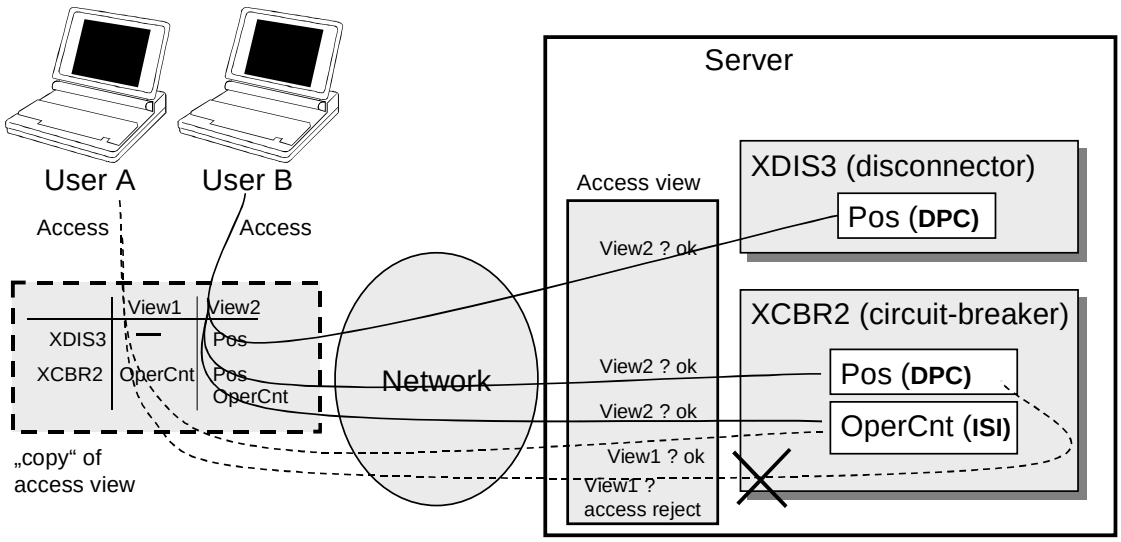


Figure 6 – Access views of a server

Two users (A and B) have different virtual access views (view1 and view2) of the server. View 1 allows just one **DATA** (XCBR.OperCnt) to be accessed remotely. View 2 allows all **DATA** to be accessed.

The intention of IEC 61850 is to implement the **virtual access view in the server** of a device, thus providing **access restriction to any user** who tries to access the instances. Independent of the implementation in the device, additional access restriction may be implemented at the user side, for example, local password or simply a key on the keyboard.

If a view hides a mandatory instance of an attribute of a **DATA** then this hidden attribute shall be implemented as required by the **DATA**.

NOTE 2 A view restricts the visibility to some users only.

A client (or a subscriber in the case of multicast application association) shall be identified by authentication parameters passed to the server when establishing the association with the server (two-party application association) or when sending information over multicast application associations.

NOTE 3 Mechanisms at the client side are outside the scope of this standard. A user may also use a “copy” of the access view to restrict the access at the client side.

NOTE 4 The details of access control including structure and content of authentication parameter are defined in the SCSMs.

7.4 TWO-PARTY-APPLICATION-ASSOCIATION (TPAA) class model

7.4.1 TWO-PARTY-APPLICATION-ASSOCIATION (TPAA) class definition

7.4.1.1 TWO-PARTY-APPLICATION-ASSOCIATION (TPAA) class syntax

A two-party application association type shall provide a bi-directional connection-oriented information exchange. The application associations shall be reliable and the information flow shall be controlled end to end. Reliable means that the connection on which the application association relies provides measures to notify reasons for non-deliverance of information in due time. End-to-end flow control means that sources of information do not send more information than the destination can buffer.

The services for associate, data exchange, and association release of the two-party application association class is depicted in Figure 7.

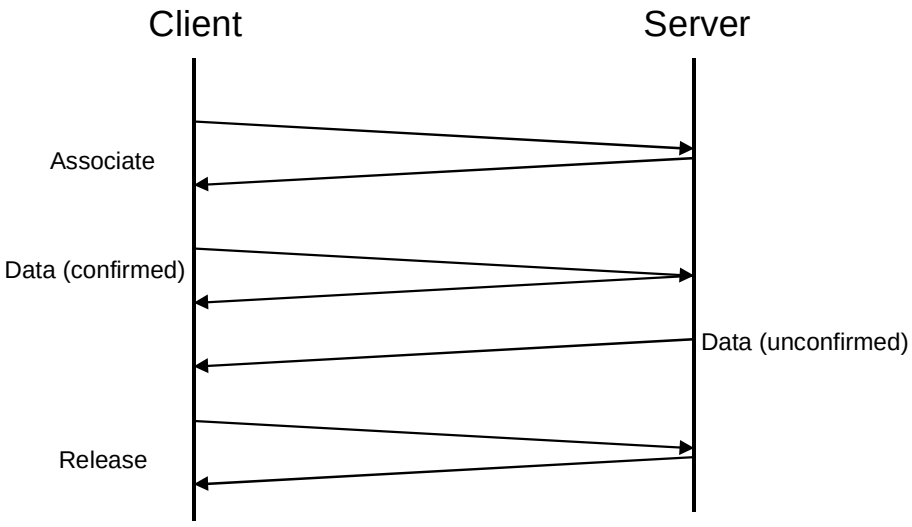


Figure 7 – Normal operation

The abort service for the two-party application association class is depicted in Figure 8.

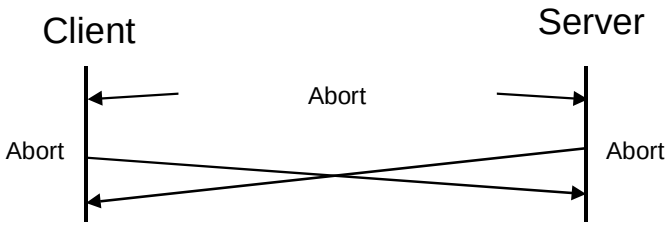


Figure 8 – Aborting association

The **TWO-PARTY-APPLICATION-ASSOCIATION** (TPAA) class shall be defined as in Table 12.

Table 12 – TWO-PARTY-APPLICATION-ASSOCIATION (TPAA) class definition

TWO-PARTY-APPLICATION-ASSOCIATION class		
Attribute name	**Attribute type**	**Value/value range/explanation**
AssociationId	(*)	(*) Type is SCSM specific
AuthenticationParameter	(*)	(*) Type is SCSM specific
Services Associate Abort Release Additional services that make use of the TWO-PARTY-APPLICATION-ASSOCIATION shall be as indicated in Table A.3 of Clause A.4 (in column Asso. marked as “TP”)		

7.4.1.2 TWO-PARTY-APPLICATION-ASSOCIATION (TPAA) class attributes

7.4.1.2.1 AssociationId

The attribute AssociationId shall specify the identification used to identify the application associations.

NOTE The type of the AssociationId is defined in the SCSMs and it may be exchanged in an SCSM or be used locally only.

7.4.1.2.2 AuthenticationParameter

The attribute authenticationParameter shall represent the information required to grant permission to access instances of a specific access view to a server.

NOTE A minimum set of parameters is user identification, view and password. The details are defined in the SCSMs.

7.4.2 Two-party application association services

7.4.2.1 Overview

For **TWO-PARTY-APPLICATION-ASSOCIATION** the following services are defined.

Service	Description
Associate	Establish an association
Abort	Abort an association
Release	Release an association

7.4.2.2 Associate

7.4.2.2.1 Associate parameter

A client shall use the **Associate** service to establish an application association of type two-party with a specific server.

Parameter name
Request
ServerAccessPointReference
AuthenticationParameter
Response+
AssociationId
Result
Response–
ServiceError

7.4.2.2.2 Request

7.4.2.2.2.1 ServerAccessPointReference

The parameter **ServeAccessPointReference** shall identify the server, with which the application association shall be established.

7.4.2.2.2.2 AuthenticationParameter

This parameter **AuthenticationParameter** shall specify the authenticationParameter for this application association to be opened. If an authenticationParameter does not match with a valid parameter, the service request shall be rejected and an appropriate reason shall be returned.

NOTE The type of the authenticationParameter is defined in the SCSM.

7.4.2.2.3 Response+

AssociationId

The parameter **AssociationId** may be used to differentiate the application associations.

NOTE The AssociationId may be exchanged in a response+ message of an SCSM or be used locally only.

7.4.2.2.4 Result

The parameter **Result** shall indicate, if the establishment of the application association was successful or not.

7.4.2.2.5 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

7.4.2.3 Abort

7.4.2.3.1 Abort parameter

The service **Abort** shall be used to abruptly disconnect a specific application association between a client and a server. Abrupt means that all service requests issued shall be discarded – no further service shall be processed.

Parameter name
Request
AssociationId
Reason
Indication
AssociationId
Reason

7.4.2.3.2 Request

7.4.2.3.2.1 AssociationId

The parameter **AssociationId** shall specify the association to be aborted. The indication may be issued by the underlying layer (locally or remotely) or it may be sent from remote user of the association.

7.4.2.3.2.2 Reason

The parameter **Reason** shall specify the reason why the association has been aborted. The reason may be provided by the underlying layer (locally or remotely) or it may be sent from remote user of the association.

7.4.2.3.3 Indication

7.4.2.3.3.1 AssociationId

The parameter **AssociationId** shall specify the association that has been aborted.

7.4.2.3.3.2 Reason

The parameter **Reason** shall specify the reason for abrupt termination the application association.

7.4.2.4 Release

7.4.2.4.1 Release parameter

The service **Release** shall be used to gracefully disconnect a specific application association between a client and a server. Graceful means that all service requests issued shall be completed before termination. New request shall not be issued after disconnect initiation.

Parameter name
Request
AssociationId
Response+
AssociationId
Result
Response–
ServiceError

7.4.2.4.2 Request

7.4.2.4.3 AssociationId

The parameter **AssociationId** shall specify the association to be terminated.

7.4.2.4.4 Response+

7.4.2.4.5 Result

The parameter **Result** shall indicate, if the termination of the application association was successful or not.

7.4.2.4.6 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

7.5 MULTICAST-APPLICATION-ASSOCIATION (MCAA) class

7.5.1 MULTICAST-APPLICATION-ASSOCIATION (MCAA) class definition

7.5.1.1 MULTICAST-APPLICATION-ASSOCIATION (MCAA) class syntax

A multicast application association type shall provide a unidirectional information exchange. Multicast information exchange shall be provided between one source (publisher) and one or many destinations (subscriber). Unidirectional information exchange shall provide sufficient information for the receivers to uniquely interpret the context in which the exchange shall be processed.

The subscriber shall be capable to detect loss and duplication of information received. The receiver shall notify the loss of information to its user and shall discard duplicated information.

NOTE The possible restriction of multicast messages to be exchanged on a single subnet or sent through routers is an issue to be defined in an SCSM.

The multicast application association class is depicted in Figure 9.

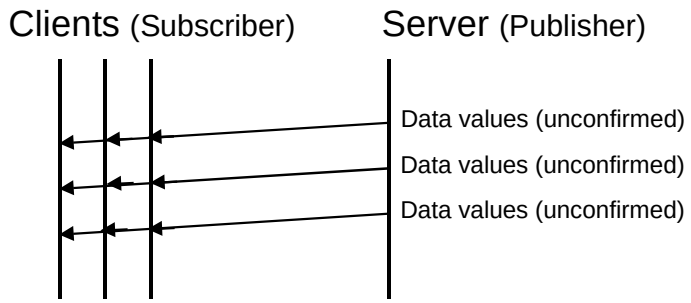


Figure 9 – Principle of multicast application association

The **MULTICAST-APPLICATION-ASSOCIATION (MCAA)** shall be as defined in Table 13.

Table 13 – MULTICAST-APPLICATION-ASSOCIATION (MCAA) class definition

MULTICAST-APPLICATION-ASSOCIATION class		
Attribute name	**Attribute type**	**Value/value range/explanation**
AuthenticationParameter	(*)	(*) Type is SCSM specific
Services Services that make use of the MULTICAST-APPLICATION-ASSOCIATION shall be as indicated in Table A.3 of Clause A.4 (in column Asso. marked as “MC”)		

7.5.1.2 MULTICAST-APPLICATION-ASSOCIATION (MCAA) class attributes

7.5.1.2.1 AuthenticationParameter

The **authenticationParameter** shall represent the information required to grant permission to access instances of a specific access view to a client.

Each multicast service shall provide a service parameter that specifies the authenticationParameter for this data exchange. If an authenticationParameter does not match with a valid parameter, the service request shall be rejected by the receiving device.

NOTE 1 The type of the authenticationParameter is defined in the SCSM.

NOTE 2 Each exchange of information using multicast services can be understood as an “associate message” that carries association parameters and data. The “application association” ceases as soon as the service has been processed.

8 LOGICAL-DEVICE class model

8.1 LOGICAL-DEVICE class definition

8.1.1 LOGICAL-DEVICE class syntax

The **LOGICAL-DEVICE** (**LD**) shall be a composition of **LOGICAL-NODE** as defined in Table 14.

NOTE A **LOGICAL-DEVICE** can be used simply as a container of a group of **LOGICAL-NODEs** or as a device that functions as a gateway or proxy. Details on the use of **LOGICAL-DEVICE** can be found in IEC 61850-7-1.

Table 14 – LOGICAL-DEVICE (LD) class definition

LOGICAL-DEVICE class		
Attribute name	**Attribute type**	**Value/value range/explanation**
LDName	ObjectName	Instance name of an instance of LOGICAL-DEVICE
LDRef	ObjectReference	Path-name of an instance of LOGICAL-DEVICE
LogicalNode [3..n]	LOGICAL-NODE	IEC 61850-7-4 specifies specialized classes of LOGICAL-NODE
Services GetLogicalDeviceDirectory		

8.1.2 LOGICAL-DEVICE class attributes

8.1.2.1 LDName – logical device name

The attribute **LDName** shall unambiguously identify a **LOGICAL-DEVICE** within the scope of a system.

8.1.2.2 LDRef – logical device ObjectReference

The attribute **LDRef** shall be the unique path-name of a **LOGICAL-DEVICE**:

LDName

NOTE The LOGICAL-DEVICE is the root of tree. Therefore the **LDName** and **LDRef** are identical. For conceptual reasons they are both included in the table.

8.1.2.3 LogicalNode [3..n]

The attribute **LogicalNode** shall identify a **LOGICAL-NODE** that is contained in a **LOGICAL-DEVICE**.

Each **LOGICAL-DEVICE** shall have one and only one **LOGICAL-NODE-ZERO** (LLN0), one and only one **LOGICAL-NODE-PHYSICAL-DEVICE** (LPHD), and at least one other **LOGICAL-NODE**.

NOTE The substation-automation-specific LLN0, LPHD, and other logical nodes are defined in IEC 61850-7-4.

8.2 LOGICAL-DEVICE class services

8.2.1 GetLogicalDeviceDirectory

8.2.1.1 GetLogicalDeviceDirectory parameter table

A client shall use the **GetLogicalDeviceDirectory** service to retrieve the list of the ObjectReferences of all **LOGICAL-NODEs** made visible and thus accessible to the requesting client by the referenced **LOGICAL-DEVICE**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter Name
Request
LDReference
Response+
LNReference [3..n]
Response–
ServiceError

8.2.1.2 Request

8.2.1.2.1 LDReference – logical device ObjectReference

The parameter **LDReference** shall contain the ObjectReference **LDRef** of a **LOGICAL-DEVICE**.

8.2.1.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. A successful result shall return the following parameter.

8.2.1.3.1 LNReference [3..n] – logical node ObjectReference

The parameter **LNReference** shall contain the ObjectReference **LNRef** of a **LOGICAL-NODE** from the referenced **LOGICAL-DEVICE**.

8.2.1.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

9 LOGICAL-NODE class model

9.1 LOGICAL-NODE class definition

9.1.1 LOGICAL-NODE class syntax

The **LOGICAL-NODE** shall be a composition of **DATA**, **DATA-SET**, **BRCB**, **URCB**, **LCB**, **LOG**, **SGCB**, **GoCB**, **GsCB**, **MSVCB**, and **USVCB** as defined in Table 15.

Table 15 – LOGICAL-NODE (LN) class definition

LOGICAL-NODE class		
Attribute name	**Attribute type**	**Explanation**
LNName	ObjectName	Instance name of an instance of LOGICAL-NODE
LNRef	ObjectReference	Path-name of an instance of LOGICAL-NODE
Data [1..n]	DATA	
DataSet [0..n]	DATA-SET	
BufferedReportControlBlock [0..n]	BRCB	
UnbufferedReportControlBlock [0..n]	URCB	
LogControlBlock [0..n]	LCB	
IF compatible LN class defined in IEC 61850-7-4 equals LLN0		
SettingGroupControlBlock [0..1]	SGCB	
Log [0..1]	LOG	
GOOSEControlBlock [0..n]	GoCB	
GSSEControlBlock [0..n]	GsCB	
MulticastSampledValueControlBlock [0..n]	MSVCB	
UnicastSampledValueControlBlock [0..n]	USVCB	
Services GetLogicalNodeDirectory GetAllDataValues		
NOTE 1 IEC 61850-7-4 defines specialized logical node classes – the compatible logical node classes, for example, XCBR representing circuit-breakers.		

The definition of **LOGICAL-NODEs** for the substation-application domain is refined by the definition of specific **DATA** in IEC 61850-7-4. The definitions in IEC 61850-7-4 (and IEC 61850-7-3 for the common **DATA** classes) shall be taken into account to get the comprehensive definition of substation-domain-specific **LOGICAL-NODEs**.

NOTE 2 IEC 61850-7-4 defines further attributes for **LOGICAL-NODEs**; for example,, the mode (behaviour: ON, BLOCKED, TEST, etc.) of the substation-specific **LOGICAL-NODE** is defined in IEC 61850-7-4. The state model of a **LOGICAL-NODE** is modelled as a specific **DATA** (named **Mod**).

9.1.2 LOGICAL-NODE class attributes

9.1.2.1 LNName – Logical node name

The attribute **LNName** shall unambiguously identify **LOGICAL-NODE** within the scope of **LOGICAL-DEVICE**.

9.1.2.2 LNRef – Logical node ObjectReference

The attribute **LNRef** shall be the unique path-name of a **LOGICAL-NODE**.

The ObjectReference LNRef shall be:

LDName/LNName

9.1.2.3 Data [1..n]

The attribute **Data** shall identify **DATA** (see Clause 10) that is contained in the **LOGICAL-NODE**.

NOTE IEC 61850-7-4 defines standardized **DATA** called compatible **DATA** classes.

9.1.2.4 DataSet [0..n]

The attribute **DataSet** shall identify a **DATA-SET** (see Clause 11) that is contained in the **LOGICAL-NODE**.

9.1.2.5 BufferedReportControlBlock [0..n]

The attribute **BufferedReportControlBlock** shall identify a **BRCB** (see 14.2) that is contained in the **LOGICAL-NODE**.

9.1.2.6 UnbufferedReportControlBlock [0..n]

The attribute **UnbufferedReportControlBlock** shall identify an **URCB** (see 14.2) that is contained in the **LOGICAL-NODE**.

9.1.2.7 LogControlBlock [0..n]

The attribute **LogControlBlock** shall identify a **LCB** (see 14.3) that is contained in the **LOGICAL-NODE**.

9.1.2.8 SettingGroupControlBlock [0..1]

The attribute **SettingGroupControl** shall identify the **SGCB** (see Clause 13) that is contained in a **LLN0**.

9.1.2.9 Log [0..1]

The attribute **Log** shall identify the **LOG** (see 14.3.3) that is contained in the **LLN0**.

9.1.2.10 GOOSEControlBlock [0..n]

The attribute **GOOSEControlBlock** shall identify a **GoCB** (see 15.2) that is contained in the **LLN0**.

9.1.2.11 GSSEControlBlock [0..n]

The attribute **GSSEControl** shall identify the **GsCB** (see 15.3) that is contained in the **LLN0**.

9.1.2.12 MulticastSampledValueControlBlock [0..n]

The attribute **MulticastSampledValueControlBlock** shall identify a **MSVCB** (see 16.2) that is contained the **LLN0**.

9.1.2.13 UnicastSampledValueControlBlock [0..n]

The attribute **UnicastSampledValueControlBlock** shall identify a **USVCB** (see 16.3) that is contained in the **LLN0**.

9.2 LOGICAL-NODE class services

9.2.1 Overview

For **LOGICAL-NODE** the following services are defined:

Service	Description
GetLogicalNodeDirectory	Retrieve ObjectReferences of a specific ACSI class contained in the **LOGICAL-NODE**
GetAllDataValues	Retrieve all **DataAttribute** values of all **DATA** contained in the **LOGICAL-NODE**

9.2.2 GetLogicalNodeDirectory

9.2.2.1 GetLogicalNodeDirectory parameter table

A client shall use the **GetLogicalNodeDirectory** service to retrieve a list of the ObjectReferences of all instances of a requested class made visible and thus accessible to the requesting client by the referenced **LOGICAL-NODE**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
LNReference
ACSIClass
Response+
InstanceName [0..n]
Response–
ServiceError

9.2.2.2 Request

9.2.2.2.1 LNReference

The parameter **LNReference** shall contain the ObjectReference **LNRef** of the **LOGICAL-NODE**.

9.2.2.2.2 ACSIClass

The parameter **ACSIClass** shall contain the selected ACSI class model for which the ObjectReferences of all ACSI class models shall be returned.

The client shall select one of the following ACSI class models:

DATA, **DATA-SET**, **BRCB**, **URCB**, **LCB**, **LOG**, **SGCB**, **GoCB**, **GsCB**, **MSVCB**, and **USVCB**.

9.2.2.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. A successful result shall return the following parameter.

InstanceName [0..n]

The parameter **InstanceName** shall contain an ObjectName of one requested ACSI class model. In the case where the referenced **LOGICAL-NODE** does not contain the requested ACSI class, the server shall indicate that no ACSI class model exists in this **LOGICAL-NODE**.

9.2.2.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

9.2.3 GetAllDataValues

9.2.3.1 GetAllDataValues parameter table

A client shall use the **GetAllDataValues** service to retrieve all **DataAttribute** values (having the same **FunctionalConstraint**) of all **DATA** made visible and thus accessible to the requesting client by the referenced **LOGICAL-NODE**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
LNReference
FunctionalConstraint [0..1]
Response+
LNReference
DataAttributeReference [1..n]
DataAttributeValue [1..n]
Response–
ServiceError

9.2.3.2 Request

9.2.3.2.1 LNReference

The parameter **LNReference** shall contain the ObjectReference **LNRef** of the **LOGICAL-NODE**.

9.2.3.2.2 FunctionalConstraint [0..1]

The parameter **FunctionalConstraint** shall contain the functional constraint parameter (**FC**) to filter the respective **DataAttributes** of all **DATA** contained in the **LOGICAL-NODE**. The FC shall be as defined in 10.2.2.4.2.

9.2.3.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. A successful result shall return the following parameters.

9.2.3.3.1 DataAttributeReference [1..n]

The parameter **DataAttributeReference** shall contain the ObjectReference of a **DataAttribute** contained in the **LOGICAL-NODE** that shall be returned according to the value of the **FunctionalConstraint** received in the request.

NOTE The ObjectReference **DataAttributeReference** is defined in 10.2.2.4.

9.2.3.3.2 DataAttributeValue [1..n]

The parameter **DataAttributeValue** shall contain the value of a **DataAttribute** of the **DATA** contained in the referenced **LOGICAL-NODE**. Only values of those **DataAttributes** that have the functional constrained equal to the value of the parameter FunctionalConstraint in the service request shall be returned.

9.2.3.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

10 DATA class model

10.1 General

DATA classes represent meaningful information of applications located in an automation devices. The values of **DATA** instances can, for example, be written (**SetDataValues**) and read (**GetDataValues**). IEC 61850-7-4 specifies a list of common and substation-domain-specific – simple and complex – **DATA**, for example, **Pos** for position, **OilFil** for oil filtration. The composition of **DATA** in IEC 61850-7-4 is based on common templates (the common **DATA** classes, **CDC**) specified in IEC 61850-7-3. The concept of **DATA** classes is introduced in this clause. Any set of **DATA** (or parts of **DATA**) instances may be grouped to build **DATA-SET** instances applying the **CreateDataSet** service. **DATA-SET** instances can, for example, be written (**SetDataSetValues**) or read (**GetDataSetValues**)

NOTE 1 The consequences of setting values to instances of **DATA** is outside this part of IEC 61850. IEC 61850-7-3 and IEC 61850-7-4 specify many substation-domain-specific **DATA** classes. These definitions provide information on the actions to be taken by the receiving application, for example, changing the **DATA Mode** from ON to TEST changes the state of the respective instance to test mode behaviour as defined in IEC 61850-7-4.

NOTE 2 To get values **DATA** compares to a polling approach. Services for unsolicited/spontaneous transmission of values of **DATA** from a server to clients (sometimes known as information report, traps, or spontaneous transmission) require a careful design. Uncontrolled spontaneous transmission may congest the network. Services for a controlled reporting are specified in Clause 14.

10.2 DATA class definition

10.2.1 DATA class syntax

The **DATA** class is a key element in IEC 61850. The class diagram in Figure 10 is intended as an introduction to the formal **DATA** class specification.

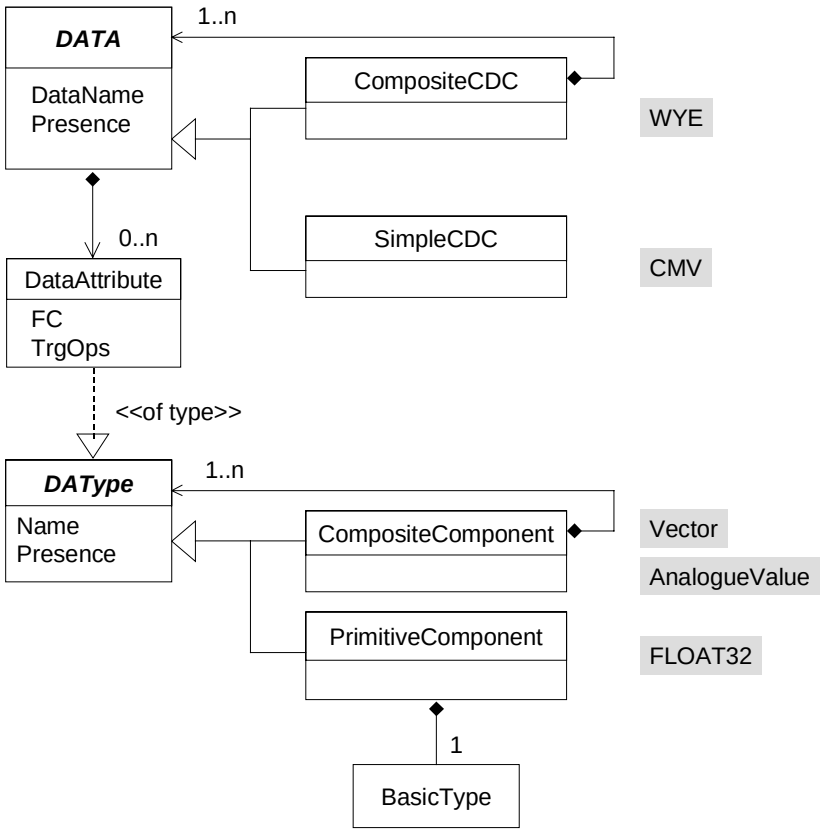


Figure 10 – Class diagram of DATA and DataAttributeType

NOTE 1 The example in Figure 10 uses definitions (for example, **WYE**, **CMV**, **Vector**, and **AnalogueValue** common **DATA** classes) found in IEC 61850-7-3. A comprehensive introduction to the modelling of **DATA** can be found in IEC 61850-7-1.

The **DATA** is a class that has a DataName, an indication (Presence) if the **DATA** is mandatory (present) or optional (not-present), and DataAttributes.

NOTE 2 The **DATA** class is an abstract class that is an auxiliary means to construct the primitive and composite common data classes.

NOTE 3 The following examples used in the text refer to Figure 11.

The DataAttributes (for example, cVal – complex value) are used to build a **SimpleCDC** (simple common data class) and **CompositeCDC** (composite common data class). **SimpleCDC** shall have a DataName, a Presence, and **DataAttributes** (for example, DataName = phsA, Presence = Mandatory, and DataAttribute = cVal). The **CompositeCDC** is constructed by one or more **SimpleCDC** and/or **DataAttributes** (for example, CDC WYE comprising a **SimpleCDC** CMV, etc.).

The **DAType** has already been explained in 5.5.1.

Figure 11 depicts an excerpt of a **DATA** instance (contained in a **LOGICAL-NODE** MMXU1). The instance of the **LOGICAL-NODE** with the name MMXU1 (instantiated from **MMXU**) is composed of the instance of the **DATA** phase voltage named **PhV** (instantiated from **WYE**), which is composed of phase A voltage **phsA** (instantiated from **CMV**), which is composed of complex value **cVal** (of type **Vector**), which is composed of voltage **vMag** (of type **AnalogueValue**), which is composed of floating-point value **f** (of type **FLOAT32**). The DataAttribute has additionally the functional constraint **FC** = **MX** (measurand) and the trigger option **TrgOp** = **dchg** (data-change).

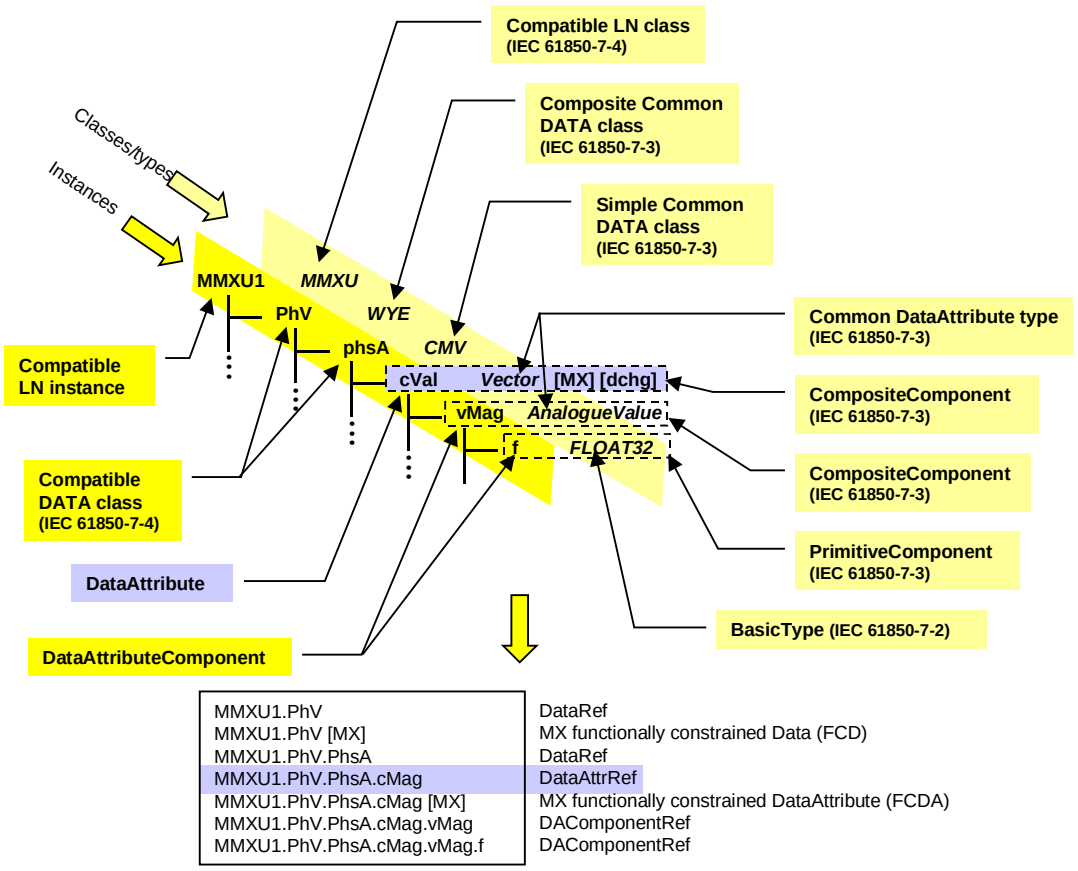


Figure 11 – Example of DATA

NOTE 4 The explanation of the **DATA** class refers to the example shown in Figure 11. The example uses some definitions from IEC 61850-7-3 just to demonstrate the formal definition of the **DATA** class. A complete definition of the compatible classes is defined in IEC 61850-7-3.

The references for the various levels are listed at the bottom of the figure.

The **DATA** shall have the structure defined in Table 16.

The inheritance and relations between the classes **DATA**, **CompositeCDC**, **SimpleCDC**, and **DAType** shall be as shown in Figure 10.

The table notation does not easily show the inheritance. Therefore the class diagram in Figure 10 shall be normative. The tables and the class diagrams shall be be used together.

Table 16 – DATA class definition

DATA class		
Attribute name	**Attribute type**	**Value/value range/explanation**
DataName	ObjectName	Instance name of an instance of DATA, for example, **PhV** (1st level), **phsA** (2nd level)
DataRef	ObjectReference	Path-name of an instance of DATA, for example, **MMXU1.PhV** or for example, **MMXU1.PhV.phsA**
Presence	BOOLEAN	Indicates mandatory/optional
DataAttribute [0..n] DataAttributeType FunctionalConstraint TrgOp [0..n]	 DAType FC TriggerConditions	 For example, **Vector** class of IEC 61850-7-3 for example, **MX** for example, **dchg**
Specializations of DATA		
CompositeCDC [0..n]	DATA	For example, **WYE** class of IEC 61850-7-3
SimpleCDC [0..n]	COMMON-DATA	For example, **CMV** class of IEC 61850-7-3
Services GetDataValues SetDataValues GetDataDirectory GetDataDefinition		

An instance of a **DATA** class may contain zero or more instances of a **CompositeCDC**, **SimpleCDC** or a **DataAttribute**. However, they cannot all be absent, so at least one of these elements shall be present.

NOTE 5 The structure of a **DATA class** is recursive since a **CompositeCDC** is also of type **DATA class**. The level of recursion may be restricted by a SCSM, so the number of levels of recursion of **CompositeCDCs** is normally no greater than 1.

NOTE 6 **DATA** or part of a **DATA** may be referenced in a **DATA-SET**. The persistent existence of **DATA** is expected as long as they are referenced as members of a **DATA-SET**. A system has to take special measures to ensure their existence.

10.2.2 DATA class attributes

10.2.2.1 DataName

The attribute **DataName** shall unambiguously identify a **DATA** within the scope of a **LOGICAL-NODE**.

10.2.2.2 DataRef – data ObjectReference

The attribute **DataRef** shall be the unique path-name of a **DATA**.

The ObjectReference **DataRef** shall be:

LDName/LNName.DataName[.DataName[. ...]]

NOTE Nesting depends on the concrete definition of a **DATA** class.

10.2.2.3 Presence

The attribute **Presence** of type BOOLEAN shall specify if a **DATA** within a compositeCDC or a **LOGICAL-NODE** is mandatory (Presence = TRUE) or optional (Presence = FALSE).

10.2.2.4 DataAttribute

10.2.2.4.1 DataAttributeType

10.2.2.4.1.1 General

The attribute **DataAttributeType** of type **DAType** shall specify a data attribute.

10.2.2.4.1.2 DAType syntax

The **DAType** shall be as defined in Table 17.

Table 17 – DAType definition

DAType		
Attribute name	**Attribute type**	**Value/value range/explanation**
DATName	ObjectName	Instance name of an instance of DAType, for example, **cVal** (1st level), **vMag** (2nd level), **f** (3rd level)
DATRef	ObjectReference	Path-name of an instance of DAType for example, **MMXU1.PhV.phsA.cVal** for example, **MMXU1.PhV.phsA.cVal.vMag** or for example, **MMXU1.PhV.phsA.cVal.vMag.f**
Presence	BOOLEAN	Indicates mandatory/optional
Specializations of DAType		
CompositeComponent [0..n]	DAType	For example, **vMag** in Vector class of IEC 61850-7-3 for example, **f** in AnalogueValue of IEC 61850-7-3
PrimitiveComponent [0..1]	BasicType	For example, **FLOAT32** class of IEC 61850-7-3 for **f**
NOTE 1 An instance of a **DAType** may contain 0 or more instances of a **CompositeComponent** or a **PrimitveDAT**. However, they cannot both be absent, so at least one of these elements must be present. NOTE 2 The structure of a **DAType** is recursive since a **CompositeComponent** is also of type **DAType**. The level of recursion may be restricted by a SCSM, so the number of levels of recursion of **CompositeComponents** is normally no greater than 2.		

DATName – data attribute type name

The attribute **DATName** shall unambiguously identify a **DAType** within the scope of a **DataAttribute** or a nested **DataAttribute**.

The DATName (if DataAttribute is not nested) or the DATName of the first level (if Data Attribute is nested) shall be called the **DataAttributeName**.

For the second and any deeper nesting levels the DATName shall be called **DAComponentName**.

The ObjectReference from the top (LD) down to the **DataAttributeName** shall be called **DataAttributeReference**.

EXAMPLE As shown in Figure 11, the **cVal** (derived from a common data attribute type – **Vector**) is the DataAttribute. The **vMag** (also derived from a common data attribute type – **AnalogueValue**) is a DataAttributeComponent.

DATRef – data attribute type ObjectReference

The attribute **DATRef** shall be the unique path-name of a **DAType**.

The ObjectReference **DATRef** shall be:

LDName/LNName.
DataName[.DataName[. ...]].DataAttributeName[.DAComponentName[. ...]]

The ObjectReference **DataAttributeReference** shall be:

LDName/LNName.DataName[.DataName[. ...]].DataAttributeName

NOTE 3 Nesting depends on the concrete definition of a **DATA** class and **DAType** class.

NOTE 4 In each path within a **DATA** there is one and only one DataAttribute (level).

Presence

The attribute **Presence** of type BOOLEAN shall specify if a **DataAttribute** is mandatory (Presence = TRUE) or optional (Presence = FALSE).

CompositeComponent [0..n] – composite component

The attribute **CompositeComponent** shall be a specialization of **DAType**.

PrimitiveComponent [0..n] – primitive component

The attribute **PrimitiveComponent** shall be a specialization of **DAType**.

10.2.2.4.2 FC [0..1] – functional constraint

From an application point of view, the **DataAttributes** are classified according to their specific use; for example, some attributes are used for **controlling** purposes, other attributes are used for **reporting** and **logging**, **configuration**, others indicate **measurements** or **setting groups**, or some identify the **description** of a specific **DataAttribute**.

The **functional constraint** (**FC**) shall be a property of the **DataAttribute** characterizing the specific use of the **DataAttribute**. The **functional constraint** (**FC**) is used in the definition of DATA (contained in LOGICAL-NODES) and in the various control blocks (for example, **BRCB**). Most attributes of control blocks have a **functional constraint** (**FC**) property.

NOTE The functional constraint could be understood as a filter of the DataAttributes. The common data classes in IEC 61850-7-3 use the functional constraint values defined in this subclause.

The **functional constraint** is used in various definitions in this part of IEC 61850. The **functional constraint** (**FC**) shall indicate the services that are allowed to be operated on a specific **DataAttribute**. The **functional constraints** shall be as specified in Table 18.

Table 18 – Functional constraints

Functional constraint (FC)					
	Semantic	**Services allowed**	**Initial values/storage/ explanation**	**D[a]**	**CB[b]**
ST	Status information	**DataAttribute** shall represent a status information whose value may be read, substituted, reported, and logged but shall not be written	Initial value of the **DataAttribute** shall be taken from the process	X	
MX	Measurands (analogue values)	**DataAttribute** shall represent a measurand information whose value may be read, substituted, reported, and logged but shall not be written	Initial value of the **DataAttribute** shall be taken from the process	X	
CO	Control	**DataAttribute** shall represent a control information whose value may be operated (control model) and read	N.a.	X	
SP	Setpoint	**DataAttribute** shall represent a set-point information whose value may be controlled (control model) and read. Values controlled shall become effective immediately	Initial value of the **DataAttribute** shall be as configured; value shall be non-volatile	X	
SV	Substitution	**DataAttribute** shall represent a substitution information whose value may be written to substitute the value attribute and read	If the value of the **DataAttribute** is volatile then the initial value shall be FALSE, else the value should be as set or configured	X	
CF	Configuration	**DataAttribute** shall represent a configuration information whose value may be written and read. Values written may become effective immediately or deferred by reasons outside the scope of this standard	Initial value of the **DataAttribute** shall be as configured; value shall be non-volatile	X	
DC	Description	**DataAttribute** shall represent a description information whose value may be written and read	Initial value of the **DataAttribute** shall be as configured; value shall be non-volatile	X	
SG	Setting group	Logical devices that implement the **SGCB** class maintain multiple grouped values of all instances of **DataAttributes** with functional constraint SG. Each group contains one value for each **DataAttribute** with functional constraint SG which shall be the current active value (for details see 13). Values the of **DataAttributes** with **FC**=**SG** shall not be writeable	Initial value of the **DataAttribute** shall be as configured; value shall be non-volatile		X
SE	Setting group ditable	**DataAttribute** which can be edited by **SGCB** services	Value of the **DataAttribute** shall be as available after **SelectEditSG** service has been processed	X	
EX	Extended definition	**DataAttribute** shall represent an extension information providing a reference to a name space. Extensions are used in conjunction with extended definitions of **LNs**, **DATA**, and **DataAttributes** in 61850-7-3 and IEC 61850-7-4. Values the of **DataAttributes** with **FC**=**EX** shall not be writeable	Value of the **DataAttribute** shall be as configured; value shall be non-volatile	X	
BR	Buffered report [c]	**Attribute** shall represent a report control information of a **BRCB** whose value may be written and read	Initial value of the **Attribute** shall be as configured; value shall be non-volatile		X
RP	Unbuffered report [c]	**Attribute** shall represent a report control information of a **URCB** whose value may be written and read	Initial value of the **Attribute** shall be as configured; value shall be non-volatile		X

	Semantic	Services allowed	Initial values/storage/ explanation	D[a]	CB[b]
LG	Logging [c]	**Attribute** shall represent a log control information of a **LCB** whose value may be written and read	Initial value of the **Attribute** shall be as configured; value shall be non-volatile		X
GO	Goose control [c]	**Attribute** shall represent a goose control information of a **GoCB** whose value may be written and read	Initial value of the **Attribute** shall be as configured; value shall be non-volatile		X
GS	Gsse control [c]	**Attribute** shall represent a goose control information of a **GsCB** whose value may be written and read	Initial value of the **Attribute** shall be as configured; value shall be non-volatile		X
MS	Multicast sampled value control [c]	**Attribute** shall represent a sampled value control information of a **MSVCB** whose value may be written and read	Initial value of the **Attribute** shall be as configured; value shall be non-volatile		X
US	Unicast sampled value control [c]	**Attribute** shall represent a sampled value control information of an instance of a UNICAST-SVC whose value may be written and read	Initial value of the **Attribute** shall be as configured; value shall be non-volatile		X
XX	Representing all **DataAttributes** as a service parameter	Shall represent all **DataAttributes** of a **DATA** (of any **FC**) to be accessed, for example, to be written and read. The FC value "xx" shall only be used in the functionally constrained data (**FCD**); "XX" shall not be used as **FC** value in a **DataAttribute**	"XX" shall be used as a wildcard in services only		

NOTE The possibility to write an **Attribute** or a **DataAttribute** may be further constrained by a view or an implementation.

[a] Column D indicates the use of the **FC** in the definition of **DATA** (i.e. common **DATA** classes in IEC 61850-7-3).

[b] Column CB indicates the use of the **FC** in the definition of control blocks in this part of IEC 61850.

[c] Reserved for control classes in this part of IEC 61850.

EXAMPLE The common data attribute for the common data class **single-point status** (**SPS**) according to IEC 61850-7-3 has the following **DataAttributes**: **stVal** (status value), **q** (quality), and **t** (time stamp) with the functional constraint **ST** (status information).

10.2.2.4.3 TrgOp [0..n] – trigger option

The attribute **TrgOp** of type **TriggerConditions** (see Table 10) shall specify the trigger conditions (associated with a DataAttribute of a **DATA**) that may cause a report to be sent or a log entry to be stored into a log (report model; see Clause 14). The services associated with the **TriggerConditions** shall be as specified in Table 19.

Table 19 – Trigger option

TrgOp	Semantic	Services allowed
dchg	data-change	A report or a log entry shall be generated due to a change of the value of the data attribute
qchg	quality-change	A report or a log entry shall be generated due to a change of the value of the quality attribute
dupd	data value update	A report or a log entry shall be generated due to freezing the value of a freezable attribute or updating the value of any other attribute. An updated value may have the same value as the old value

NOTE The trigger conditions integrity and general-interrogation of the TriggerConditions type (see Table 10) are used independent of instances of **DATA**; they can be set from remote by services and thus trigger sending reports or placing log entries into logs.

As depicted in Figure 12 the value of a **DataAttribute** that provides a specific TrgOp (trigger option) shall be monitored for reporting and logging if the report control block has enabled the

specific trigger option (TrgOpEna). In the upper example of Figure 12 the TrgOpEna is dchg; the TrgOp of the DataAttributes is dchg for the first, dupd for the second, and qchg for the last DataAttribute. Reports are sent on data changes only, because only dchg is enabled in the report control block. In the second example, all changes will be reported. In addition, a report will be sent on the expiration of the integrity period.

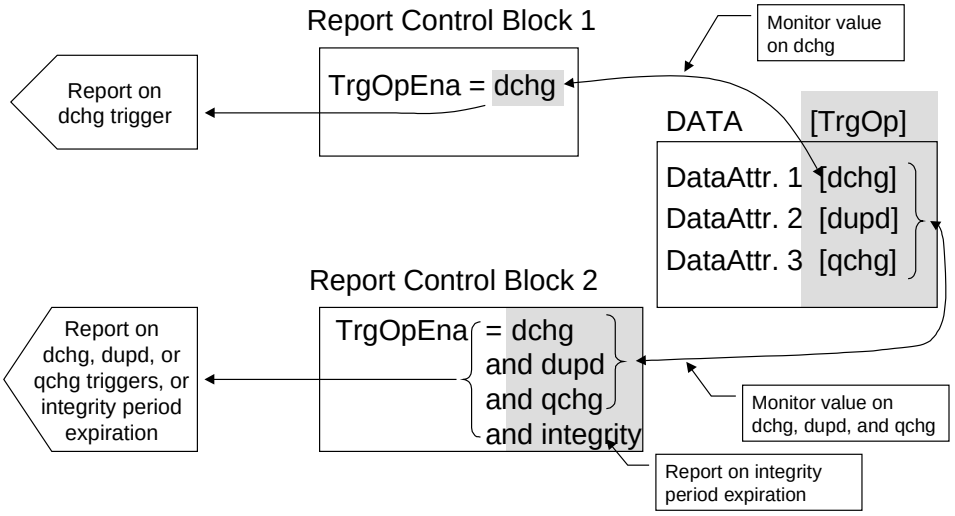


Figure 12 – Relation of TrgOp and Reporting

DATA whose **DataAttributes** shall be monitored for change detection shall be referenced by a **DATA-SET**.

EXAMPLE Common data attributes in IEC 61850-7-3, for example, **stVal** (status value) provides a trigger option dchg, the common data attribute **q** (quality) provides the trigger option qchg.

NOTE The data attributes of **DATA-SET** which will be reported or logged after a change has been detected depend on the definition of the data set used for reporting. For details see Clause 11.

10.2.2.4.4 Functionally constrained data (FCD)

The reference of an ordered collection of **DataAttributes** of a **DATA** having the same **functional constraint** (**FC**) value shall be called **functionally constrained data** (**FCD**). The order of the collection of the **FCD** shall be the order of the appearance of the **DataAttributes** in the **DATA**. A **functionally constrained data** shall be defined as the **DataRef** accompanied by a value of a **functional constraint** (**FC**).

NOTE All measured values of a **DATA** (FC = MX) are referenced by the **measurement FCD**. The functionally constrained data is used, for example, to describe and to remotely create **DATA-SETs**. The syntax notation for **FCD** is defined in a SCSM.

EXAMPLE Figure 11 shows a [MX] FCD in the second line.

10.2.2.4.5 Functionally constrained data attribute (FCDA)

A reference of a single **DataAttribute** of a **DATA** having a specific **functional constraint** (**FC**) value shall be called **functionally constrained data attribute** (**FCDA**). A functionally constrained data attribute shall be defined as a **DataAttributeReference** accompanied by a value of a **functional constraint** (**FC**).

NOTE A single measured value of a **DATA** (FC = MX) is referenced by an **FCDA**. The functionally constrained data attribute is used, for example, to describe and to remotely create **DATA-SETs**. The syntax notation for **FCDA** is defined in a SCSM.

EXAMPLE Figure 11 shows a [MX] FCDA in the fifth line.

10.2.2.5 CompositeCDC [0..n]

The attribute **CompositeCDC** shall be a specialization of **DATA**.

10.2.2.6 SimpleCDC [0..n]

10.2.2.6.1 SimpleCDC syntax – General

The attribute **SimpleCDC** shall be a specialization of **DATA**.

10.2.2.6.2 COMMON-DATA class syntax

The **COMMON-DATA** class shall be as defined in Table 20.

Table 20 – COMMON-DATA class definition

COMMON-DATA class		
Attribute Name	**Attribute Type**	**Value/value range/explanation**
DataName	ObjectName	Instance name of an instance of DATA, for example, **PhV** (1st level), **phsA** (2nd level),
DataRef	ObjectReference	Path-name of an instance of DATA, for example, **MMXU1.PhV** or for example, **MMXU1.PhV.phsA**
Presence	BOOLEAN	Indicates mandatory/optional
DataAttribute [1..n] DataAttributeType FunctionalConstraint TrgOp [0..n]	 DAType FC TriggerConditions	 For example, **Vector** class of IEC 61850-7-3 for example, **MX** for example, **dchg**
Services GetDataValue SetDataValue GetDataDirectory GetDataDefinition		
NOTE 1 The **CommonDATA** is a subclass of the **DATA class**. NOTE 2 **DATA** or DataAttribute may be referenced in a **DATA-SET**. The persistent existence of **DATA** and DataAttribute is expected as long as they are referenced as members of a **DATA-SET**. A system has to take special measures to ensure their existence. NOTE 3 IEC 61850-7-2 defines the basic class model. IEC 61850-7-3 defines specialized **DATA** classes – the common **DATA** classes, for example, **SPS** modelling a single-point status **DATA** class. IEC 61850-7-4 defines specialized common **DATA** classes – the compatible **DATA** classes, for example, **Pos** modelling a position (specializing an **SPS** common **DATA** class).		

DataName

The attribute **DataName** shall identify a **DATA** within the scope of a **LOGICAL-NODE** or a nested **DATA**.

DataRef – data ObjectReference

The attribute **DataRef** shall be the unique path-name of a **DATA**.

The ObjectReference **DataRef** shall be:

LDName/LNName.DataName[.DataName[. ...]]

NOTE Nesting depends on the concrete definition of a **DATA** class.

Presence

The attribute **Presence** of type BOOLEAN shall specify if a **DATA** is mandatory (Presence = TRUE) or optional (Presence = FALSE).

DataAttribute

The attribute **DataAttribute** shall be as defined in 10.2.2.4.

10.3 Relation of DATA, common DATA, and compatible DATA classes

The **DATA** defines a class that is specialized in IEC 61850-7-3 to define the common **DATA**. IEC 61850-7-4 specializes the common **DATA** (to define the compatible **DATA**). The relation between these parts is depicted in Figure 13.

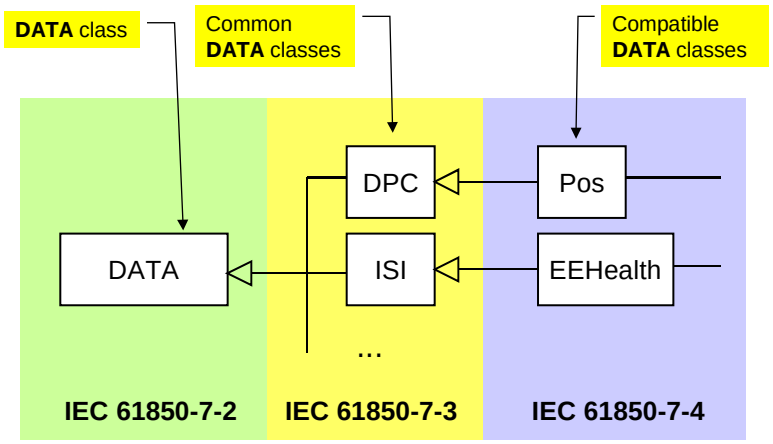


Figure 13 – Relation of DATA classes

NOTE The common **DATA** class in IEC 61850-7-3 "adds" common structures (the **DataAttributes**) to the **DATA** class; the compatible **DATA** class in IEC 61850-7-4 "adds" specific semantic to a specialized common **DATA** class.

EXAMPLE The compatible **DATA** class with the name "Pos" represents a switch position. "Pos" is a specialization of the common **DATA** class "DPC" (double-point control). The **DATA** "Pos" may be used in one or several **LOGICAL-NODEs.**

10.4 DATA class services

10.4.1 General definitions and overview

For **DATA** the following services are defined.

Service	Description
GetDataValues	Retrieve values of **DATA** contained in the **LOGICAL-NODE**
SetDataValues	Write values of **DATA** contained in the **LOGICAL-NODE**
GetDataDirectory	Retrieve ObjectReferences of all **DataAttributes** contained in the **DATA**
GetDataDefinition	Retrieve definitions of all **DataAttributes** contained in the **DATA**

Excerpts of the four services are depicted in Figure 14.

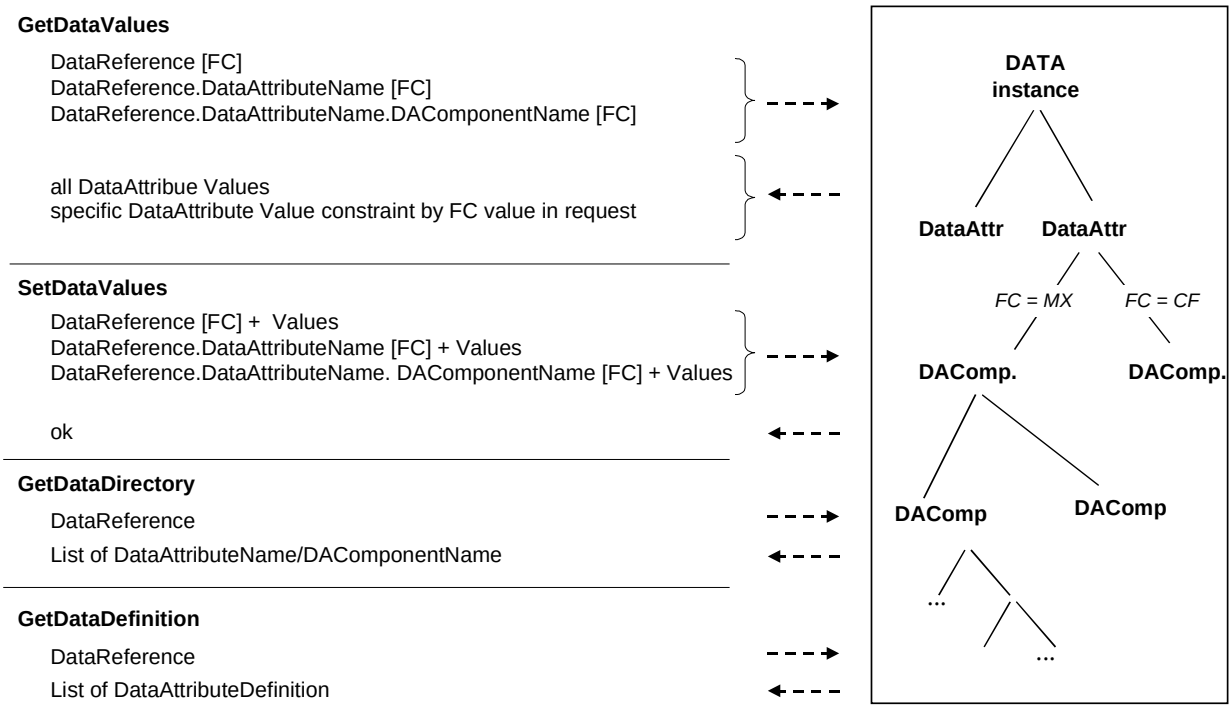


Figure 14 – Excerpt of data class services

The **GetDataValues** and **SetDataValues** services allow to access a complete **DATA** or any part of it.

10.4.2 GetDataValues

10.4.2.1 GetDataValues parameter table

A client shall use the **GetDataValues** service to retrieve values of **DataAttributes** of the referenced **DATA** made visible and thus accessible to the requesting client by the referenced **LOGICAL-NODE**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
Reference
Response+
DataAttributeValue [1..n]
Response–
ServiceError

10.4.2.2 Request

10.4.2.2.1 Reference

The parameter **Reference** shall specify the **functionally constrained data** (**FCD**) or **functionally constrained data attributes** (**FCDA**) of the **DATA** whose **DataAttribute** values are to be retrieved. The **Reference** shall be **FCD** or **FCDA**.

NOTE An SCSM may provide access to a range of ARRAY elements or a single ARRAY element.

10.4.2.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. A successful result shall return the following parameter.

10.4.2.3.1 DataAttributeValue [1..n]

The parameter **DataAttributeValue** shall contain

– the values of all **DataAttributes** of a **DATA** referenced by **FCD**; or

– the value of a **DataAttribute** referenced by **FCDA**.

NOTE The syntax of the **DataAtributeValue** is defined in an SCSM.

10.4.2.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

10.4.3 SetDataValues

10.4.3.1 SetDataValues parameter table

A client shall use the **SetDataValues** service to set values of **DataAttributes** of the referenced **DATA** made visible and thus accessible to the requesting client by the referenced **LOGICAL-NODE**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
Reference
DataAttributeValue [1..n]
Response+
Response–
ServiceError

10.4.3.2 Request

10.4.3.2.1 Reference

The parameter **Reference** shall specify the **functionally constrained data** (**FCD**) or **functionally constrained data attributes** (**FCDA**) of the **DATA** whose **DataAttribute** values are to be retrieved. The **Reference** shall be **FCD** or **FCDA**.

NOTE An SCSM may provide access to a range of ARRAY elements or a single ARRAY element.

10.4.3.2.2 DataAttributeValue [1..n]

The parameter **DataAttributeValue** shall contain

– the values of all **DataAttributes** of a **DATA** referenced by **FCD**; or

– the value of a **DataAttribute** referenced by **FCDA**.

NOTE The syntax of the **DataAtributeValue** is defined in an SCSM.

10.4.3.3 Response+

The parameter **Response+** shall indicate that the service request succeeded.

NOTE 1 For the **SetDataValues** service, a successful result means that the service request was acceptable to the server and that the server has attempted to move the value of each **DataAttribute** of the **DATA** requested by the service to the corresponding application.

NOTE 2 The action to be taken by an application receiving the value for a **DATA** to be set is outside the scope of this standard.

10.4.3.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

10.4.4 GetDataDirectory

10.4.4.1 GetDataDirectory parameter table

A client shall use the **GetDataDirectory** service to retrieve the list of all **DataAttributeNames** of the referenced **DATA** made visible and thus accessible to the requesting client by the referenced **LOGICAL-NODE**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
DataReference
Response+
DataAttributeName [1..n]
Response–
ServiceError

10.4.4.2 Request

DataReference – data reference

The parameter **DataReference** shall contain the ObjectReference of a **DATA**. The ObjectReference shall be **DataRef**.

10.4.4.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. A successful result shall return the following parameter.

DataAttributeName [1..n]

The parameter **DataAttributeName** shall contain a **DataAttrName** of the highest level of a **DataAttribute** of the **DATA**.

10.4.4.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

10.4.5 GetDataDefinition

10.4.5.1 GetDataDefinition parameter table

A client shall use the **GetDataDefinition** service to retrieve the complete list of all **DataAttribute** definitions of the referenced **DATA** made visible and thus accessible to the requesting client by the referenced **LOGICAL-NODE**.

NOTE 1 Complete means that the whole structure (the tree with all its branches and leaves) of each **DataAttribute** shall be retrieved, i.e., all nested DataAttribute.

NOTE 2 The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
DataReference
Response+
DataAttributeDefinition
Response–
ServiceError

10.4.5.2 Request

DataReference – data ObjectReference

The parameter **DataReference** shall contain the ObjectReference of the **DATA**. The ObjectReference shall be **DataRef**.

NOTE An SCSM may bundle several DataReference parameters into one message.

10.4.5.3 Response+

DataAttributeDefinition

The parameter **DataAttributeDefinition** shall contain a **DataAttrName** and **DataAttrType** of the first level and of all nested levels below of the referenced **DATA** and the functional constraints of each **DataAttribute** where applicable.

10.4.5.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

11 DATA-SET class model

11.1 General

A **DATA-SET** is an ordered group of ObjectReferences of **DATA** or **DataAttributes** (called the data set members), organized as a single collection for the convenience of the client. The membership and order of the ObjectReferences in a **DATA-SET** shall be known to both the client and the server, so that only the name of the **DATA-SET** and the current values of the referenced **DATA** or **DataAttributes** need to be transmitted. This capability thus permits more efficient use of the communications bandwidth.

NOTE 1 The membership and order of the **DATA** or **DataAttribute** in a **DATA-SET** can be retrieved with the **DataSetDirectory** service. The persistent existence of **DATA** and **DataAttribute** is expected as long as they are referenced as members of a **DATA-SET**. A system has to take special measures to ensure their persistent existence.

DATA-SETs are also important for control models, for example, reporting, logging, **GOOSE**. **DATA-SETs** are used, for example, to define the values of **DATA** or **DataAttributes** to be transmitted in case of a value change of one of its members.

DATA-SETs may be configured or created through the **CreateDataSet** service.

Any **DATA** or **DataAttributes** in a **SERVER** may be referenced by one or more **DATA-SETs**.

A **DATA-SET** may be created through the **CreateDataSet** service as a persistent or a non-persistent instance of **DATA-SET** (see Figure 15). A persistent instance of **DATA-SET** shall be visible to clients of any **TWO-PARTY-APPLICATION-ASSOCIATION**. Non-persistent instances shall be visible only to the client that created the instance. Pre-defined (configured) instances of **DATA-SET** shall be visible to clients of any **TWO-PARTY-APPLICATION-ASSOCIATION** and they shall be non-deletable.

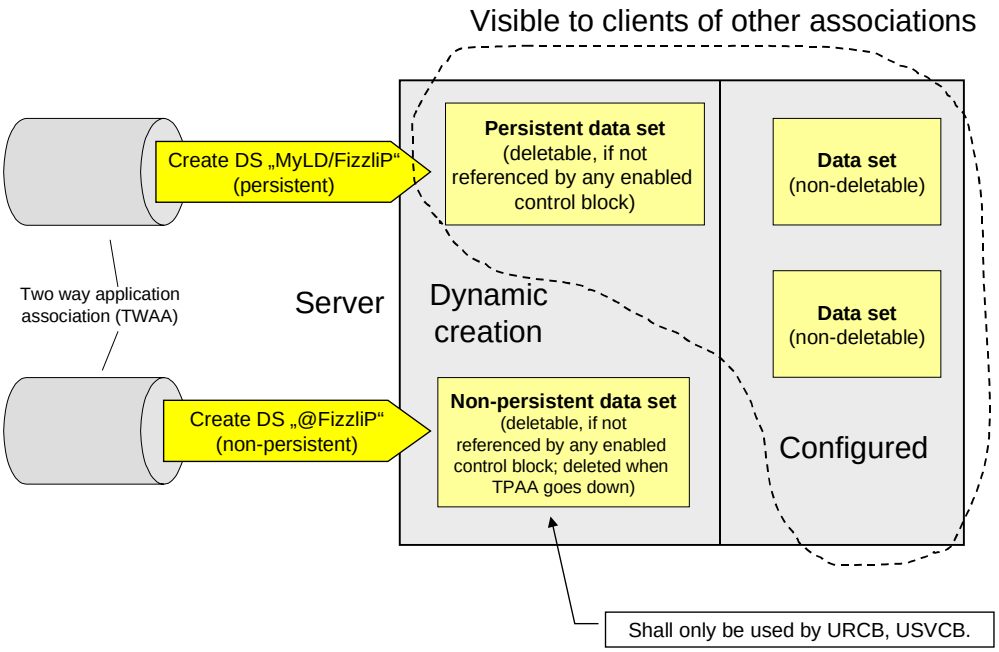


Figure 15 – Dynamic creation of data set instances

Persistent instances of **DATA-SETs** shall **not be deleted** when the **TWO-PARTY-APPLICATION-ASSOCIATION** over which the instance has been created is released or aborted. Non-persistent instances shall be **automatically deleted** when the **TWO-PARTY-APPLICATION-ASSOCIATION** over which the instance has been created is released or aborted. Persistent **DATA-SETs** created through the **CreateDataSet** service shall **not be deleted** as long as they are referenced by a control class (for example, **URCB** or **GoCB**).

A non-persistent **DATA-SET** may be accessed using the services **GetDataSetValues**, **SetDataSetValues**, and **GetDataSetDirectory**, and shall be referenced only by **URCB** and **USVCB**.

NOTE 2 Local reconfiguration of members of a **DATA-SET** may cause critical misoperations. To prevent unintended changes in the **DATA-SET** configuration, special measures have to be taken in a system (the measures are outside the scope of this part of IEC 61850).

11.2 DATA-SET class definition

11.2.1 DATA-SET class syntax

The **DATA-SET** shall have the structure as defined in Table 21.

Table 21 – DATA-SET (DS) class definition

DATA-SET class		
Attribute name	**Attribute type**	**Value/value range/explanation**
DSName	ObjectName	Instance name of an instance of DATA-SET
DSRef	ObjectReference	Path-name of an instance of DATA-SET
DSMemberRef [1..n]	(*)	(*) Functionally constrained data (**FCD**) or functionally constrained data attribute (**FCDA**)
Services GetDataSetValues SetDataSetValues CreateDataSet DeleteDataSet GetDataSetDirectory		

11.2.2 DATA-SET class attributes

11.2.2.1 DSName

The attribute **DSName** shall unambiguously identify **DATA-SET** within the scope of a **LOGICAL-NODE** or within a two-party-application-association.

11.2.2.2 DSRef

The attribute **DSRef** shall be the unique path-name of an instance of DATA-SET.

The ObjectReference **DSRef** shall be one of the following two options.

LDName/LNName.DataSetName	To reference a **persistent** instance of **DATA-SET**
@DataSetName	To reference a **non-persistent** instance of **DATA-SET**

11.2.2.3 DSMemberRef [1..n] – data set member reference

The attribute **DSMemberRef** shall specify the functionally constrained data (**FCD**) or functionally constrained data attribute (**FCDA**) of **DATA**.

The value of a member of a **DATA-SET** retrieved, set, reported, or logged shall be determined by the functionally constrained **DATA** (**FCD**) or functionally constrained data attribute (**FCDA**).

NOTE A **DATA-SET** does not contain **DATA**. A **DATA-SET** contains references, the functionally constrained data (**FCD**) or functionally constrained data attribute (**FCDA**). A **DATA-SET** may contain references to functionally constrained **DATA** (**FCD**) or functionally constrained data attribute (**FCDA**) contained in different **LOGICAL-NODE**s.

11.3 DATA-SET class services

11.3.1 Overview

For **DATA-SET** the following services are defined.

Service	Description
GetDataSetValues	Retrieve all values of **DATA** referenced by the members of the **DATA-SET**
SetDataSetValues	Write all values of **DATA** referenced by the members of the **DATA-SET**
CreateDataSet	Create a **DATA-SET** by providing the **FCD** (**FCDA**) references or that form the **DATA-SET**
DeleteDataSet	Delete a **DATA-SET**
GetDataSetDirectory	Retrieve **FCD** references of all **members** referenced in the **DATA-SET**

11.3.2 GetDataSetValues

11.3.2.1 GetDataSetValues parameter table

The client shall use the **GetDataSetValues** service to retrieve the values of all referenced **DataAttributes** made visible and thus accessible to the requesting client by the referenced **DATA-SET**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
DataSetReference
Response+
DataSetReference
DataAttributeValue [1..n]
Response–
ServiceError

11.3.2.2 Request

DataSetReference – data set ObjectReference

The parameter **DataSetReference** shall specify the ObjectReference of the **DATA-SET**. The ObjectReference **DataSetReference** shall be one of the following two options.

– **LDName/LNName.DataSetName** to reference a **persistent DATA-SET**, or

– **@DataSetName** to reference a **non-persistent DATA-SET**.

11.3.2.3 Response+

DataAttributeValue [1..n]

The parameter **DataAttributeValue** shall contain values of a member of the **DATA-SET**. The value of the **DataAttributes** of the **DATA** may be simple or complex depending on the definition of the **DATA**. For complex **DataAttrTypes** the values of all **DataAttributes** of all nesting levels shall be returned.

Each element of the list shall either contain the value of the **DataAttribute** at the time of access, or a reason for an access error.

11.3.2.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

11.3.3 SetDataSetValues

11.3.3.1 SetDataSetValues parameter table

The client shall use the **SetDataSetValues** service to set the values of all **DataAttributes** made visible and thus accessible to the requesting client by the referenced **DATA-SET**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
DataSetReference
DataAttributeValue [1..n]
Response+
Result
Response–
ServiceError

11.3.3.2 Request

11.3.3.2.1 DataSetReference – data set ObjectReference

The parameter **DataSetReference** shall specify the ObjectReference of a **DATA-SET**. The ObjectReference DataSetReference shall be one of the following two options:

– **LDName/LNName.DataSetName** to reference a **persistent DATA-SET**, or

– **@DataSetName** to reference a **non-persistent DATA-SET**.

11.3.3.2.2 DataAttributeValue [1..n]

The parameter **DataAttributeValue** shall contain a value of a member of the **DATA-Set**. The value of the **DataAttribute** of the **DATA** may be simple or complex depending on the definition of the **DATA**. For complex **DataAttrTypes** the values of all **DataAttributes** of all nesting levels shall be contained.

11.3.3.3 Response+

The parameter **Response+** shall indicate that the service request succeeded.

NOTE The action to be taken by an application receiving the values for the instances of DataAttributes to be set is outside the scope of this service definition.

A successful result shall return the following parameter.

Result

The parameter **Result** shall return a list, specified in the order of the ObjectReferences of the **DATA** that are referenced in the **DATA-SET**. This list shall indicate, for each **DATA**, either a confirmation that the service **SetDataSetValue** to the referenced instance succeeded or a reason why the service **SetDataSetValue** to the referenced **DATA** failed.

11.3.3.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

11.3.4 CreateDataSet

11.3.4.1 CreateDataSet parameter table

The client shall use the **CreateDataSet** service to request the server to create a **DATA-SET** with a list of members defined with the functionally constrained data (**FCD**) or functionally constrained data attribute (**FCDA**) made visible and thus accessible to the requesting client.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
DataSetReference
DSMemberRef [1..n]
Response+
Response–
ServiceError

11.3.4.2 Request

11.3.4.2.1 DataSetReference – data set ObjectReference

The parameter **DataSetReference** shall specify the ObjectReference of **DATA-SET** that is to be created. The ObjectReference DataSetReference shall be one of the following two options:

– **LDName/LNName.DataSetName** to create a **persistent DATA-SET**, or

– **@DataSetName** to create a **non-persistent DATA-SET**.

11.3.4.2.2 DSMemberRef [1..n] – data set member ObjectReference

The parameter **DSMemberRef** shall specify the functionally constrained data (**FCD**) or functionally constrained data attribute (**FCDA**) of a **DATA**.

11.3.4.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. If one of the referenced functionally constrained data (**FCD**) are not available to that client then the service shall fail.

11.3.4.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

11.3.5 DeleteDataSet

11.3.5.1 DeleteDataSet parameter table

The client shall use the **DeleteDataSet** service to request the server to delete a **DATA-SET** made visible and thus accessible to the requesting client.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
DataSetReference
Response+
Response–
ServiceError

11.3.5.2 Request

11.3.5.2.1 DataSetReference – data set ObjectReference

The parameter **DataSetReference** shall specify the ObjectReference of a **DATA-SET** that shall be deleted. The ObjectReference DataSetReference shall be one of the following two options.

– **LDName/LNName.DataSetName** to delete a dynamically created **persistent DATA-SET**, or

– **@DataSetName** to delete a **non-persistent DATA-SET**.

11.3.5.3 Response+

The parameter **Response+** shall indicate that the service request succeeded.

11.3.5.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

11.3.6 GetDataSetDirectory

11.3.6.1 GetDataSetDirectory parameter table

The client shall use the **GetDataSetDirectory** service to retrieve the list of the ObjectReferences of all data set members referenced by the **DATA-SET** made visible and thus accessible to the requesting client.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
DataSetReference
Response+
DSMemberRef [1..n]
Response–
ServiceError

11.3.6.2 Request

DataSetReference – data set ObjectReference

The parameter **DataSetReference** shall specify the ObjectReference of the **DATA-SET**. The ObjectReference **DataSetReference** shall be one of the following two options:

– **LDName/LNName.DataSetName** to reference a **persistent DATA-SET**, or

– **@DataSetName** to reference a **non-persistent DATA-SET**.

11.3.6.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. A successful result shall return the following parameter.

DSMemberRef [1..n] – data set member ObjectReference

The parameter **DSMemberRef** shall contain the ObjectReferences of the members of the **DATA-SET**.

NOTE The syntax of the **DSMemberRef** is defined in an SCSM.

11.3.6.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

12 Substitution model

The substitution model provides the substitution of values of **DataAttributes** whose functional constraint equals **MX** (for analogue values) or **ST** (for status values). Basically, substitution applies to **DataAttributes** with FC (= **MX** and **ST**) and to the associated quality attribute. When substitution is enabled for a specific **DataAttribute**, the **DATA** shall provide the substituted values instead of the process value to the clients.

NOTE 1 Substituting values is part of the normal operation of a substation and has nothing to do with system or device tests. Tests are performed by setting a logical device into the test mode or setting the service parameter (=Test) of the control services to TRUE.

In the typical use case for substitution, an operator on the client side enters manually a value for a **DataAttribute** located in a specific device. The client sets the **DataAttribute** to the value entered. If a client accesses the value of that **DataAttribute** (for example, using a **GetdataValue** service or subscribing to a report) the client shall receive the manual entered (substituted) value instead of the value determined by the process.

The substitution model relies on four specific **DataAttributes** defined in IEC 61850-7-3.

– **subEna** (enable substitution): The current process value shall be replaced by the value provided in the **DataAttribute subVal**.

– **subVal**, **subMag**, and **subCMag** (values for substitution of process values): The current process value shall be replaced by the value provided by the **DataAttribute subVal**, **subMag** and **subCMag** respectively.

– **subQ** (value for substitution of quality): The current process value shall be replaced by the value provided by the **DataAttribute subQ**.

– **subID** (value to indicate the initiator of the substitution).

The detailed specification of these **DataAttributes** (defined in IEC 61850-7-3) shall be followed in conjunction with this clause.

The concept of substitution is shown in Figure 16. Usually, input from the process or the result of the calculation from a function provides the value of a **DataAttribute** (in that case, the source is called “process”). In case of substitution, the value of a **DataAttribute** may be provided by an operator making use of a client. This selection of the source of the value (substitution value or process value) shall be controlled by the service **SetDataValues** (“xy.subEna” <TRUE>) to substitute or **SetDataValues** (“xy.subEna” <FALSE>) to unsubstitute. The service **SetDataValues** (“xy.subVal” <value for substitution>) shall be used to set the substituted value. There may be cases, where a local automatic function disables substitution, for example, if blocking of information exchange is disabled or communication is no longer interrupted.

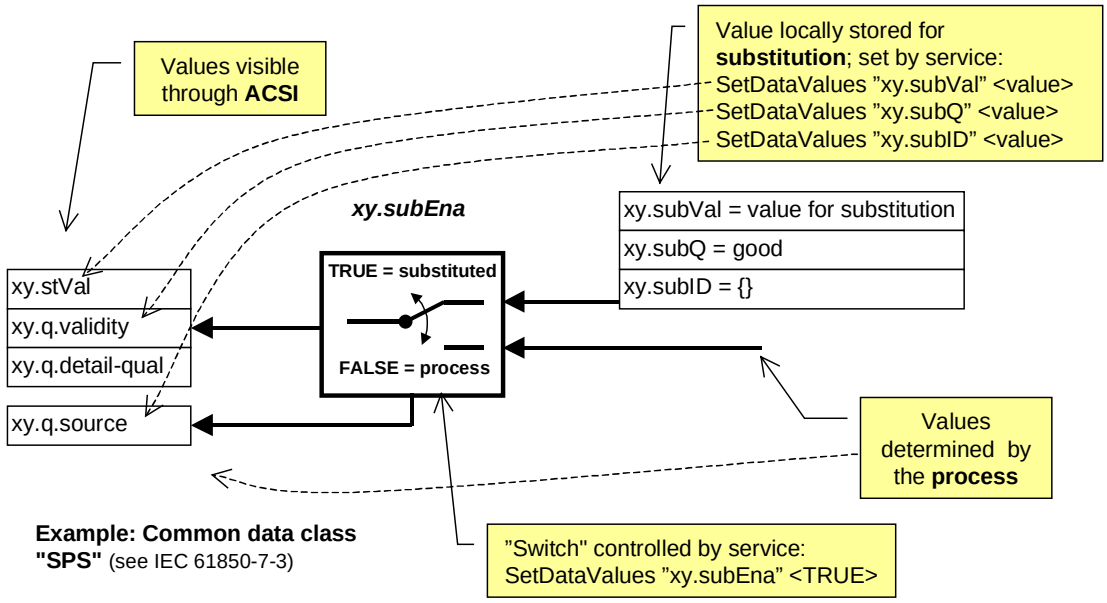


Figure 16 – Principles of substitution

The client shall set first the values to be substituted (xy.subVal, xy.subQ, xy.subID) and then enable the substitution by setting the attribute xy.subEna to TRUE.

NOTE 2 In an SCSM mapping it is recommended to use two SetDataValues services: the first to set the values used for substitution and the second to enable the substitution.

DataAttributes that provide the possibility of substitution shall have a functional constraint value of **SV** (substitutable value).

In case the association over which the substitution has been enabled fails, the substituted values shall remain unchanged. Changes shall be initiated by a service or by local means in the server device.

If the client has no direct access to the server responsible for the data acquisition (for example, in a hierarchical system, with a gateway in between, where the client needs to access a proxy), it shall be a local issue of the proxy how to handle the substitution.

13 SETTING-GROUP-CONTROL-BLOCK class model

13.1 General

An instance of a **DATA** usually has one value. The **SETTING-GROUP-CONTROL-BLOCK** (**SGCB**) model allows for an instance to have several values that can be used one at a time. The **SGCB** provides mechanisms to switch between several values of one or more **DATA**. Values that belong together build the setting group (SG).

NOTE A logical node zero (LLN0) may have one **SETTING-GROUP-CONTROL-BLOCK**. Many setting **DATA** are defined in IEC 61850-7-4.

The **SGCB** model provides services to handle different values for one or more **DATA**. The SG whose values are currently used by the **DATA** of a **LOGICAL-NODE** shall be in the state "active". The SG that can be edited shall be in the state "edit".

The **SGCB** model is depicted in the example in Figure 17. The **LOGICAL-NODE "PVOC"** (voltage controlled/dependent time overcurrent according to IEC 61850-7-4) comprises eight **DATA** for settings (LN **PDIF** has one **DATA** for settings) – (MinOpTmms, ..., RstrMode). The **SGCB** "SG Control" provides three SGs (#1, #2, and #3) each with independent values for the nine **DATA**. Each SG contains nine values (one for each of: MinOpTmms, ..., RstrMode). The members of the active SG are referenced by the ObjectReferences of the **DATA** with functional constraint SG. The members of the SG in the "edit buffer" are referenced by the ObjectReferences of the **DATA** with functional constraint **SE**.

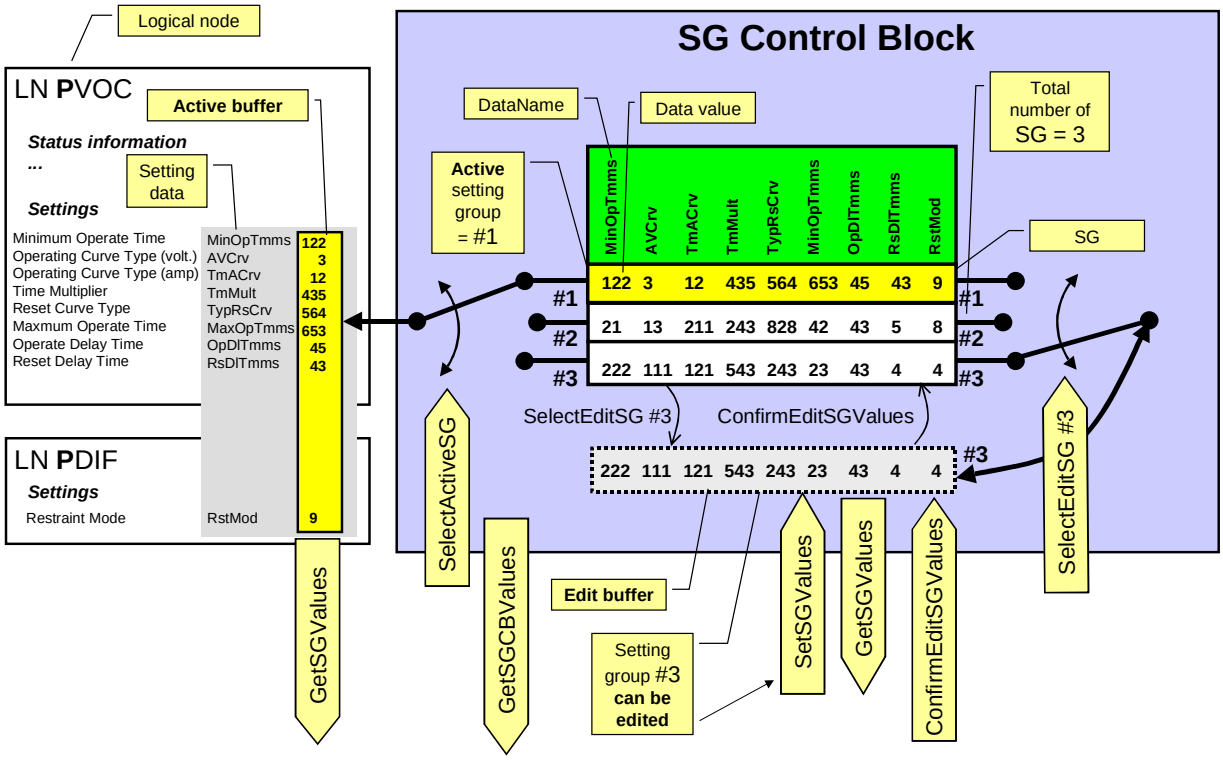


Figure 17 – Basic model of the settings model

The values of the **DATA** of the **LOGICAL-NODE PVOC** are derived from the values of one of the SGs. This is accomplished by the multiplexer on the left. The service **SelectActiveSG** determines which values (of SG #1, #2, or #3) shall be copied to the “active buffer” and be used by **PVOC**. In the example the SG #1 has been set to be in the active state.

A SG contains values for **DATA** that are contained in several **LOGICAL-NODEs**. The SGs in the example provide values for **DATA** in two **LOGICAL-NODEs** (**PDIF** and **PVOC**).

The values of SG #3 can be edited (the **SelectEditSG** switched the right multiplexer to #3); the values of this SG (now in the edit buffer) can be set and get (**SetSGValues** and **GetSGValues**). After values have been set in the edit buffer (values of SG #3), the client shall confirm that the new values (stored in the edit buffer) shall be taken over by the selected SG (SG #3).

The attributes of the **SGCB** can be retrieved (**GetSGCBValues**).

The **DATA** contained in the SG can be accessed directly with **GetSGValues**.

13.2 SGCB class definition

13.2.1 SGCB class syntax

The **SGCB** shall have the structure defined in Table 22.

Clients should use the existence of a **SGCB** to determine if the **LOGICAL-DEVICE** contains SGs.

Table 22 – SGCB class definition

SGCB class				
Attribute name	**Attribute type**	**FC**	**TrgOp**	**Value/value range/explanation**
SGCBName	ObjectName	-	-	Instance name of an instance of SGCB
SGCBRef	ObjectReference	-	-	Path-name of an instance of SGCB
NumOfSG	INT8U	SP	-	n = NumOfSG
ActSG	INT8U	SP	dchg	Allowable range: 1 … n
EditSG	INT8U	SP	dchg	Allowable range: 0 … n
LActTm	TimeStamp	SP	dchg	
Services SelectActiveSG SelectEditSG SetSGValues ConfirmEditSGValues GetSGValues GetSGCBValues				

Values of the attributes of the instances of **SGCB** shall be configured.

The setting group shall behave as shown in Figure 18.

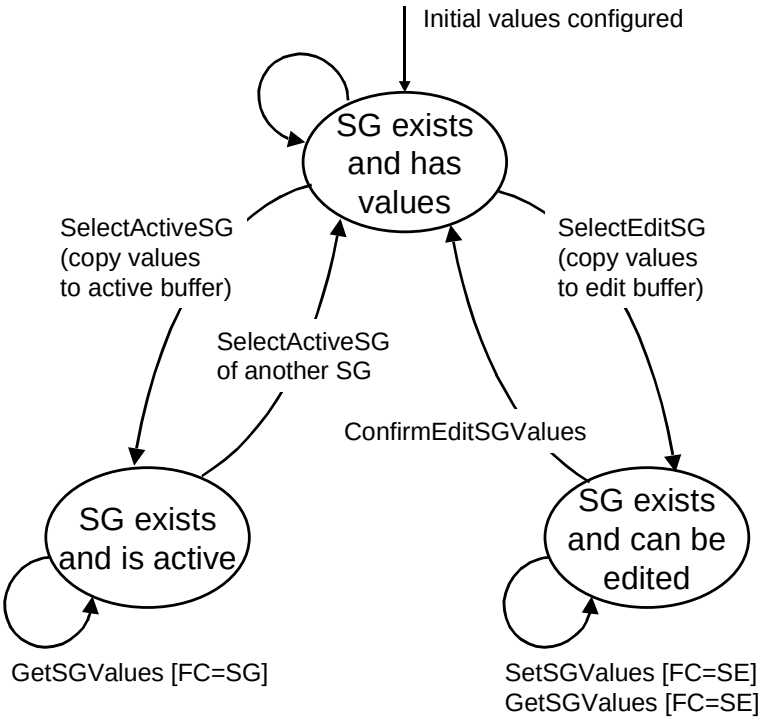


Figure 18 – Setting group state machine

The state changes shall be issued as defined with the corresponding attributes of the **SGCB** and the corresponding services of the **SGCB**.

13.2.2 SGCB class attributes

13.2.2.1 SGCBName – setting group control name

The attribute **SGCBName** shall be **SGCB** within the scope of a **LLN0**.

13.2.2.2 SGCBRef – setting group control ObjectReference

The attribute **SGCBRef** shall be the unique path-name of an **SGCB**.

The ObjectReference **SGCBRef** shall be:

LDName/LLN0.SGCB

NOTE SGCB is the standardized instance name of the **SGCB**.

13.2.2.3 NumOfSG – number of setting groups

The attribute **NumOfSG** shall identify the total number of SGs that are available in a **LOGICAL-DEVICE**.

If there are any **DataAttributes** with functional constraint **SG** in a **LOGICAL-DEVICE** then a single **SGCB** shall be present in the **LOGICAL-DEVICE**.

The attribute **NumOfSG** shall not be settable. The value of **NumOfSG** is a local matter.

13.2.2.4 ActSG – active setting group

The attribute **ActSG** shall identify the values of the SG that are in the active buffer. The attribute **ActSG** shall specify the SG whose values shall be used by the respective **LOGICAL-NODE** to performing its function. The values the **DataAttribute** of the active SG can retrieved by the service **GetSGValues**.

13.2.2.5 EditSG – edit setting group

The attribute **EditSG** shall identify the values of the SG in the edit buffer. The values of the edit buffer can be set and retrieved by the services **SetSGValues** and **GetSGValues**. The original values in the SG shall be unchanged until the client has confirmed to overwrite the values with those values stored in the edit buffer (**ConfirmEditSGValues**).

If the value of **EditSG** (= 0) then the use of services **SetSGValues** and **GetSGValues** shall cause a **Response–**.

13.2.2.6 LActTm – last activation time

The attribute **LActTm** shall identify the time when the last service **SelectActiveSG** has been processed.

13.3 SGCB class services

13.3.1 Overview

For **SGCB** the following services are defined.

Service	Description
SelectActiveSG	Select which SG shall become the active SG
SelectEditSG	Select which SG shall become the SG that can be edited after selecting
SetSGValues	Write values to the SG which has been selected for editing
ConfirmEditSGValues	Confirm that the new values to the SG which has been selected for editing become the values of the SG
GetSGValues	Read values from the SG which has been selected for editing (**FC** = **SE**) or of the active SG (**FC** = **SG**)
GetSGCBValues	Read all attribute values of the **SGCB**

13.3.2 SelectActiveSG

13.3.2.1 SelectActiveSG parameter table

A client shall use the **SelectActiveSG** service to load the values of the specified SG into the active buffer.

Parameter name
Request
SGCBReference
SettingGroupNumber
Response+
Response–
ServiceError

13.3.2.2 Request

13.3.2.2.1 SGCBReference

The parameter **SGCBReference** shall contain the ObjectReference **LDName/LLN0.SGCB**.

13.3.2.2.2 SettingGroupNumber

The parameter **SettingGroupNumber** shall specify the number ActSG of the SG (between 1 and **NumOfSG**) that shall be used to determine the new values of **DATA** of the respective **LOGICAL-NODEs**.

The values of all instances of the setting **DATA** of all **LOGICAL-NODEs** (that get their setting values from the setting group specified in the service request) shall be over-written with the new values of the data of the setting group referenced in the service request.

13.3.2.3 Response+

The parameter **Response+** shall indicate that the service request succeeded.

13.3.2.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

13.3.3 SelectEditSG

13.3.3.1 SelectEditSG parameter table

A client shall use the **SelectEditSG** service to set the **EditSG** value of the referenced **SGCB** made visible and thus accessible to the requesting client by the referenced **LLN0**.

It is the client's responsibility to check the attributes of a **SGCB** before it continues editing (confirming) the setting group in the edit buffer after an association was down. After loss of an association the **SelectEditSG** service shall be re-issued to copy the values of the selected SG to the edit buffer.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
SGCBReference
SettingGroupNumber
Response+
Response–
ServiceError

13.3.3.2 Request

13.3.3.2.1 SGCBReference

The parameter **SGCBReference** shall contain the ObjectReference of the **SGCB**.

The ObjectReference **SGCBReference** shall be:

LDName/LLN0.SGCB

13.3.3.2.2 SettingGroupNumber

The parameter **SettingGroupNumber** shall specify the number EditSG of the SG (between 1 and **NumOfSG**) that shall be used to set values (**SetSGValues**), confirm values (**ConfirmEditSGValues**), and retrieve values (**GetSGValues**) of the specified SG.

13.3.3.3 Response+

The parameter **Response+** shall indicate that the service request succeeded.

13.3.3.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

13.3.4 SetSGValues

13.3.4.1 SetSGValues parameter table

A client shall use the **SetSGValues** service to set the value of the **DATA** of the SG identified by the value of the attribute **EditSG** of the **SGCB** made visible and thus accessible to the requesting client by the referenced **LLN0**.

Setting new values shall become effective only after the client has confirmed the values by issuing the service **ConfirmEditSGValues**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
Reference
DataAttributeValue [1..n]
Response+
Response–
ServiceError

13.3.4.2 Request

13.3.4.2.1 Reference

The parameter **Reference** shall specify the **functionally constrained data** (**FCD**) or **functionally constrained data attributes** (**FCDA**) of the **DATA** whose **DataAttribute** values are to be written. The **Reference** shall be **FCD** or **FCDA**.

The **FunctionalConstraint** value of the **FCD** or **FCDA** shall be **SE**.

13.3.4.2.2 DataAttributeValue [1..n]

The parameter **DataAttributeValue** shall contain

– the values of all **DataAttributes** of a **DATA** referenced by **FCD**; or

– the value of a **DataAttribute** referenced by **FCDA**

of the SG identified by the value of the attribute **EditSG** of the **SGCB**.

NOTE The syntax of the **DataAtributeValue** is defined in an SCSM.

13.3.4.3 Response+

The parameter **Response+** shall indicate that the service request succeeded.

13.3.4.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

13.3.5 ConfirmEditSGValues

13.3.5.1 ConfirmEditSGValues parameter table

A client shall use the **ConfirmEditSGValues** service to confirm that the values of the SG (identified by the attribute **EditSG**) set with the service **SetSGValues** shall overwrite the old values of the SG of the **SGCB** made visible and thus accessible to the requesting client by the referenced **LLN0**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
SGCBReference
Response+
Result
Response–
ServiceError

13.3.5.2 Request

SGCBReference

The parameter **SGCBReference** shall contain the ObjectReference **LDName/LLN0.SGCB**.

13.3.5.3 Response+

The parameter **Response+** shall indicate that the service request succeeded.

13.3.5.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

13.3.6 GetSGValues

13.3.6.1 GetSGValues parameter table

A client shall use the **GetSGValues** service to retrieve the values of **DATA** of SGs made visible and thus accessible to the requesting client by the referenced **LLN0**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
Reference
Response+
DataAttributeValue [1…]
Response–
ServiceError

13.3.6.2 Request

Reference

The parameter **Reference** shall specify the **functionally constrained data** (**FCD**) or **functionally constrained data attributes** (**FCDA**) of the **DATA** whose **DataAttribute** values are to be retrieved. The **Reference** shall be **FCD** or **FCDA**.

The **FC** value of the **FCD** or **FCDA** shall be

– **SE** to retrieve the values of the SG in the edit buffer; and

– **SG** to retrieve the values of the active SG.

13.3.6.3 Response+

DataAttributeValue [1..n]

The parameter **DataAttributeValue** shall contain

– the values of all **DataAttributes** of a **DATA** referenced by **FCD**; or

– the value of a **DataAttribute** referenced by **FCDA**.

The **FC** value of the **FCD** or **FCDA** shall be **SE** or **SG** respectively.

NOTE The syntax of the **DataAtributeValue** is defined in an SCSM.

13.3.6.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

13.3.7 GetSGCBValues

13.3.7.1 GetSGCBValues parameter table

A client shall use the **GetSGCBValues** service to retrieve the list of attribute values of the referenced **SGCB** made visible and thus accessible to the requesting client by the referenced **LLN0**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
SGCBReference
FunctionalConstraint
Response+
NumberOfSettingGroup
ActiveSettingGroup
EditSettingGroup
LastActivateTime
Response–
ServiceError

13.3.7.2 Request

13.3.7.2.1 SGCBReference

The parameter **SGCBReference** shall contain the ObjectReference **LDName/LLN0.SGCB**.

13.3.7.2.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to identify the functional constraint of the respective attribute of **SGCB** contained in the LLN0. The value shall be **SP**.

13.3.7.3 Response+

13.3.7.3.1 NumberOfSettingGroup – number of setting group controls

The parameter **NumberOfSettingGroup** shall specify the total number of the SG of the attribute **NumOfSG** of the referenced **SGCB**.

13.3.7.3.2 ActiveSettingGroup – active setting group

The parameter **ActiveSettingGroup** shall specify the number of the SGs of the attribute **ActiveSG** from which the current active SG values shall be derived.

13.3.7.3.3 EditSettingGroup – edit setting group

The parameter **EditSettingGroup** shall specify the number of the SG of the attribute **EditSG** whose values can be set and retrieved.

13.3.7.3.4 LastActivateTime – last time of activation of a setting group

The parameter **LastActivateTime** shall specify the time of the last activation of the attribute **LActTm**.

13.3.7.4 Response–

The **Response–** parameter shall indicate that the service request failed. The appropriate ServiceError shall be returned.

14 REPORT-CONTROL-BLOCK and LOG-CONTROL-BLOCK class models

14.1 Overview

Reporting and logging meets a number of crucial requirements for event-driven information exchange. The data transfer models described in this clause provide mechanisms for transferring data values caused by well-defined conditions from a logical node to one client or storing the data in a server's log for future querying.

In contrast to high bandwidth and time-consuming fast reading (polling) devices for extraordinary event occurrences, the reporting provides immediate transmission of events. Reporting is controlled by constraints.

The main characteristics of reporting and logging are:

– timely reports serve as an indication to clients under real-time constraints (optionally keeping sequence-of-events to the client),

– logging of events for later retrieval (sequence-of-events stored in server),

– the impact on network bandwidth is minimized,

– sending reports only when required (controlled by several attributes),

– low-frequency integrity scan and client-initiated general interrogation.

Reporting provides mechanisms to report packed values of instances of DATA immediately or after some buffer time. The logging model provides mechanisms to store events in the log in sequence. A client may query a range of log entries at any time.

Reporting and logging as well as the basic services of the data model provide flexible data retrieval schemas, for example:

– change-of-state notification of clients: immediate reports,

– sequence-of-events: keeping reports in sequence or storing and querying sequences of log entries,

– polling data at any time: GetDataValues and GetDataSetValues

NOTE 1 Subclause 14.3.5.3.4 provides special services for event distribution (generic substation event model, GSE). Reporting and GSE have totally different qualities of services and behaviour. Reporting is connection-oriented (GSE uses multicast), reporting transmits data once (GSE transmits and retransmits data with heartbeat). IEC 61850-7-1 provides a comparison of the models.

NOTE 2 Clause 16 specifies special services for communication of measured values of, for example, voltage transformer (VT) and current transformer (CT) under crucial time constraints.

The principle building blocks and services for reporting and logging are depicted in Figure 19.

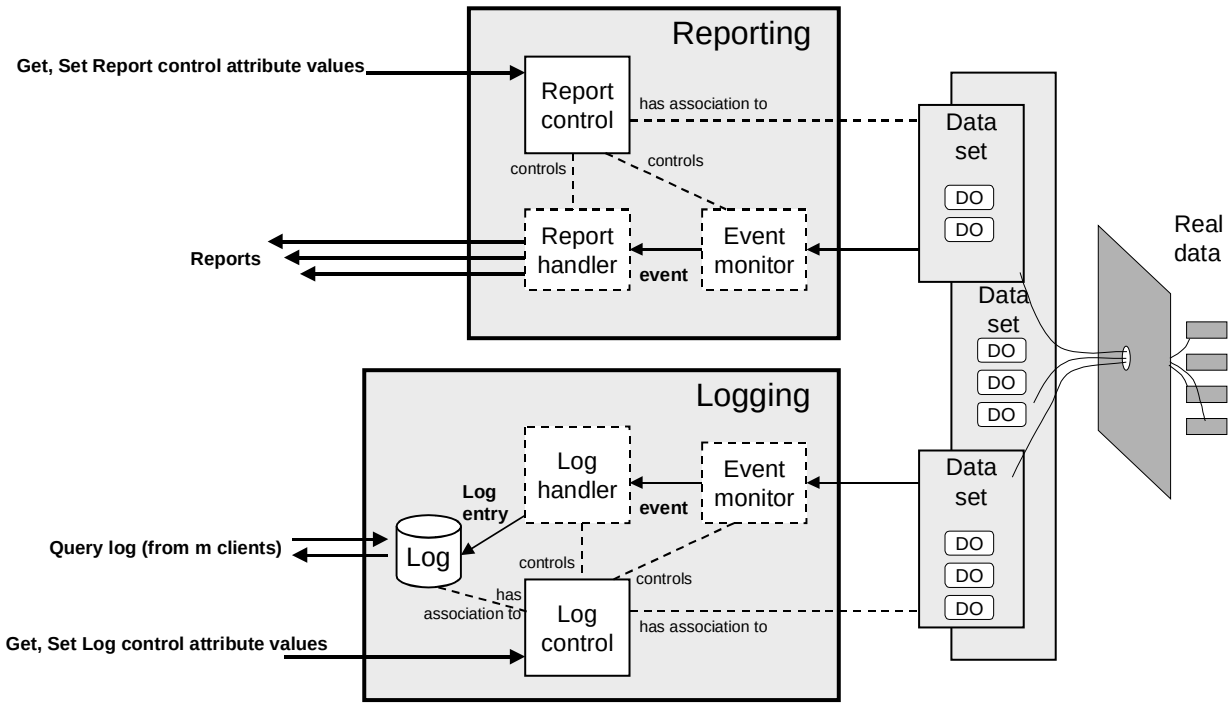


Figure 19 – Basic building blocks for reporting and logging

The reporting model is composed of three building blocks. The logging model has four building blocks. Classes are defined for the report control, the log control and the log.

NOTE 3 The handler and monitor are introduced here for conceptual reasons only.

The **DATA-SET** (referencing **DATA**) represent the real data values. The real data values are conceptually monitored by the event monitors. An event monitor determines, on the basis of the state of the real data and the attributes of the control class, when to inform the handler of the occurrence of an internal event. The report handler decides when and how to send a report to the subscribed client. The log handler stores a log entry to the log.

A filter mechanism reduces the amount of data values to be reported or stored in a log. Instead of sending any change of status or measured values, the server can be pre-configured or configured remotely by activating report control instances (subscription) to report (publish) only the changed data values since the last report or to send all data values of an application specific set of data when certain conditions are met (for example, data-change or cyclic). The report control continuously reports data values without further client actions. A client may remotely disable the issuance of further reports to this client.

Additionally, a client may initiate a general interrogation at any time to receive all data values of an application specific set of data.

NOTE 4 Using this mechanism, clients can synchronize their databases with the current status of a logical node.

The QueryLog service provides retrieval of a set of selective log entries. Selection criteria are the time range or the range of entryIDs.

14.2 REPORT-CONTROL-BLOCK class model

14.2.1 Basic concepts

The **REPORT-CONTROL-BLOCK** shall control the procedures that are required for reporting values of **DATA** from one or more **LOGICAL-NODEs** to one client. Instances of report control shall be configured in the server at configuration time.

A server shall restrict access to an instance of a report control to one client at a time. That client exclusively shall “own” that instance and shall receive reports from that instance of report control.

There are two classes of report control defined, each with a slightly different behaviour.

– **BUFFERED-REPORT-CONTROL-BLOCK** (**BRCB**) – internal events (caused by trigger options data-change, quality-change, and data-update) issue immediate sending of reports or buffer the events (to some practical limit) for transmission, such that values of **DATA** are not lost due to transport flow control constraints or loss of connection. **BRCB** provides the sequence-of-events (**SOE**) functionality.

– **UNBUFFERED-REPORT-CONTROL-BLOCK** (**URCB**) – internal events (caused by trigger options data-change, quality-change, and data-update) issue immediate sending of reports on a “best efforts” basis. If no association exists, or if the transport data flow is not fast enough to support it, events may be lost. Or

To allow multiple clients to receive the same values of **DATA**, multiple instances of the report control classes shall be made available.

For the **unbuffered report control**, this is achieved as follows.

a) A server shall make multiple instances of a report control class available with all instances visible to all clients. Each instance name shall be distinct by appending a technological index (1..n). Clients may be configured to directly use a specific instance, or may browse the instances for one that is currently disabled and hence available. When an **UNBUFFERED-REPORT-CONTROL-BLOCK** is reserved by a client, all other clients shall have no set access to its parameters.

b) A server shall permit a client to view only one instance of the report control class based on either the client’s connection or on its authentication’s view of the IED’s data model. The instance name shall be the same for each client and the server shall manage the separation of the instances. The number of concurrent clients that can use these instances may be limited by the resources of the server.

The above two approaches are equivalent from the point of view of a client wishing to use reporting services: the client uses the first report control which it can enable.

For the **buffered report controls**, this is achieved as follows.

The **buffered report controls** shall be configured. These report controls are usually intended to be used by a client implementing a well-defined functionality, for example, a SCADA master. The client may know the ObjectReference of the **BRCB** by configuration or by the use of a naming convention. The visibility of these instances may be a subject of access control. When a **BRCB** is enabled, all other clients have no access right to set its parameters. A **BRCB** shall be disabled by explicit request by the client.

14.2.2 BUFFERED-REPORT-CONTROL-BLOCK (BRCB) class definition

14.2.2.1 BRCB class Syntax

The **BRCB** class shall have the structure defined in Table 23.

Table 23 – BRCB class definition

BRCB class				
Attribute name	**Attribute type**	**FC**	**TrgOp**	**Value/value range/explanation**
BRCBName	ObjectName	-	-	Instance name of an instance of BRCB
BRCBRef	ObjectReference	-	-	Path-name of an instance of BRCB
Specific to report handler				
RptID	VISIBLE STRING65	BR	-	
RptEna	BOOLEAN	BR	dchg	
DatSet	ObjectReference	BR	dchg	
ConfRev	INT32U	BR	dchg	
OptFlds	PACKED LIST	BR	dchg	
sequence-number	BOOLEAN			
report-time-stamp	BOOLEAN			
reason-for-inclusion	BOOLEAN			
data-set-name	BOOLEAN			
data-reference	BOOLEAN			
buffer-overflow	BOOLEAN			
entryID	BOOLEAN			
BufTm	INT32U	BR	dchg	
SeqNum	INT8U	BR	-	
TrgOpEna	TriggerConditions	BR	dchg	
IntgPd	INT32U	BR	dchg	0.. MAX; 0 implies no integrity report.
GI	BOOLEAN	BR	-	
PurgeBuf	BOOLEAN	BR	-	
EntryID	EntryID	BR	-	
Services Report GetBRCBValues SetBRCBValues				

These attributes determine the service procedures of the **Report** service. The impact of the various values shall be as defined in the following attribute definitions.

14.2.2.2 BRCBName – buffered report control name

The attribute **BRCBName** shall be the name of the **BRCB** that unambiguously identifies the **BRCB** within an **LOGICAL-NODE**.

14.2.2.3 BRCBRef – buffered report control ObjectReference

The attribute **BRCBRef** shall be the unique path-name of a **BRCB**.

The ObjectReference BRCBRef shall be:

LDName/LNName. BRCBName

14.2.2.4 RptID – report identifier

The attribute **RptID** shall be the client-specified report identifier of the **BRCB** that has caused the generation of the report. If the report identifier value of the **BRCB** is NULL, then the instance name (the whole path-name) of the **BRCB** shall be reported as the report identifier.

NOTE The report identifier field may be used by clients to distinguish between reports from various **BRCBs**. This value is mirrored by the server.

14.2.2.5 RptEna – report enable

The attribute **RptEna** shall be used to control and indicate the current state of the **BRCB**. The state machine for the attribute RptEna shall be as depicted in Figure 20.

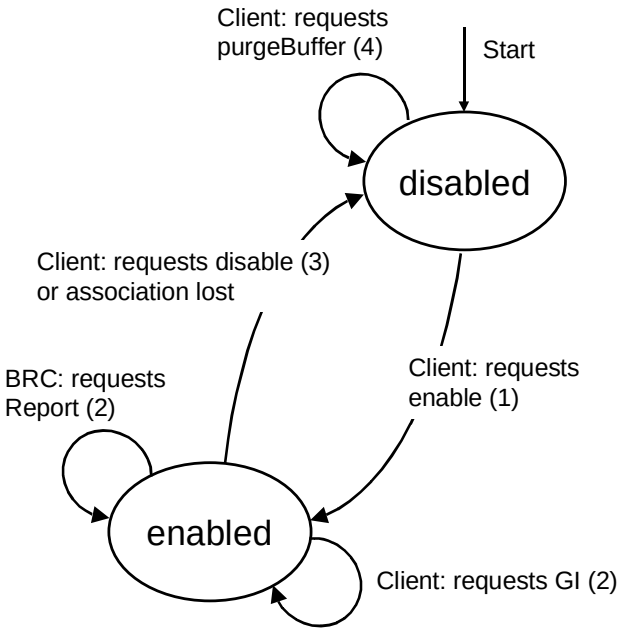


Figure 20 – BRCB state machine

disabled: the **BRCB** is available. No reports shall be issued.

The **BRCB** shall monitor the values of the **DataAttribute** referenced by the **DATA-SET**. Internal events as result of the trigger conditions data-change (dchg), quality-change (qchg), and data-update (dupd) shall be buffered (up to a practical limit).

The client shall configure the **BRCB** and shall then set this attribute to **enabled** (see (1) in Figure 20).

enabled: the **BRCB** shall generate reports for the buffered events and new events as specified in the **BRCB**.

The client shall set the attribute to **enabled**, when he reserves the report for exclusive use for this client (see (1) in Figure 20). If the association with that client is lost, the **BRCB** shall continue to buffer internal events. After a client has established a new association, he shall set the attribute to **enabled** following that, the **BRCB** shall continue sending the reports stored in the buffer over the association over which the attribute has been set to enabled.

NOTE The client who has enabled the **BRCB** receives the stored reports. A restricted view to the **BRCB** to one client guarantees that only the same client as before can receive the buffered reports.

To release the **BRCB** the client shall set this attribute to **disabled**.

While in the state **enabled** no changes of attribute values of the **BRCB** shall be allowed except disabling and activating general-integration.

The attributes of a **BRCB** in state enabled shall be read only to the clients of all other associations.

14.2.2.6 DatSet – Data set reference

The attribute **DatSet** shall specify the ObjectReference of the **DATA-SET** being monitored and whose values of the members of the **DATA-SET** (one, a subset, or all) shall be reported.

The **DatSet** shall be included in the report if data-set-name in **OptFlds** of the **BRCB** is set to TRUE otherwise it shall be omitted in the report.

A change of the value of the attribute **DatSet** shall have the same effect as setting **purgeBuf** to TRUE.

14.2.2.7 ConfRev – configuration revision

The attribute **ConfRev** shall represent a count of the number of times that the configuration of the **DATA-SET** referenced by **DatSet** has been changed. Changes that shall be counted are:

– any deletion of a member of the **DATA-SET**; and

– the reordering of members of the **DATA-SET**.

The counter shall be incremented when the configuration changes.

The initial value for **ConfRev** is outside the scope of this part of IEC 61850. The value of 0 shall be reserved. A restart of the IED shall not reset the value.

NOTE Configuration changes of **DATA-SETs** due to processing of services are not allowed (see **DATA-SET** model). Changes to be taken into account for the **ConfRev** are those made by local means like system configuration.

14.2.2.8 OptFlds – optional fields to include in report

The attribute **OptFlds** shall be the client-specified optional fields to be included in the report issued by this **BRCB**. This attribute defines a subset of the optional header fields of the report (see 14.2.3.2.2.1) that shall be included in the report:

– sequence-number (if TRUE SeqNum shall be included in the report);

– report-time-stamp (if TRUE TimeOfEntry shall be included in the report);

– reason-for-inclusion (if TRUE ReasonCode shall be included in the report);

– data-set-name (if TRUE DatSet shall be included in the report);

– data-reference (if TRUE DataRef or DataAttributeReference shall be included in the report);

– buffer-overflow (if TRUE BufOvfl shall be included in the report);

– entryID (if TRUE EntryID shall be included in the report).

If a **BRCB** does not support one of the above options, then an attempt to set the corresponding bit to TRUE shall cause a negative response of the **SetBRCBValues** service.

14.2.2.9 BufTm – buffer time

The attribute **BufTm** (see Figure 21) shall specify the time interval in milliseconds for the buffering of internal notifications caused by data-change (dchg), quality-change (qchg), data-update (dupd) by the **BRCB** for inclusion into a single report.

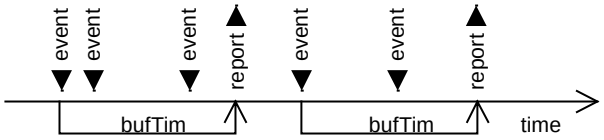


Figure 21 – Buffer time

Upon receipt of the first internal event notification of the referenced **DATA-SET**, the **BRCB** shall start a timer of the duration buffer time. When the timer expires, the **BRCB** shall combine all internal events that have been received during the time interval into a single report. The next internal event following the timer expiration shall signal the new start of that timer. The default value of 0 shall be reserved to indicate that the buffer time attribute is not to be used by the **BRCB**. Each internal event shall cause the **BRCB** to send a single report. The value shall be settable in 1 ms increments and shall be able to convey up to 1 h of buffer time.

NOTE 1 The standard does not require a specific implementation of the monitoring function in a server. The mechanism of how to monitor the application data is outside the scope of this part of IEC 61850. An internal event is understood as an abstract internal indication that, for example, a specific status value has been changed.

In the case where a second internal notification of the same member of a **DATA-SET** has occurred prior to the expiration of **BufTm**, the **BRCB**

– shall for status information behave as if **BufTm** has expired and immediately send the report, restart the timer with value **BufTm** and process the second notification; or

– may for analogue information behave as if **BufTm** has expired and immediately transmit the report for transmission, restart the timer with value **BufTm** and process the second notification; or

– may for analogue information substitute the current value in the pending report with the new one.

NOTE 2 A second status change of the same member will cause the immediate queuing of the report for transmission; a second analogue change may not. Changes of the same member are communicated in consecutive reports. No reports will be lost because the BRCB buffers them.

If a **BRCB** does not support buffer time then an attempt to set the **BufTm** attribute to a value greater than zero shall cause a negative response of the **SetReportControlValues** service.

14.2.2.10 SeqNum – sequence number

The attribute **SeqNum** shall specify the sequence number for each **BRCB** that has report enable set to TRUE. This number is to be incremented by the **BRCB** for each report generated and sent. The increment shall occur once the **BRCB** has formatted the report and queued the report to the N-1 protocol layer. The first report following the setting of the report enable to TRUE shall contain sequence number 0.

14.2.2.11 TrgOpEna – trigger options enabled

The attribute **TrgOpEna** shall specify the trigger conditions which shall be monitored by this BRCB. The following values are defined:

– data-change (dchg)

– quality-change (qchg)

– data-update (dupd)

– integrity

– general-interrogation

The trigger options dchg, qchg, and dupd refer to the attribute trigger option (TrgOp) of the **DataAttribute** of the common **DATA** classes in IEC 61850-7-3. Integrity (general-interrogation) shall be a trigger defined by the attribute **IntgPd** (**GI**).

Details related to the generation of a report based on the different trigger options shall be as specified in 14.2.3.2.3.

If a **BRCB** does not support one or more of the trigger options, the attempt to set the **TrgOpEna** attribute to TRUE for one of these not supported values, shall cause a negative response of the SetReportControlValues service.

14.2.2.12 IntgPd – integrity period

If **TrgOpEna** is set to integrity, the attribute **IntgPd** shall indicate the period in milliseconds used for generating an integrity report. An integrity report shall report the values of **all** members of the related **DATA-SET**. **BufTm** shall have no effect when this change issues a report.

If a **BRCB** does not support integrity period then an attempt to set the **IntgPd** attribute to a value greater than 0 shall cause a negative response of the **SetReportControlValues** service.

A value of 0 shall indicate that no integrity reports shall be issued.

NOTE An integrity scan may transmit the same values as a general interrogation. The integrity scan is issued by the server. The general-interrogation is issued by the client.

14.2.2.13 GI – general-interrogation

The attribute **GI** shall indicate the request to start the general-interrogation process. After setting to TRUE, the **BRCB** shall start the general-interrogation process. After initiation of the general interrogation, this attribute shall be automatically set to FALSE by the **BRCB**.

If a **BRCB** does not support general-interrogation then an attempt to set the **GI** attribute to TRUE shall cause a negative response of the **SetReportControlValues** service.

14.2.2.14 PurgeBuf – purge buffer

The attribute **PurgeBuf** shall indicate the request to discard buffered events. After setting to TRUE, the **BRCB** shall discard all buffered events that have not yet been sent to the client. After discarding the buffered events, this attribute shall be automatically set to FALSE by the **BRCB**.

14.2.2.15 EntryID – entry identifier

The attribute **EntryID** shall represent an arbitrary OCTET STRING used to identify an entry in a sequence of events of a buffered report. The value of the **EntryID** shall be used by the **BRCB** to start sending the next report which follows the **EntryID** value set in the **BRCB**.

NOTE This allows a client to set the **EntryID** to the last value of the **EntryID** received with the last proper report in order to synchronize with the server. Setting the **EntryID** may also be used to acknowledge the reception of reports (or to resend reports).

After an association (which was down) has been re-established by a client, the client shall set the **EntryID** to the value received last. The **BRCB** shall continue sending reports with the next value of **EntryID** after enabling the **BRCB** to receive the reports buffered.

If the client has not set the **EntryID** to a specific value when the **BRCB** is enabled, the **BRCB** shall use the first value available.

14.2.3 BRCB class services

14.2.3.1 Overview

For **BRCB** the following services are defined:

Service	Description
Report	Send a report
GetBRCBValues	Read an attribute of a **BRCB**
SetBRCBValues	Write an attribute of a **BRCB**

14.2.3.2 Report

14.2.3.2.1 Report parameter table

The report service shall be used by **BRCB** to send reports from the server to the client.

Parameter name
Request
ReportFormat

NOTE The Report service is an unconfirmed service. It consists only of a request service primitive. The **DATA-SET** values are sent from the server to the client. In a SCSM this service may be confirmed at, for example, the transport layer.

14.2.3.2.2 Request

14.2.3.2.2.1 ReportFormat Syntax

The parameter **ReportFormat** shall specify the information to be included in the report. The structure of the report shall be as specified in Table 24.

Table 24 – Report format specification

ReportFormat		
Parameter name	**Parameter type**	**Explanation**
RptID	VISIBLE STRING65 [a]	Report identification
OptFlds	[a]	Optional fields to be included in the report
IF sequence-number = TRUE in optFlds		
SeqNum	INT16U	Sequence number
SubSeqNum	INT16U	Subsequence number
MoreSegmentsFollow	BOOLEAN	More report segments with the same sequence number follow
IF dat-set-name = TRUE in optFlds		
DatSet	ObjectReference [a]	Data set reference
IF buffer-overflow = TRUE in optFlds		
BufOvfl	BOOLEAN	Buffer overflow has occurred
Entry		
IF report-time-stamp = TRUE in optFlds		
TimeOfEntry	EntryTime	
IF entryID = TRUE in optFlds		
EntryID	EntryID	
EntryData [1..n]		
IF data-reference = TRUE in optFlds		
DataRef	ObjectReference	Respective DataAttrRef
Value	[a]	Type(s) depend on the definition of common data classes in IEC 61850-7-3
ReasonCode	TriggerConditions	If reason-for-inclusion (= TRUE) in optFlds

[a] The type and value of this parameter shall be derived from the attribute OptFlds of the respective **BRCB**.

14.2.3.2.2.2 RptID – report ID

The parameter **RptID** shall be derived from the respective attribute in the **BRCB**.

14.2.3.2.2.3 OptFlds – optional fields to include in report

The parameter **OptFlds** shall specify which of the optional fields (**sequence-number**, **report-time-stamp**, **reason-for-inclusion**, **data-set-name**, **data-reference**, **buffer-overflow**, or **entryID**) are included in the Report.

The parameter **OptFlds** shall be derived from the attribute **OptFlds** of the respective **BRCB**.

14.2.3.2.2.4 SeqNum – sequence number

The **BRCB** that has report enable set to TRUE shall maintain the parameter **SeqNum**. This number shall be incremented by the **BRCB** for each report generated and sent on the basis of the **BRCB**. The increment shall occur once the **BRCB** has formatted the report for

transmission. The first report following the setting of the report enable to TRUE shall contain sequence number 0. The sequence number shall roll over to 0 at its maximal value.

The sequence number shall be included in the report if the optional fields to include in report attribute (**OptFlds**) of the **BRCB** includes the sequence-number (=TRUE); otherwise, it shall be omitted. Figure 22 gives an example of report generation and sequence number.

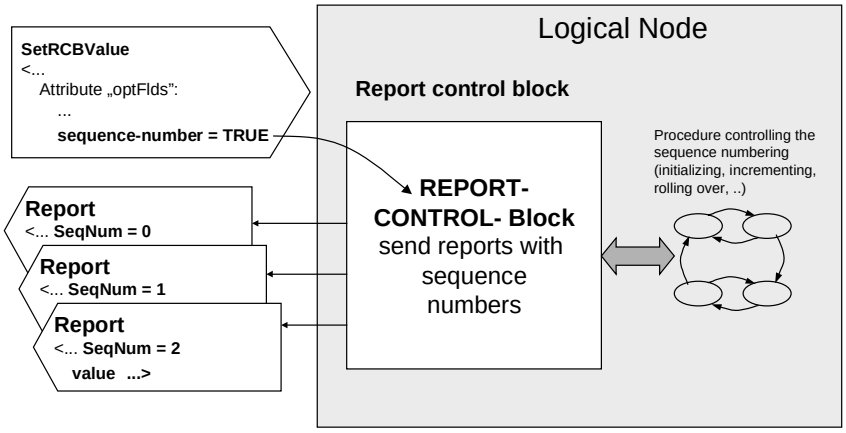


Figure 22 – Report example on the use of sequence number

14.2.3.2.2.5 SubSeqNum – subsequence number

For the case of long reports that do not fit into one message, a single report shall be divided into subreports. Each segment – of one report – shall be numbered with the same sequence number and a unique **SubSeqNum**.

The **BRCB** shall maintain a subsequence number for each report. This number shall be incremented for each subreport generated and sent based upon the report control instance. The increment shall occur once the server has formatted the subreports and queued the sub-report to the next lower protocol layer. The first subreport of the report shall have a subsequence number of zero. The subsequence number shall roll over to 0 after all subreports of one specific report have been queued.

The subsequence number shall be included in the report if the optional fields to include in report attribute (OptFlds) of the **BRCB** includes sequence-number (=TRUE); otherwise, it shall be omitted.

If a **BRCB** does not support sequence numbering then an attempt to set the sequence-number of the **OptFlds** attribute to TRUE shall cause a negative response of the **SetBRCBValues** service.

14.2.3.2.2.6 MoreSegmentsFollow – more report segments follow

The parameter **MoreSegmentsFollow** indicates that more report segments with the same sequence number follow.

14.2.3.2.2.7 DatSet – data set reference

The parameter **DatSet** shall be derived from the respective attribute in the **BRCB**.

14.2.3.2.2.8 BufOvl – buffer overflow occurred

The parameter **BufOvl** shall indicate to the client that a buffer overflow occurred. The **BRCB** shall set this field in the first report that is sent with events that occurred after the overflow.

14.2.3.2.2.9 Entry

TimeOfEntry – report time stamp

The parameter **TimeOfEntry** shall specify the time at which the report was generated. The **TimeOfEntry** shall be included in the report if the optional fields to include attribute (**OptFlds**) of the **BRCB** includes report-time-stamp (=TRUE), otherwise it shall be omitted.

NOTE The event “time at which the report was generated” is determined by a specific implementation.

If the **BRCB** does not support **TimeOfEntry** then an attempt to set the report-time-stamp of the **OptFlds** attribute to TRUE shall cause a negative response of the **SetBRCBValues** service.

Reports with the same sequence number but different subsequence numbers shall use the same **TimeOfEntry**.

EntryID – entry identifier

The parameter **EntryID** shall represent an arbitrary OCTET STRING used to identify an entry in a sequence of events of a **BRCB**.

EntryData [1..n]

The parameter **EntryData** shall contain the data reference, value, and reasonCode of each member of the **DATA-SET** to be included in the report. The value shall comprise the value of all data attributes of the member of **DATA-SET**.

DataRef

The parameter **DataRef** shall contain the functional constrained data (**FCD**) of the **DataAttribute** values included in the report.

Value

The parameter **Value** shall contain the **DataAttribute** values included in the report.

The number of members of the **DATA-SET** whose values shall be included in the report shall depend on the control attribute buffer time (**BufTm**) and the occurrences of internal events.

BufTm = 0

In case of (BufTm = 0) only the value of the member of a **DATA-SET** shall be included that produced the internal event.

EXAMPLE The data attribute stVal of the **DATA** MyLD/XCBR1.Pos (Position) in Figure 23 is referenced in two different **DATA-SETs**. The figure displays two different instances that reference the data attributes of the position. In the left case the **DATA-SET** references 9 individual **DATA-SET** members (all of functional constraint ST): Pos.stVal is one of the nine members. In case of the change produced by the **member stVal**, the value for exactly that member will be included in the report. The **DATA-SET** in the right example has just two members. The **DATA** Pos (which has six data attributes: stVal, q, t, ...) is one of the two members. A change produced in the **member Pos** (for example, by the change in the **DataAttribute** stVal) causes the inclusion of the values of all **DataAttribute** of the **DATA-SET** member Pos (i.e., the complete member comprising all six **DataAttributes** stVal, q, t, ...).

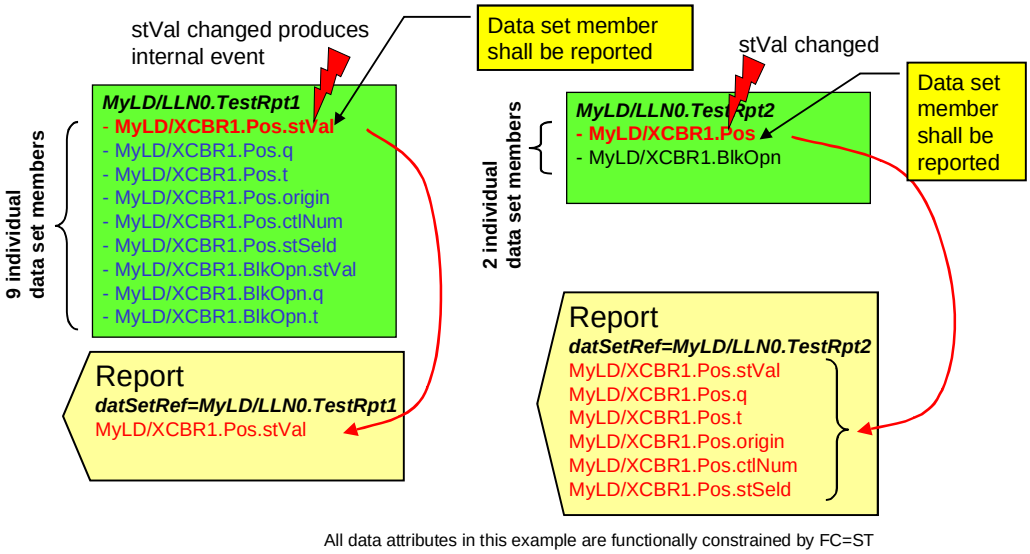


Figure 23 – Data set members and reporting

BufTm > 0

In the case of (**BufTm** > 0) the values of all members of a **DATA-SET** shall be included that produced an internal event during the buffer time. Further constraints apply; see 14.2.2.7 for additional details on **BufTm**.

ReasonCode – reason for inclusion

The reason for inclusion shall be included in the report if the optional fields to include in report attribute (OptFlds) of the **BRCB** includes reason for inclusion (=TRUE); otherwise, it shall be omitted. The value for the reason for inclusion shall be set according to the **TrgOp** that caused the creation of the report. The value range for reasons for inclusion shall be as listed:

– data-change (caused by TrgOp = dchg in an instance of **DATA**);

– quality-change (caused by TrgOp = qchg in an instance of **DATA**);

– data-update (caused by TrgOp = dupd in an instance of **DATA**);

– integrity (caused by the attribute **IntgPd** in the **BRCB**);

– general-interrogation (caused by setting the attribute **GI** of the **BRCB** to TRUE by a client).

14.2.3.2.3 Procedures for report generation

14.2.3.2.3.1 Overview

Figure 24 shows the principle relation between a **BRCB** and the processing of the report. The information that is to be included in the report and how it is to be included depends on the attribute settings of the **BRCB**.

NOTE Not all attributes and not all details are shown in Figure 24.

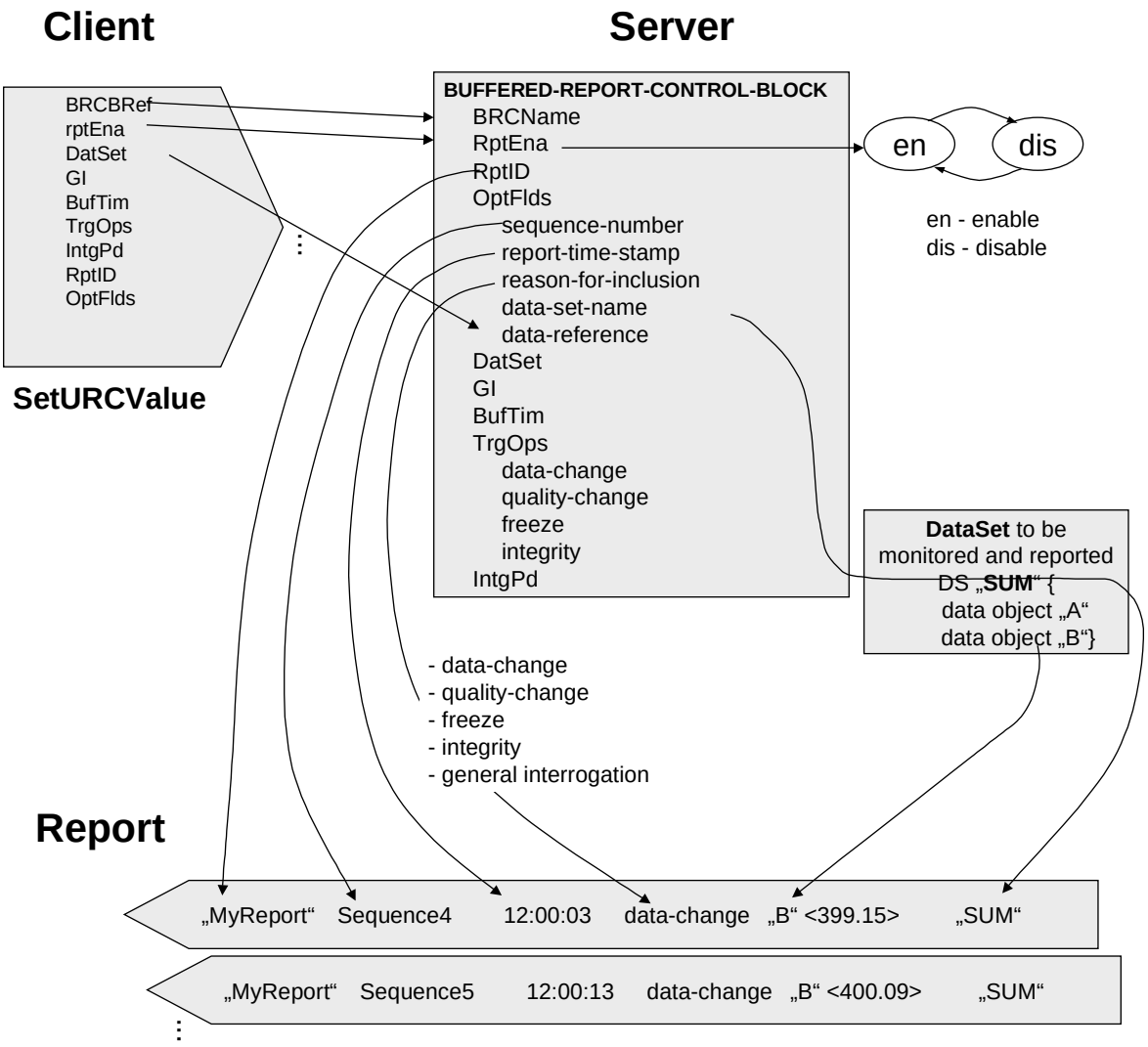


Figure 24 – Report example

Pre-condition

A **BRCB** shall have been configured and enabled for reporting and shall have an established association with the client to which the information is to be reported.

14.2.3.2.3.2 Data-change, quality-change, and data-update

These three trigger options support report generation based on change or update in a value of a **DataAttribute** of a member of a **DATA-SET**.

data-change

The trigger option **data-change** (TrgOpEna = dchg) relates to a change in a value of a **DataAttribute** representing the process-related value of the data. If the TrgOpEna (= dchg) is FALSE then no report should be issued on a data-change in the value of that **DataAttribute**.

quality-change

The trigger option **quality-change** (TrgOp = qchg) relates to a change in the quality value of a **DataAttribute**. If the TrgOpEna (= qchg) is FALSE then no report should be issued on a data-change in the value of that **DataAttribute**.

data-update

The trigger option **data update** (TrgOp = dupd) relates to a freeze event in a value of a **DataAttribute** representing a freeze value of the data (for example, frozen counters) or to an event triggered by updating the value of a **DataAttribute**. If the TrgOpEna (= dupd) is FALSE then no report should be issued on a data-change in the value of that **DataAttribute**.

NOTE 1 Data-update trigger condition may be used to issue sending a report or storing a log entry into a log when a value of a **DataAttribute** has updated. Updating may mean that the value has changed or has been "overwritten" with the same value as before. The dupd trigger condition can be used as a trigger for statistics values that may be calculated and updated on a periodic base. Independently of whether the statistics value has changed or not, the value will be reported or logged.

NOTE 2 With the specification of the common data classes in IEC 61850-7-3, the trigger option applying to a specific **DataAttribute** is defined.

When the **BRCB** is notified by an internal event of a data-change, quality-change, or data-update event of a member of the referenced **DATA-SET** whose values are to be reported, the **BRCB** shall include the value of the member of the referenced **DATA-SET** that produced the internal event in the report according to 14.2.3.2.2.9. The value to be reported shall be the value at the time when the internal event occurred.

NOTE 3 For changes that meet more than one **TrgOp** criteria (for example, data-change and quality-change), it is preferable to send only a single report in such a case.

14.2.3.2.3.3 Integrity

The trigger option **integrity** supports integrity report generation. In addition, to activate this trigger option (set TrgOpEna integrity to TRUE), a client shall set the integrity period (IntgPd) to a value greater than 0. When integrity reports are enabled, the **BRCB** shall be notified each time the value of the time as specified in IntgPd has expired. The **BRCB** shall then build a report with the values of **all** members of the referenced **DATA-SET**. If the TrgOpEna (= integrity) is FALSE then no integrity report should be issued.

All buffered events shall be sent before integrity reports can be sent.

A new internal event caused by data-change, quality-change, or data-update (while the transmission of the integrity report is still going on) shall use a new sequence number (and subsequence number starting with 0) and may pass the remaining segments of the integrity report that is still going on.

A new event caused by integrity time (while the transmission of the integrity report is still going on) shall be interpreted as a mis-configured **BRCB**. The new event shall have no effect.

A new general-interrogation request (while the transmission of the integrity report is still going on) shall be deferred until the ongoing transmission of the integrity report has completed. A new general-interrogation report with a new sequence number (and subsequence number starting with 0) shall be generated and sent.

14.2.3.2.3.4 General interrogation (GI)

The trigger option **general interrogation** (GI) shall be used to indicate the request of a general interrogation. After setting the attribute GI to TRUE the **BRCB** shall start the interrogation process and create a report that includes all **DataAttribute** values of the referenced **DATA-SET**. After initiation of the interrogation process the **BRCB** shall automatically set the value of **GI** to FALSE. If the TrgOpEna (= general-interrogation) is FALSE then no integrity report should be issued.

All buffered events shall be sent before general-interrogation reports can be sent.

A new request for general-interrogation (while the transmission of the general-interrogation report is still going on) shall stop sending the remaining segments of the general-interrogation report that is still going on. A new general-interrogation report with a new sequence number (and subsequence number starting with 0) shall be generated and sent.

An new event caused by integrity time (while the transmission of the general-interrogation report is still going on) shall be deferred until the ongoing transmission of the general-interrogation report has completed.

NOTE The general-interrogation is initiated by the client. The integrity report, which also transmits all values of a data set, is initiated by the **BRCB**.

14.2.3.2.3.5 Time sequence order of reports

The **BRCB** within the implementation resource limits shall send all reports in the time sequence order in which the related internal events have been made available.

Reports generated as result of the trigger options **integrity** or **general-interrogation** provide a snapshot of the values of **all** members of the **DATA-SET**. The transmission of these reports shall start with the next sequence number. If all values of the referenced data set do not fit into one single report, several subreports with incremented subsequence number (starting with subsequence number equal shall be sent until all values have been sent. If – while sending these reports or subreports respectively – **DATA** values caused by data-change, quality-change, or data-update need to be sent, this may be done with a new report sent in-between the transmission of the **integrity** or **general-interrogation** reports (subreports) respectively using a new sequence number. In that case, the time sequence order is not maintained, but the higher sequence number may be used by the client to determine the newer values.

NOTE This allows a client to keep a process data image consistent when a report is received while a general-interrogation is in progress. The client needs to keep track of the sequence numbers. When receiving information for a specific data in a report with sequence number (for example, 22) older than the sequence number (for example, 23) of a previously received report with the same data, the client may not use this information to update the process data image.

14.2.3.2.3.6 Buffering events

The **BRCB** shall buffer events based on the trigger options **data-change**, **quality-change**, **data-update**, and **integrity** during loss of association.

After the association is available again, after the client has set the **EntryID**, and enabled the **BRCB**, the **BRCB** shall start sending the reports of events that have been buffered. The **BRCB** shall use the sequence and subsequence numbers so that no gaps occur.

14.2.3.3 GetBRCBValues

14.2.3.3.1 GetBRCBValues parameter table

A client shall use the **GetBRCBValues** service to retrieve attribute values of **BRCB** made visible and thus accessible to the requesting client by the referenced **LOGICAL-NODE**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
BRCBReference
FunctionalConstraint
Response+
ReportIdentifier
ReportEnable
DataSetReference
ConfigurationRevision
OptionalFields
BufferTime
SequenceNumber
TriggerOptionsEnabled
IntegrityPeriod
GeneralInterrogation
EntryIdentifier
Response–
ServiceError

14.2.3.3.2 Request

14.2.3.3.2.1 BRCBReference

The parameter **BRCBReference** shall specify the ObjectReference of the **BRCB**.

The service parameter **BRCBReference** shall be **BRCBRef**.

14.2.3.3.2.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to filter the respective instances of attributes of a **BRCB**.

The service parameter **FunctionalConstraint** shall be **BR**.

14.2.3.3.3 Response+

The parameter **Response+** shall indicate that the service request succeeded.

14.2.3.3.3.1 ReportIdentifier

The parameter **ReportIdentifier** shall contain the value of the corresponding attribute **RptID** of the referenced **BRCB**.

14.2.3.3.3.2 ReportEnable

The parameter **ReportEnable** shall contain the value of the corresponding attribute **RptEna** of the referenced **BRCB**.

14.2.3.3.3.3 DataSetReference

The parameter **DataSetReference** shall contain the value of the corresponding attribute **DatSet** of the referenced **BRCB**.

14.2.3.3.3.4 ConfigurationRevision

The parameter **ConfigurationRevision** shall contain the value of the corresponding attribute **ConfRev** of the referenced **BRCB**.

14.2.3.3.3.5 OptionalFields

The parameter **OptionalFields** shall contain the value of the corresponding attribute **OptFlds** of the referenced **BRCB**.

14.2.3.3.3.6 BufferTime

The parameter **BufferTime** shall contain the value of the corresponding attribute **BufTm** of the referenced **BRCB**.

14.2.3.3.3.7 SequenceNumber

The parameter **SequenceNumber** shall contain the value of the corresponding attribute **SeqNum** of the referenced **BRCB**.

14.2.3.3.3.8 TriggerOptionsEnabled

The parameter **TriggerOptionsEnabled** shall contain the value of the corresponding attribute **TrgOpEna** of the referenced **BRCB**.

14.2.3.3.3.9 IntegrityPeriod

The parameter **IntegrityPeriod** shall contain the value of the corresponding attribute **IntgPd** of the referenced **BRCB**.

14.2.3.3.3.10 GeneralInterrogation

The parameter **GeneralInterrogation** shall contain the value of the corresponding attribute **GI** of the referenced **BRCB**.

14.2.3.3.3.11 EntryIdentifier

The parameter **EntryIdentifier** shall contain the value of the corresponding attribute **EntryID** of the referenced **BRCB**.

14.2.3.3.4 Response–

The **Response–** parameter shall indicate that the service request failed. The appropriate ServiceError shall be returned.

14.2.3.4 SetBRCBValues

14.2.3.4.1 SetBRCBValues parameter table

A client shall use the **SetBRCBValues** service to set attribute values of **BRCB** made visible and thus accessible to the requesting client by the referenced **LOGICAL-NODE**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
BRCBReference
FunctionalConstraint
ReportIdentifier [0..1]
ReportEnable [0..1]
DataSetReference [0..1]
OptionalFields [0..1]
BufferTime [0..1]
TriggerOptionsEnabled [0..1]
IntegrityPeriod [0..1]
GeneralInterrogation [0..1]
PurgeBuffer [0..1]
EntryIdentifier [0..1]
Response+
Response–
ServiceError

14.2.3.4.2 Request

14.2.3.4.2.1 BRCBReference

The parameter **BRCBReference** shall specify the ObjectReference of the **BRCB**.

The service parameter **BRCBReference** shall be **BRCBRef**.

14.2.3.4.2.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to filter the respective attributes of a **BRCB**.

The service parameter **FunctionalConstraint** shall be **BR**.

14.2.3.4.2.3 ReportIdentifier [0..1]

The parameter **ReportIdentifier** shall contain the value for the corresponding attribute **RptID** of the referenced **BRCB**.

14.2.3.4.2.4 ReportEnable [0..1]

The parameter **ReportEnable** shall contain the value for the corresponding attribute **RptEna** of the referenced **BRCB**.

14.2.3.4.2.5 DataSetReference [0..1]

The parameter **DataSetReference** shall contain the value for the corresponding attribute **DatSet** of the referenced **BRCB**.

14.2.3.4.2.6 OptionalFields [0..1]

The parameter **OptionalFields** shall contain the value for the corresponding attribute **OptFlds** of the referenced **BRCB**.

14.2.3.4.2.7 BufferTime [0..1]

The parameter **BufferTime** shall contain the value for the corresponding attribute **BufTm** of the referenced **BRCB**.

14.2.3.4.2.8 TriggerOptionsEnabled [0..1]

The parameter **TriggerOptions** shall contain the value for the corresponding attribute **TrgOpEna** of the referenced **BRCB**.

14.2.3.4.2.9 IntegrityPeriod [0..1]

The parameter **IntegrityPeriod** shall contain the value for the corresponding attribute **IntgPd** of the referenced **BRCB**.

14.2.3.4.2.10 GeneralInterrogation [0..1]

The parameter **GeneralInterrogation** shall contain the value for the corresponding attribute **GI** of the referenced **BRCB**.

14.2.3.4.2.11 PurgeBuffer [0..1]

The parameter **PurgeBuffer** shall contain the value for the corresponding attribute **PurgeBuf** of the referenced **BRCB**.

14.2.3.4.2.12 EntryIdentifier

The parameter **EntryIdentifier** shall contain the value of the corresponding attribute **EntryID** of the referenced **BRCB**.

14.2.3.4.3 Response+

The parameter **Response+** shall indicate that the service request succeeded.

14.2.3.4.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

This service shall return a failure if the service has been issued for any attribute of a **BRCB** other than setting the **RptEna** to FALSE while **BRCB** is enabled.

14.2.4 UNBUFFERED-REPORT-CONTROL-BLOCK (URCB) class definition

14.2.4.1 URCB class Syntax

The **URCB** class shall have the structure defined in Table 25.

Table 25 – URCB class definition

URCB class				
Attribute name	**Attribute type**	**FC**	**TrgOp**	**Value/value range/explanation**
URCBName	ObjectName	-	-	Instance name of an instance of URCB
URCBRef	ObjectReference	-	-	Path-name of an instance of URCB
Specific to report handler				
RptID	VISIBLE STRING65	RP	-	
RptEna	BOOLEAN	RP	dchg	
Resv	BOOLEAN	RP	-	
DatSet	ObjectReference	RP	dchg	
ConfRev	INT32U	RP	dchg	
OptFlds	PACKED LIST	RP	dchg	
reserved	BOOLEAN			
sequence-number	BOOLEAN			
time-of-entry	BOOLEAN			
reason-for-inclusion	BOOLEAN			
data-set-name	BOOLEAN			
data-reference	BOOLEAN			
reserved	BOOLEAN			Used for buffer-overflow in BRCB
reserved	BOOLEAN			Used for entryID in BRCB
BufTm	INT32U	RP	dchg	0 .. MAX
SeqNum	INT8U	RP	-	
TrgOpEna	TriggerConditions	RP	dchg	
IntgPd	INT32U	RP	dchg	0.. MAX
GI	BOOLEAN	BR	-	
Services Report GetURCBValues SetURCBValues				

Except **URCBName**, **URCBRef**, **RptEna**, and **Resv** all other attributes shall be as defined for the **BRCB** in 14.2.2.

14.2.4.2 URCBName – unbuffered report control name

The attribute **URCBName** shall be the name of the **URCB** that unambiguously identifies the **URCB** within **LOGICAL-NODE**.

14.2.4.3 URCBRef – unbuffered report control ObjectReference

The attribute **URCBRef** shall be the unique path-name of **URCB**.

The ObjectReference **URCBRef** shall be:

LDName/LNName.URCBName

14.2.4.4 RptEna – report enable

The attribute **RptEna** (if set to TRUE) shall indicate that the **URCB** is currently enabled to report values of the **DATA-SET**. If set to TRUE, the **URCB** shall monitor the referenced value of the **DATA-SET** and generate the reports as specified in the **URCB**. If set to FALSE the **URCB** shall stop issuing reports.

While being TRUE (report enabled), no changes of attribute values of the **URCB** other than disabling and activating the trigger options general-integration shall be allowed.

If the **TWO-PARTY-APPLICATION-ASSOCIATION** to the client over which **URCB** has been enabled is lost, the server shall set the attribute **RptEna** to FALSE.

14.2.4.5 Resv – reserve URCB

The attribute **Resv** (if set to TRUE) shall indicate that the **URCB** is currently exclusivly reserved for the client that has set the value to TRUE. Other clients shall not be allowed to set any attribute of that **URCB**.

If the attribute **Resv** is not set to TRUE, then setting the attribute **RptEna** to TRUE reserves the instance implicitly.

NOTE The attribute **Resv** functions as a semaphore for the configuration, enabling and disabling of the **URCB**.

14.2.5 URCB class services

14.2.5.1 Overview

For the **URCB** the following services are defined.

Service	Description
Report	Send a report
GetURCBValues	Read an attribute of an instance of **URCB**
SetURCBValues	Write an attribute of an instance of **URCB**

14.2.5.2 Report

The report service shall be as defined for **BRCB** in 14.2.3.2, except that the parameter **BufOvfl** of the report format shall not be available.

14.2.5.3 GetURCBValues

A client shall use the **GetURCBValues** service to retrieve attribute values of an **URCB** made visible and thus accessible to the requesting client by the referenced **LOGICAL-NODE**.

The service shall be as defined in 14.2.3.3, except that the parameter **BRCBReference** shall be **URCBReference**, the parameter **PurgeBuffer** shall not be available, and the parameter **functional constraint** shall be **RP**.

14.2.5.4 SetURCBValues

A client shall use the **SetURCBValues** service to set attribute values of **URCB** made visible and thus accessible to the requesting client by the referenced **LOGICAL-NODE**.

The service shall be as defined in 14.2.3.4, except that the parameter **BRCBReference** shall be **URCBReference**, the parameter **PurgeBuffer** shall not be available, and the parameter **functional constraint** shall be **RP**.

14.3 LOG-CONTROL-BLOCK class model

14.3.1 General

14.3.1.1 Basic concepts

Many IEDs have requirements for the internal storage of historical data values and retrieval over communications systems. This data values fall into two general categories: **periodic recordings** (commonly referred to in metering applications as profiles) and **event-triggered or "sequence-of-events" (SOE)** data. Several criteria are be used to differentiate historical data logging requirements from report-oriented information transfer.

– Data logging shall be independent of external application associations or other communication transactions. Even if communication is lost, historical events shall occur and shall be logged.

– The process of storing the historical records is completely asynchronous with retrieval over communications.

– The rate of generation of historical records can in some cases be much faster than the ability of communication processes to report the values to an external data base.

– Record retrieval shall allow external applications to request subsets of the entire historical data base for the purpose of maintaining an external, complete time or event-sequenced historical record.

– The source of the data may be external to the device. Thus, the historical repository may simply be a central point of storage.

– Records have relative significance with regard to time or ordering and may require the assignment of a sequence number.

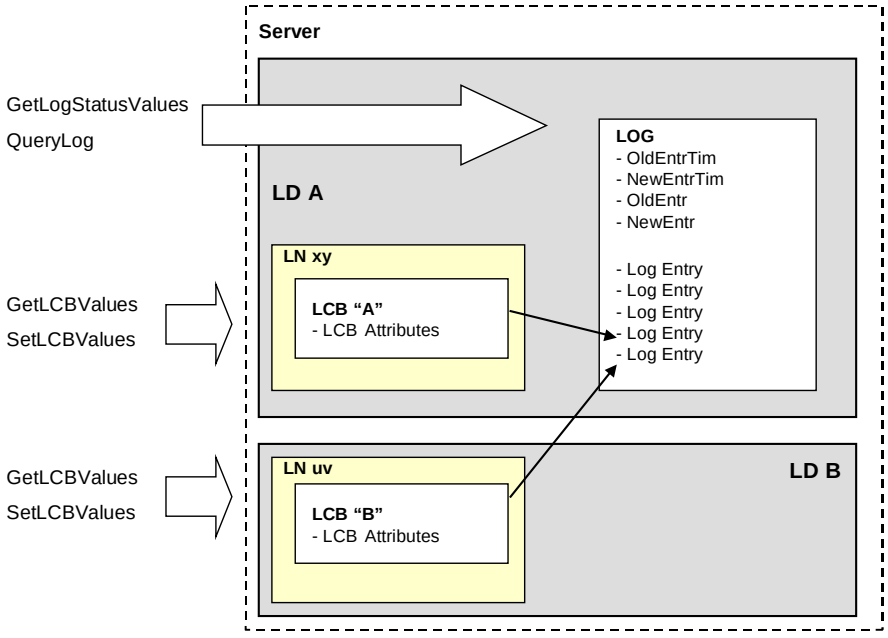


Figure 25 – Log model overview

Figure 25 gives an overview of the **LOG** and **LCB** classes. One **LOG** may be controlled by multiple **LCBs**.

14.3.1.2 The log buffer concept

From an implementation view, the **LOG** may be considered as a circular buffer that overwrites the oldest values in the **LOG**. However, this is hidden from the client. The client view of the **LOG** is a linear buffer, where the **LOG** entries are identified by

– **EntryID**: a unique identifier of a **LOG** entry;

– **TimeOfEntry**: the time, when the **LOG** entry has been added to the **LOG**.

EntryID shall be a counter that rolls over when the maximal value has been reached. The size of that counter shall be larger than the maximal number of entries that can be stored in a **LOG** so that there may not be two entries in the log with the same value of **entryID**. **EntryID** together with **TimeOfEntry** provide a unique identification of the entry.

A client may query the **LOG** by **entryID** or by **TimeOfEntry**.

14.3.2 LCB class definition

14.3.2.1 LCB class syntax

The **LCB** shall control the procedures that are required for storing values of **DataAttribute** (the log entry) into a **LOG**. Each enabled **LCB** shall associate **DATA-SET** with a **LOG**. Changes in a value of a member of a **DATA-SET** shall be stored as LOG entry. Multiple **LCBs** allow multiple **DATA-SETs** to feed a **LOG**.

It shall be the responsibility of access control, to prevent unauthorized clients to modify an **LCB**.

NOTE The internal notification, local storage mechanism, internal formats, etc. for log entries are all local issues and outside the scope of this part of IEC 61850.

The **LCB** shall have the structure specified in Table 26.

Table 26 – LCB class definition

LCB class				
Attribute name	**Attribute type**	**FC**	**TrgOp**	**Value/value range/explanation**
LCBName	ObjectName	-	-	Instance name of an instance of LCB
LCBRef	ObjectReference	-	-	Path-name of an instance of LCB
Specific to log handler				
LogEna	BOOLEAN	LG	dchg	
DatSet	ObjectReference	LG	dchg	
TrgOpEna	TriggerConditions	LG	dchg	Valid values for TrgOpEna of type TriggerConditions shall be dchg, qchg, dupd, and integrity.
IntgPd	INT32U	LG	dchg	1..MAX; 0 implies no integrity logging.
Specific to building the log				
LogRef	ObjectReference	LG		
Services GetLCBValues SetLCBValues				

14.3.2.2 LCB class attributes

14.3.2.2.1 LCBName – log control name

The attribute **LCBName** shall unambiguously identify a **LCB** within the scope of a **LOGICAL-NODE**.

14.3.2.2.2 LCBRef – log control ObjectReference

The attribute **LCBRef** shall be the unique path-name of a **LCB**.

The ObjectReference **LCBRef** shall be:

LDName/LNName.LCBName

14.3.2.2.3 LogEna – log enable

The attribute **LogEna** shall indicate that this **LCB** is recording into the **LOG** specified by **LogRef**.

A transition of **LogEna** from disabled to enabled or from enabled to disabled shall cause a log entry to be placed into the **LOG**.

NOTE The attribute LogEna may be set to TRUE automatically by a server after turning the server on.

While in the state enabled no changes of attribute values of **LCB** other than disabling shall be allowed.

14.3.2.2.4 DatSet – data set reference

The attribute **DatSet** shall indicate the **DATA-SET**, whose member values are to be logged.

14.3.2.2.5 TrgOpEna – trigger options enabled

The attribute **TrgOpEna** shall specify the trigger conditions that shall be monitored by this **URCB** to cause a Log entry to be created. The values defined are the same as for reporting (see 14.2.2.11).

The TrgOpEna general-interrogation shall not be supported for logging.

14.3.2.2.6 IntgPd – integrity period

If **TrgOpEna** is set to integrity, the attribute **IntgPd** indicates the period in milliseconds used for logging caused by integrity scans.

14.3.2.2.7 LogRef – log reference

The attribute **LogRef** shall be the reference of the **LOG** to which values of members of the referenced **DATA-SET** shall be recorded.

14.3.2.3 LCB services – Overview

For the **LCB** the following services are defined:

Service	Description
GetLCBValues	Retrieve the attribute values of a **LCB**
SetLCBValues	Set the attributes values of a **LCB**

14.3.2.4 GetLCBValues

A client shall use the **GetLCBValues** service to retrieve attribute values of **LCB** made visible and thus accessible to the requesting client by the referenced **LOGICAL-NODE**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
LCBReference
FunctionalConstraint
Response+
LogEnable
DataSetReference
OptionalFields
IntegrityPeriod
LogReference
Response–
ServiceError

14.3.2.4.1 Request

14.3.2.4.1.1 LCBReference

The parameter **LCBReference** shall specify the ObjectReference of the **LCB**.

The service parameter **LCBReference** shall be **LCBRef**.

14.3.2.4.1.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to filter the respective instances of attributes of a **LCB**.

The service parameter **FunctionalConstraint** shall be **LG** (logging).

14.3.2.4.2 Response+

The parameter **Response+** shall indicate that the service request succeeded.

14.3.2.4.2.1 LogEnable

The parameter **LogEnable** shall contain the value for the corresponding attribute **LogEna** of the referenced **LCB**.

14.3.2.4.2.2 DataSetReference

The parameter **DataSetReference** shall contain the value for the corresponding attribute **DatSet** of the referenced **LCB**.

14.3.2.4.2.3 OptionalFields

The parameter **OptionalFields** shall contain the value for the corresponding attribute **TrgOpEna** of the referenced **LCB**.

14.3.2.4.2.4 IntegrityPeriod

The parameter **IntegrityPeriod** shall contain the value for the corresponding attribute **IntgPd** of the referenced **LCB**.

14.3.2.4.2.5 LogReference

The parameter **LogReference** shall contain the value for the corresponding attribute **LogRef** of the referenced **LCB**.

14.3.2.4.3 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

14.3.2.5 SetLCBValues

A client shall use the **SetLCBValues** service to set attribute values of **LCB** made visible and thus accessible to the requesting client by the referenced **LOGICAL-NODE**

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
LCBReference
FunctionalConstraint
LogEnable [0..1]
DataSetReference [0..1]
OptionalFields [0..1]
IntegrityPeriod [0..1]
LogReference [0..1]
Response+
Response–
ServiceError

14.3.2.5.1 Request

14.3.2.5.1.1 LCBReference

The parameter **LCBReference** shall specify the ObjectReference of **LCB**.

The service parameter **LCBReference** shall be **LCBRef**.

14.3.2.5.1.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to filter the respective instances of attributes of a **LCB**.

The service parameter **FunctionalConstraint** shall be **LG** (logging).

14.3.2.5.1.3 LogEnable [0..1]

The parameter **LogEnable** shall contain the value of the corresponding attribute **LogEna** of the referenced **LCB**.

14.3.2.5.1.4 DataSetReference [0..1]

The parameter **DataSetReference** shall contain the value of the corresponding attribute **DatSet** of the referenced **LCB**.

14.3.2.5.1.5 OptionalFields [0..1]

The parameter **OptionalFields** shall contain the value of the corresponding attribute **TrgOpEna** of the referenced **LCB**.

14.3.2.5.1.6 IntegrityPeriod [0..1]

The parameter **IntegrityPeriod** shall contain the value of the corresponding attribute **IntgPd** of the referenced **LCB**.

14.3.2.5.1.7 LogReference [0..1]

The parameter **LogReference** shall contain the value of the corresponding attribute **LogRef** of the referenced **LCB**.

14.3.2.5.2 Response+

The parameter **Response+** shall indicate that the service request succeeded.

14.3.2.5.3 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

This service shall return a failure if the service has been issued for any attribute of a **LCB** other than enable while **LCB** is enabled.

14.3.3 LOG class definition

14.3.3.1 LOG class syntax

The **LOG** shall be filled on a first-in first-out basis. When the list of log entries reaches a point where the stored data reaches the maximal size of the log, the oldest log entry shall be overwritten. This action shall have no impact to the further incrementing of the **EntryID** of the added log entries.

The **LOG** shall have the structure defined in Table 27.

Table 27 – LOG class definition

LOG class			
Attribute name	**Attribute type**	**FC**	**Value/value range/explanation**
LogName	ObjectName		Instance name of an instance of LOG
LogRef	ObjectReference		Path-name of an instance of LOG
OldEntrTm	TimeStamp	LG	
NewEntrTm	TimeStamp	LG	
OldEntr	INT32U	LG	
NewEntr	INT32U	LG	
Entry [1..n]			
TimeOfEntry	EntryTime		
EntryID	EntryID		
EntryData [1..n]			
DataRef	ObjectReference		
Value	(*)		(*) type(s) depend on the definition of common data classes in IEC 61850-7-3
ReasonCode	TriggerConditions		ReasonCode general-interrogation shall never occur as TRUE. OPTIONAL
Services			
QueryLogByTime QueryLogAfter GetLogStatusValues			

14.3.3.2 LOG class attributes

14.3.3.2.1 LogName – log name

The attribute **LogName** shall unambiguously identify a **LCB** within the scope of LLN0. The **LogName** shall be the name of the **LOGICAL-DEVICE**.

14.3.3.2.2 LogRef – log reference

The attribute **LogRef** shall be the unique path-name of a **LOG**.

The ObjectReference **LogRef** shall be:

LDName/LDName

Only one **Log** per **LD** shall be allowed.

14.3.3.2.3 OldEntrTm – oldest log entry time of log

The attribute **OldEntrTm** shall indicate the time when the oldest log entry has been stored.

NOTE That is the time when the entry has been stored in the log. This is different from the time stamp of the entry itself, which indicates when the event that caused the creation of the log entry has occurred.

14.3.3.2.4 NewEntrTm – newest log entry time of log

The attribute **NewEntrTm** shall indicate the time when the newest log entry has been stored.

14.3.3.2.5 OldEntr – oldest log entry sequence number

The attribute **OldEntr** shall indicate the **EntryID** for the oldest entry available in the log.

14.3.3.2.6 NewEntr – newest log entry sequence number

The attribute **NewEntr** shall indicate the **EntryID** for the newest entry available in the log.

14.3.3.2.7 Entry [1..n]

14.3.3.2.7.1 TimeOfEntry – time of log entry

The attribute **TimeOfEntry** shall be the time, when the log entry is added to a **LOG**. That time may be different to the time stamp of the data, which shall be the time when the event occurred that caused the log entry to be created.

14.3.3.2.7.2 EntryID – entry identifier

The attribute **EntryID** shall be a unique reference to all log entries having the same value of TimeOfEntry.

14.3.3.2.7.3 EntryData [1..n] – Data of Entry

The parameter **EntryData** shall contain the data reference, values, and reasonCode of each member of the **DATA-SET** to be included in the log entry. The value shall comprise the values of all data attributes of the member of **DATA-SET**.

DataRef

The parameter **DataRef** shall contain the **functionally constrained data attribute** (**FCDA**) of the **DataAttribute** values included in the report.

NOTE **FCDA** may reference **DataAttribute** values contained in different **LOGICAL-NODE**s.

Value

The parameter **Value** shall contain the **DataAttribute** values included in the log entry.

The number of members of the **DATA-SET** whose values shall be included in the report shall depend on the **TrgOpEna** of the **LCB** selected and the following values of **TrgOp** of the respective **DataAttributes**:

In case of **TrgOp** (**dchg**, **qchg**, and **data-update**) only the value of the member of a **DATA-SET** shall be included in the log entry that produced the internal event.

In case of setting the **LCB** attribute **IntPd** to TRUE and **TrgOpEna integrity** (=TRUE) **all** values of **all** members of a **DATA-SET** shall be included in the log entry that produced the internal event.

ReasonCode – reason for inclusion

The reason for inclusion shall be set according the **TrgOp** that caused the creation of the **EntryData**. The value for reason for inclusion shall be set according the **TrgOp** that caused the creation of the report. The value range for reasons for inclusion shall be as listed:

– data-change (caused by TrgOp = dchg in an instance of **DATA**)
– quality-change (caused by TrgOp = qchg in an instance of **DATA**)
– data-update (caused by TrgOp = dupd in an instance of **DATA**)
– integrity (caused by the attribute **IntgPd** in the **LCB**)

14.3.4 Procedures to generate the log entries

14.3.4.1 Overview

Basically, the conditions and constraints for log generation shall be the same as for report generation (see 14.2.3.2.3). Subclause 14.3.4 specifies the differences only.

14.3.4.2 Trigger Options data-change, quality-change, or data-update

When the **LCB** is notified by an internal event of a data-change, a quality-change, a data-update of the referenced member of a **DATA-SET**, the **LCB** shall create a LOG entry with the value of the member of **DATA-SET** that produced the internal event.

14.3.4.3 Trigger options integrity

When a **LCB** is notified as a result of the trigger options integrity, the **LCB** shall create a LOG entry for each member of the referenced **DATA-SET**.

14.3.5 LOG services

14.3.5.1 Overview

For the **LOG** model, the following services are defined:

Service	Description
QueryLogByTime	Read the log entries selected by time
QueryLogAfter	Read the log entries selected by entryID
GetLogStatusValues	Get the status values of a **LOG**

14.3.5.2 QueryLogByTime

14.3.5.2.1 QueryLogByTime parameter table

A client shall use the **QueryLogByTime** service to retrieve a range of **LOG** entries from a **LOG** based on time ranges (RangeStartTime and RangeStopTime).

Parameter name
Request
LogReference
RangeStartTime
RangeStopTime
Response+
ListOfLogEntries
Response–
ServiceError

14.3.5.2.2 Request

14.3.5.2.2.1 LogReference

The parameter **LogReference** shall contain the ObjectReference **LogRef** of a **LOG**. The ObjectReference **LogReference** shall be:

LDName/LDName

14.3.5.2.2.2 RangeStartTime

The parameter **RangeStartTime** shall contain the range start time to retrieve log entries. The first log entry selected shall be the first entry in the log with a **RangeStartTime** greater than, or equal to, the **RangeStartTime**. In the case where no **RangeStartTime** is specified, the first log entry contained in the log shall be the first entry selected for transmission.

14.3.5.2.2.3 RangeStopTime

The parameter **RangeStopTime** shall contain the range stop time to retrieve log entries. The last log entry selected shall be the last entry in the log with a **RangeStopTime** less than, or equal to, the **RangeStopTime**. For the case where no **RangeStopTime** is specified, the last log entry contained in the log shall be the last entry selected.

14.3.5.2.3 Response+

ListOfLogEntries

The parameter **ListOfLogEntries** shall contain the list of log entries that are in the range as specified with the parameters **RangeStartTime** and **RangeStopTime** of the service request.

14.3.5.2.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

14.3.5.3 QueryLogAfter

14.3.5.3.1 QueryLogAfter parameter table

A client shall use the **QueryLogAfter** service to retrieve a range of LOG entries from the referenced LOG based on ranges of IDs that are after the **RangeStartTime** and **Entry**.

Parameter Name
Request
LogReference
RangeStartTime
Entry
Response+
ListOfLogEntries
Response-
ServiceError

14.3.5.3.2 Request

14.3.5.3.2.1 LogReference

The parameter **LogReference** shall specify the ObjectReference **LogRef** of the **LOG**. The ObjectReference **LogReference** shall be:

LDName/LDName

14.3.5.3.2.2 RangeStartTime

The parameter **RangeStartTime** shall contain the time of the log entry (or log entries – in case of multiple entries for a single time stamp) selected.

14.3.5.3.2.3 Entry

The parameter **Entry** shall reference the LOG entry of the selected **RangeStartTime** after which the log entries shall be selected.

14.3.5.3.3 Response+

14.3.5.3.3.1 ListOfLogEntries

The parameter **ListOfLogEntries** shall contain the list of log entries that follow after the entries as specified with the parameters **RangeStartTime** and **Entry** of the service request.

14.3.5.3.4 Response-

The parameter **Response-** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

14.3.5.4 GetLogStatusValues

A client shall use the **GetLCBValues** service to retrieve the attribute values of a **LOG** made visible and thus accessible to the requesting client by the referenced **LLN0**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
LogReference
FunctionalConstraint
Response+
OldestEntryTime
NewestEntryTime
OldestEntry
NewestEntry
Response–
ServiceError

14.3.5.4.1 Request

14.3.5.4.1.1 LogReference

The parameter **LogReference** shall specify the ObjectReference of the **LOG**.

The service parameter **LogReference** shall be:

LDName/LDName

14.3.5.4.1.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to filter the respective instances of attributes of a **LOG**.

The service parameter **FunctionalConstraint** shall be **LG** (logging).

14.3.5.4.2 Response+

The parameter **Response+** shall indicate that the service request succeeded.

14.3.5.4.2.1 OldestEntryTime

The parameter **OldestEntryTime** shall contain the value for the corresponding attribute **OldEntrTm** of the referenced **LOG**.

14.3.5.4.2.2 NewestEntryTime

The parameter **NewestEntryTime** shall contain the value for the corresponding attribute **NewEntrTm** of the referenced **LOG**.

14.3.5.4.2.3 OldestEntry

The parameter **OldestEntry** shall contain the value for the corresponding attribute **OldEntr** of the referenced **LOG**.

14.3.5.4.2.4 NewestEntry

The parameter **NeestEntry** shall contain the value for the corresponding attribute **NewEntr** of the referenced **LOG**.

14.3.5.4.3 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

15 Generic substation event class model (GSE)

15.1 Overview

The generic substation event model provides the possibility for a fast and reliable system-wide distribution of input and output data values. The generic substation event model is based on the concept of an autonomous decentralization, providing an efficient method allowing the simultaneous delivery of the same generic substation event information to more than one physical device through the use of multicast/broadcast services.

For the purposes of the generic substation event model, conveyed values are seen from the viewpoint of the reporting logical device.

NOTE 1 It is a matter for the mapping and implementation how reliability and a short transmission delay are achieved. Depending on the SCSM and communication stack being used, different methods may be implemented.

The generic substation event model applies to the exchange of values of a collection of **DataAttribute**. Two control classes and the structure of two messages are defined in this clause:

– generic object oriented substation event (**GOOSE**) supports the exchange of a wide range of possible common data organized by a **DATA-SET**.

– generic substation state event (**GSSE**) provides the capability to convey state change information (bit pairs).

NOTE 2 The GSSE represents the GOOSE model as defined in UCA™ Version 2.

The information exchange is based on a publisher/subscriber mechanism. The publisher writes the values in a local buffer at the sending side; the receiver reads the values from a local buffer at the receiving side. The communication system is responsible to update the local buffers of the subscribers. A generic substation event control class in the publisher is used to control the procedure.

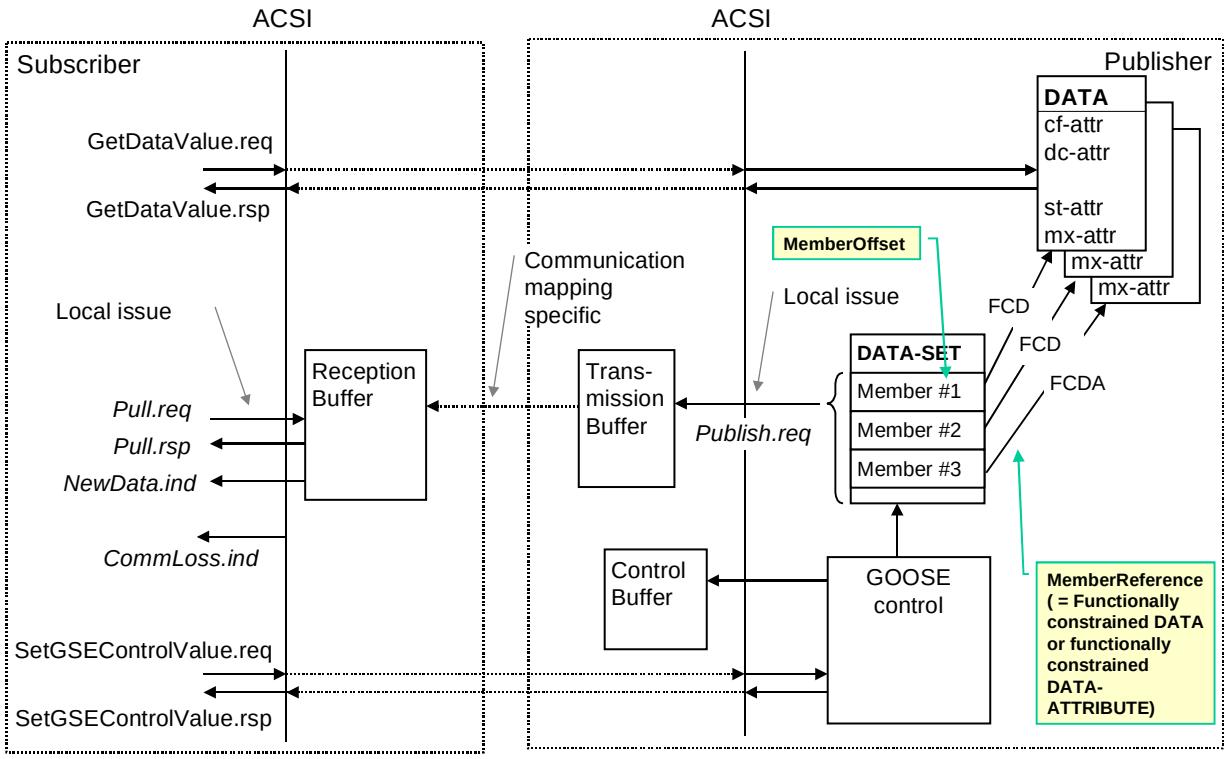


Figure 26 – GoCB model

Figure 26 gives an overview of the classes and services of the GOOSE model. The message exchange is based on the multicast application association. If the value of one or several **DataAttributes** of a specific functional constraint (for example, st) in the **DATA-SET** change, the transmission buffer of the publisher is updated with the local service "publish" and the values are transmitted with a GOOSE message. The **DATA-SET** may have several members (numbered from 1 up – the numbers shall be called **MemberOffset**). Each member shall have a **MemberReference** referencing the **DataAttribute** with a specific functional constraint (**FC**). Mapping specific services of the communication network will update the content of the buffer in the subscribers. New values received in the reception buffer are signalled to the application.

The GOOSE messages contain information that allow the receiving device to know that a status has changed and the time of the last status change. The time of the last status change allows a receiving device to set local timers relating to a given event.

A newly activated device, upon power-up or reinstatement to service, shall send current data (status) or values as the initial GOOSE message. Moreover, all devices sending GOOSE messages shall continue to send the message with a long cycle time, even if no status/value change has occurred. This ensures that devices that have been activated recently will know the current status values of their peer devices.

NOTE 3 The GSSE model is similar to the GOOSE model. The basic concept described above applies also to the GSSE model. One major difference is the kind of information exchanged. GOOSE provides a flexible means to specify which information is to be exchanged (**DATA-SET**) whereas GSSE provides a simple list of status information.

The behaviour of the **GoCB** shall apply to the **GsCB**.

15.2 GOOSE-CONTROL-BLOCK (GoCB) class

15.2.1 GoCB definition

The **GoCB** shall be as defined in Table 28.

Table 28 – GOOSE control block class definition

GoCB class				
Attribute name	**Attribute type**	**FC**	**TrgOp**	**Value/value range/explanation**
GoCBName	ObjectName	GO	-	Instance name of an instance of GoCB
GoCBRef	ObjectReference	GO	-	Path-name of an instance of GoCB
GoEna	BOOLEAN	GO	dchg	Enabled (TRUE) | disabled (FALSE)
AppID	VISIBLE STRING65	GO		Attribute that allows a user to assign a system unique identification for the application that is issuing the GOOSE. DEFAULT GoCBRef
DatSet	ObjectReference	GO	dchg	
ConfRev	INT32U	GO	dchg	
NdsCom	BOOLEAN	GO	dchg	
Services SendGOOSEMessage GetReference GetGOOSEElementNumber GetGoCBValues SetGoCBValues				

15.2.1.1 GoCBName – GOOSE control name

The attribute GoCBName shall unambiguously identify a **GoCB** within the scope of a **LLN0**.

15.2.1.2 GoCBRef – GOOSE control reference

The attribute **GoCBRef** shall be the unique path-name of a **GoCB** within the **LLN0**.

The ObjectReference **GoCBRef** shall be:

LDName/LLN0.GoCBName

15.2.1.3 GoEna – GOOSE enable

The attribute **GoEna** (if set to TRUE) shall indicate that the **GoCB** is currently enabled to send GOOSE messages. If set to FALSE the **GoCB** shall stop sending **GOOSE** messages.

While being TRUE (**GoCB** enabled), no changes of attribute values of the **GoCB** other than disabling shall be allowed.

15.2.1.4 AppID – application identification

The attribute **AppID** shall be a visible string that represents a **LOGICAL-DEVICE** in which the **GoCB** is located. The default value of AppID shall be that of the ObjectReference of a **GoCB**. However, the value may be set to another value as part of a system wide configuration.

NOTE Depending upon the SCSM and actual implementation, it may not be possible to uniquely identify the GOOSE control or GSSE control through the control reference. Therefore, a standardized control attribute must be provided to allow the system configuration process to be able to uniquely identify the control within the scope of the substation.

15.2.1.5 DatSet – data set reference

The attribute **DatSet** shall represent the reference of the **DATA-SET** whose values of members shall be transmitted. The members of the **DATA-SET** shall be uniquely numbered beginning with 1. This number is called the **MemberOffset** of a given member. Each member of the **DATA-SET** has a unique number and a **MemberReference** (the functionally constraint DATA (FCD) or DataAttribute (FCDA)).

NOTE The service GetReference retrieves the **FCD/FCDA** for a given number, and the service GetGOOSEElementNumber retrieves a number for a given **FCD/FCDA**.

The initial value of the referenced members of the **DATA-SET** shall be a local issue.

15.2.1.6 ConfRev – configuration revision

The attribute **ConfRev** shall represent a count of the number of times that the configuration of the **DATA-SET** referenced by **DatSet** has been changed. Changes that shall be counted are:

– any deletion of a member of the **DATA-SET**;

– the reordering of members of the **DATA-SET**; and

– changing the value of the attribute **DatSet**.

The counter shall be incremented when the configuration changes.

The initial value for **ConfRev** is outside the scope of this part of IEC 61850. The value of 0 shall be reserved. A restart of the IED shall not reset the value.

NOTE Configuration changes of **DATA-SETs** due to processing of services are not allowed (see **DATA-SET** model). Changes to be taken into account for the **ConfRev** are those made by local means like system configuration.

15.2.1.7 NdsCom – needs commissioning

The attribute **NdsCom** shall have a value of TRUE if the attribute **DatSet** has a value of NULL. It shall be used to indicate that the **GoCB** requires further configuration.

NOTE Certain implementations and mappings may have a constraint placed upon the number of values and amount of information that can be sent via GOOSE. This attribute represents a mechanism for generating an indication that the configured data set with reference DatSetRef has exceeded the local limit.

If the number or size of values being conveyed by the elements in the **DatSet** referenced **DATA-SET** exceeds the SCSM determined maximum number, then the **NdsCom** attribute shall be set TRUE.

15.2.2 GOOSE service Definitions

15.2.2.1 Overview

For the **GoCB** the following services are defined:

Service	Description
SendGOOSEMessage	Send GOOSE message
GetReference	Retrieve the **FCD/FCDA** of a specific member of **DATA-SET** associated with the GOOSE message
GetGOOSEElementNumber	Retrieve the position of the member in the **DATA-SET** associated with the GOOSE message of a **FCD/FCDA**
GetGoCBValues	Retrieve the attributes of a **GoCB**
SetGoCBValues	Write the attributes of a **GoCB**

15.2.2.2 SendGOOSEMessage

15.2.2.2.1 SendGOOSEMessage parameter table

The **SendGOOSEMessage** service shall be used by a **GoCB** to send a GOOSE message over a MULTICAST-APPLICATION-ASSOCIATION.

Parameter name
Request
GOOSE message

15.2.2.2.2 Request

GOOSE message

The parameter **GOOSE message** shall specify the GOOSE message as defined in 15.2.3 of the given **GoCB**.

15.2.2.3 GetReference

15.2.2.3.1 GetReference parameter table

A client shall use the **GetReference** service to retrieve the **MemberReferences** of specific members of the **DATA-SET** of the referenced **GoCB**.

Parameter name
Request
GoCBReference
MemberOffset [1..n]
Response+
GoCBReference
ConfigurationRevision
MemberReference [1..n]
Response–
ServiceError

15.2.2.3.2 Request

15.2.2.3.2.1 GoCBReference

The parameter **GoCBReference** shall identify the attribute **GoCBRef** of the **GoCB** for which **MemberReferences** are being requested.

15.2.2.3.2.2 MemberOffset [1..n]

The parameter **MemberOffset** shall contain a number identifying a member of the **DATA-SET** referenced by the attribute **DatSet**.

15.2.2.3.3 Response+

15.2.2.3.3.1 GoCBReference

The parameter **GoCBReference** shall contain the parameter that identifies the attribute **GoCBRef** of the **GoCB** for which **MemberReferences** are returned.

15.2.2.3.3.2 ConfigurationRevision

The parameter **ConfigurationRevision** shall contain the attribute **ConfRev** of the **GoCB**.

15.2.2.3.3.3 MemberReference [1..n]

The parameter **MemberReference** shall contain the **MemberReference** requested for the **MemberOffset** of a member of the **DATA-SET**. A value of NULL shall indicate that no member of the referenced **DATA-SET** is defined for the member being requested with a **MemberOffset**.

15.2.2.3.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

15.2.2.4 GetGOOSEElementNumber

15.2.2.4.1 GetGOOSEElementNumber parameter table

A client shall use the **GetGOOSEElementNumber** service to retrieve the member position of a selected **DataAttribute** in the **DATA-SET** associated with a **GoCB**.

Parameter name
Request
GoCBReference
MemberReference [1..n]
Response+
GoCBReference
ConfigurationRevision
MemberOffset [1..n]
Response–
ServiceError

15.2.2.4.2 Request

15.2.2.4.2.1 GoCBReference

The parameter **GoCBReference** shall identify the attribute **GoCBRef** of the **GoCB** for which **MemberOffset** are being requested.

15.2.2.4.2.2 MemberReference [1..n]

The parameter **MemberReference** shall contain the **MemberReference** for which the **MemberOffset** of a member of the **DATA-SET** is requested. A value of NULL is reserved to indicate that no member of the referenced **DATA-SET** is defined for the member being requested with a **MemberReference**.

15.2.2.4.3 Response+

15.2.2.4.3.1 GoCBReference

The parameter **GoCBReference** shall contain the parameter that identifies the attribute **GoCBRef** of the **GoCB** for which **MemberOffsets** are returned.

15.2.2.4.3.2 ConfigurationRevision

The parameter **ConfigurationRevision** shall contain the attribute **ConfRev** of the **GoCB**.

15.2.2.4.3.3 MemberOffset [1..n]

The parameter **MemberOffset** shall contain the **MemberOffset** requested for the **MemberReference** of a member of the **DATA-SET**. A value of NULL shall indicate that no member of the referenced **DATA-SET** is defined matching with a **MemberReference**.

15.2.2.4.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

15.2.2.5 GetGoCBValues

A client shall use the **GetoCBValues** service to retrieve attribute values of **GoCB** made visible and thus accessible to the requesting client by the referenced **LLN0**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
GoCBReference
FunctionalConstraint
Response+
GoEnable
ApplicationID
DataSetReference
ConfigurationRevision
NeedsCommissioning
Response–
ServiceError

15.2.2.5.1 Request

15.2.2.5.1.1 GoCBReference

The parameter **GoCBReference** shall specify the ObjectReference of the **GoCB**.

The service parameter **LCBReference** shall be **LDName/LLN0.GoCBName**.

15.2.2.5.1.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to filter the respective instances of attributes of a **GoCB**.

The service parameter **FunctionalConstraint** shall be **GO** (goose control).

15.2.2.5.2 Response+

The parameter **Response+** shall indicate that the service request succeeded.

15.2.2.5.2.1 GoEnable

The parameter **GoEnable** shall contain the value of the corresponding attribute **GoEna** of the referenced **GoCB**.

15.2.2.5.2.2 ApplicationID

The parameter **ApplicationID** shall contain the value of the corresponding attribute **AppID** of the referenced **GoCB**.

15.2.2.5.2.3 DataSetReference

The parameter **DataSetReference** shall contain the value of the corresponding attribute **DatSet** of the referenced **GoCB**.

15.2.2.5.2.4 ConfigurationRevision

The parameter **ConfigurationRevision** shall contain the value of the corresponding attribute **ConfRev** of the **GoCB**.

15.2.2.5.2.5 NeedsCommissioning

The parameter **NeedsCommissioning** shall contain the value of the corresponding attribute **NdsCom** of the **GoCB**.

15.2.2.5.3 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

15.2.2.6 SetGoCBValues

A client shall use the **SetGoCBValues** service to set attribute values of **GoCB** made visible and thus accessible to the requesting client by the referenced **LLN0**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
GoCBReference
FunctionalConstraint
GoEnable [0..1]
ApplicationID [0..1]
DataSetReference [0..1]
Response+
Response–
ServiceError

15.2.2.6.1 Request

15.2.2.6.1.1 GoCBReference

The parameter **GoCBReference** shall specify the ObjectReference of the **GoCB**.

The service parameter **GoCBReference** shall be **LDName/LLN0.GoCBName**.

15.2.2.6.1.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to filter the respective instances of attributes of a **GoCB**.

The service parameter **FunctionalConstraint** shall be **GO** (goose control).

15.2.2.6.1.3 GoEnable [0..1]

The parameter **GoEnable** shall contain the value for the corresponding attribute **GoEna** of the referenced **GoCB**.

15.2.2.6.1.4 ApplicationID [0..1]

The parameter **ApplicationID** shall contain the value for the corresponding attribute **AppID** of the referenced **GoCB**.

15.2.2.6.1.5 DataSetReference [0..1]

The parameter **DataSetReference** shall contain the value for the corresponding attribute **DatSet** of the referenced **GoCB**.

15.2.2.6.2 Response+

The parameter **Response+** shall indicate that the service request succeeded.

15.2.2.6.3 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

This service shall return a failure if the service has been issued for any attribute of a **GoCB** other than **GoEnable** while **GoCB** is enabled.

15.2.3 Generic object oriented substation event (GOOSE) message

15.2.3.1 GOOSE message syntax

The abstract GOOSE message format shall specify the information to be included in the GOOSE message. The structure of the GOOSE message shall be as specified in Table 29.

A GOOSE message shall at least be sent each time when a value from one or more members referenced by the **DATA-SET** change.

Table 29 – GOOSE message definition

GOOSE message		
Parameter name	**Parameter type**	**Value/value range/explanation**
DatSet	ObjectReference	Value from the instance of GoCB
AppID	VISIBLE STRING65	Value from the instance of GoCB
T	EntryTime	
StNum	INT32U	
SqNum	INT32U	
Test	BOOLEAN	(TRUE) test | (FALSE) no-test
ConfRev	INT32U	Value from the instance of GoCB
NdsCom	BOOLEAN	Value from the instance of GoCB
GOOSEData [1..n]		
Value	(*)	(*) type depends on the common data classes defined in IEC 61850-7-3. The parameter shall be derived from GOOSE control

15.2.3.2 DatSet – data set

The parameter **DatSet** shall contain the ObjectReference of the **DATA-SET** (taken from the **GoCB**) whose values of the members shall be transmitted.

15.2.3.3 AppID – application identifier

The parameter **AppID** shall contain the identifier of the **LOGICAL-DEVICE** (taken from the **GoCB**) in which the **GoCB** is located.

15.2.3.4 T – time stamp

The parameter **T** shall contain the time at which the attribute **StNum** was incremented.

15.2.3.5 StNum – state number

The parameter **StNum** shall contain the counter that increments each time a GOOSE message has been sent and a value change has been detected within the **DATA-SET** specified by **DatSet**.

The initial value for **StNum** shall be 1. The value of 0 shall be reserved.

15.2.3.6 SqNum – sequence number

The parameter **SqNum** shall contain the counter that shall increment each time a GOOSE message has been sent.

The initial value for **SqNum** shall be 1. The value of 0 shall be reserved.

15.2.3.7 Test – test

The parameter **Test** shall indicate with the value of TRUE that the values of the message shall not be used for operational purposes.

15.2.3.8 ConfRev – configuration revision

The parameter **ConfRev** (taken from the **GoCB**) shall contain the count of the number of times that the configuration of the **DATA-SET** referenced by **DatSet** has been changed.

15.2.3.9 NdsCom – needs commissioning

The parameter **NdsCom** shall contain the attribute **NdsCom** (taken from the **GoCB**) of the **GoCB**.

15.2.3.10 GOOSEData [1..n]

The parameter **GOOSEData** shall contain the user-defined information (of the members of **DATA-SET**) to be included in a GOOSE message.

The parameter **Value** shall contain the value of a member of the **DATA-SET** referenced in the **GoCB**.

15.3 Generic substation state event (GSSE) control block (GsCB)

15.3.1 GsCB class definition

The specifics for the **GsCB** model (compared to the **GoCB** model) is depicted in the shadowed area in Figure 27.

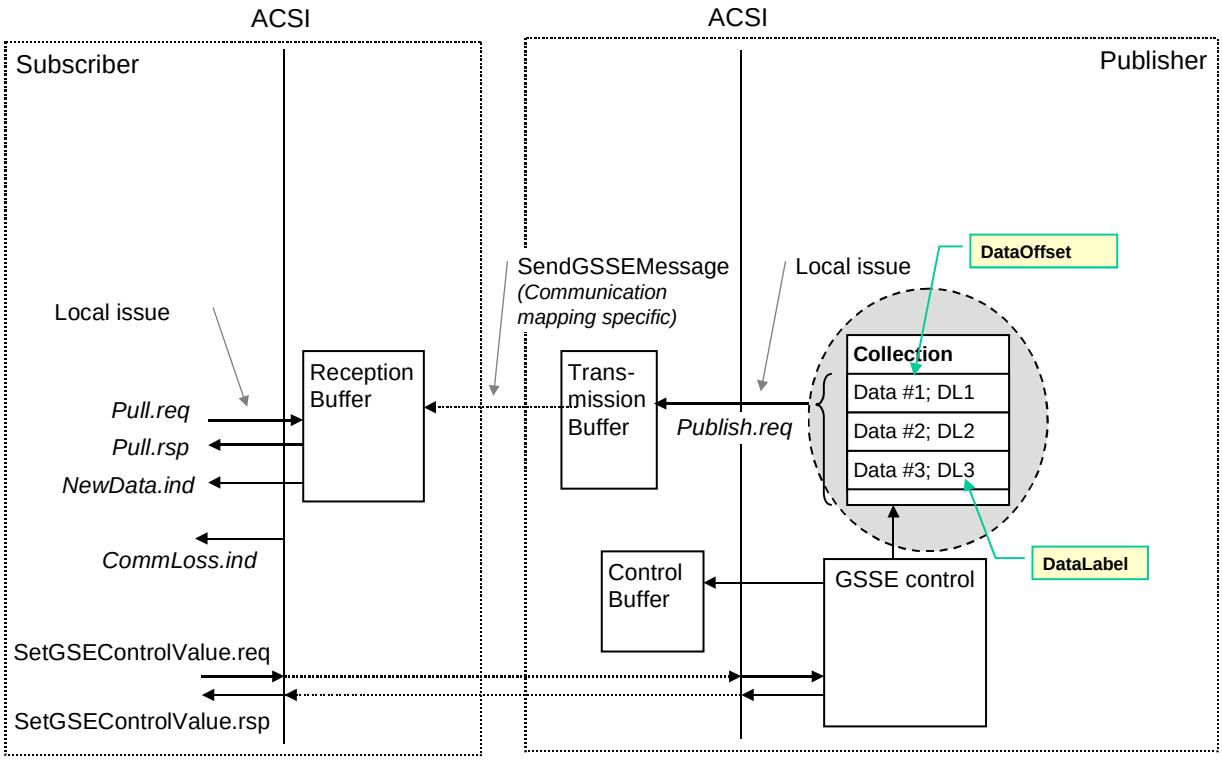


Figure 27 – Specifics for GsCB model

The information to be sent shall be a **Collection** of data. The data shall be uniquely numbered from 1 to higher numbers. Each data shall have a **DataLabel**.

The **GsCB** shall be as defined in Table 30.

Table 30 – GSSE control block class definition

GsCB class			
Attribute name	**Attribute type**	**FC**	**Value/value range/explanation**
GsCBName	ObjectName		Instance name of an instance of GsCB
GsCBRef	ObjectReference		Path-name of an instance of GsCB
GsEna	BOOLEAN	GS	Enabled (TRUE) | disabled (FALSE)
AppID	VISIBLE STRING65	GS	
DataLabel [1..n]	VISIBLE STRING65	GS	
LSentData [1..n]	GSSEData	GS	Derived from GSSE message
Services SendGSSEMessage GetReference GetGSSEDataOffset GetGsCBValues SetGsCBValues			

15.3.2 Generic substation state event (GSSE) control block class attributes

15.3.2.1 GsCBName – GSSE control name

The attribute **GsCBName** shall unambiguously identify a **GsCB** within the scope of a **LLN0**.

15.3.2.2 GsCBRef – GSSE control reference

The attribute **GsCBRef** shall be the unique path-name of a **GsCB** within a **LLN0**.

The ObjectReference **GsCBRef** shall be:

LDName/LLN0.GsCBName

15.3.2.3 GsEna – GSSE enable

The attribute **GsEna** (if set to TRUE) shall indicate that **GsCB** is currently enabled to send values of the **GsCB**. If set to FALSE the **GsCB** shall stop sending GSSE messages.

While being TRUE (**GsCB** enabled), no changes of attribute values of the **GsCB** other than disabling shall be allowed.

If the **TWO-PARTY-APPLICATION-ASSOCIATION** to the client that has enabled the **GsCB** is lost, the instance of **GsCB** shall set the attribute to FALSE.

15.3.2.4 AppID – application identification

The attribute **AppID** shall be a visible string that represents a **LOGICAL-DEVICE** in which the **GsCB** is located. The default value of **AppID** shall be that of the ObjectReference of a **GsCB**. However, the value may be set to another value as part of a system wide configuration.

NOTE Depending upon the SCSM and actual implementation, it may not be possible to uniquely identify the GSSE control through the control reference. Therefore, a standardized control attribute must be provided to allow the system configuration process to be able to uniquely identify the control within the scope of the substation.

15.3.2.5 DataLabel [1..n]

The attribute **DataLabel** of visible strings shall contain a reference for each entry used within the attribute **LastSentData**. A NULL value shall indicate that that particular **LastSentData** data entry is not in use. The DEFAULT value is a local issue.

The visible string shall hold the value of the ObjectReference if the corresponding element is being sent. Otherwise the value of the ObjectReference shall be NULL. The DEFAULT value shall be GsCBName.

NOTE The attribute **DataLabel** allows a user to assign a system unique identifier for the application that is issuing the GSSE.

15.3.2.6 LSentData [1..n] – last sent data values

The attribute **LSentData** shall represent the data values that have been sent with the last GSSE message.

The minimum maximum for the number of data values shall be 24; i.e. the attribute **LSentData** shall be capable of holding at least 24 double-bit status values).

NOTE The maximum number of data values may be constrained by the SCSM and local means.

15.3.3 GSSE service definitions

15.3.3.1 Overview

For the **GsCB** the following services are defined:

Service	Description
SendGSSEMessage	Send GSSE message
GetReference	Retrieve the **DataLabel** of a specific value associated with the GSSE message
GetGSSEElementNumber	Retrieve the position of the specific value associated with the GSSE message of a **DataLabel**
GetGsCBValues	Retrieve the attributes of a **GsCB**
SetGsCBValues	Write the attributes of a **GsCB**

15.3.3.2 SendGSSEMessage

15.3.3.2.1 SendGSSEMessage parameter table

The **SendGSSEMessage** service shall be used by a **GsCB** to send a GSSE message over a MULTICAST-APPLICATION-ASSOCIATION.

Parameter name
Request
GSSE message

15.3.3.2.2 Request

15.3.3.2.2.1 GSSE message

The parameter **GSSE message** shall specify the GSSE message as defined in 15.3.4 of the given **GsCB**.

15.3.3.3 GetReference

15.3.3.3.1 GetReference parameter table

A client shall use the **GetReference** service to retrieve the **DataLabels** of specific members of the **Collection** of the referenced **GsCB**.

Parameter name
Request
GsCBReference
DataOffset [1..n]
Response+
GsCBReference
DataLabel [1..n]
Response–
ServiceError

15.3.3.3.2 Request

15.3.3.3.2.1 GsCBReference

The parameter **GsCBReference** shall identify the attribute **GsCBRef** of the **GsCB** for which **DataLabels** are being requested.

15.3.3.3.2.2 DataOffset [1..n]

The parameter **DataOffset** shall contain a number identifying a member of the **Collection**.

15.3.3.3.3 Response+

15.3.3.3.3.1 GsCBReference

The parameter **GsCBReference** shall contain the parameter that identifies the attribute **GoCBRef** of the **GsCB** for which **DataLabels** are returned.

15.3.3.3.3.2 DataLabel [1..n]

The parameter **DataLabel** shall contain the **DataLabel** requested for the **DataOffset** of the **Collection**. A value of NULL shall indicate that no member is defined for the member being requested with the respective **DataOffset**.

15.3.3.3.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

15.3.3.4 GetGSSEDataOffset

15.3.3.4.1 GetGSSEDataOffset parameter table

A client shall use the **GetGSSEDataOffset** service to retrieve the data position of a selected data in the **Collection** associated with a **GsCB**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
GsCBReference
DataLabel [1..n]
Response+
GsCBReference
DataOffset [1..n]
Response–
ServiceError

15.3.3.4.2 Request

15.3.3.4.2.1 GsCBReference

The parameter **GsCBReference** shall identify the attribute **GsCBRef** of the **GsCB** for which **MemberOffset** are being requested.

15.3.3.4.2.2 DataLabel [1..n]

The parameter **DataLabel** shall contain the **DataLabel** for which the **DataOffset** of the **Collection** is requested.

15.3.3.4.3 Response+

15.3.3.4.3.1 GsCBReference

The parameter **GsCBReference** shall contain the parameter that identifies the attribute **GoCBRef** of the **GsCB** for which **DataLabels** are returned.

15.3.3.4.3.2 DataOffset [1..n]

The parameter **DataOffset** shall contain a number identifying a member of the **Collection**. A value of NULL shall indicate that no **DataOffset** is defined for the member being requested with the respective **DataLabel**.

15.3.3.4.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

15.3.3.5 GetGsCBValues

A client shall use the **GetGsCBValues** service to retrieve attribute values of **GsCB** made visible and thus accessible to the requesting client by the referenced **LLN0**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
GsCBReference
FunctionalConstraint
Response+
GsEnable
ApplicationID
DataLabel [1..n]
LastSentData [1..n]
Response–
ServiceError

15.3.3.5.1 Request

15.3.3.5.1.1 GsCBReference

The parameter **GsCBReference** shall specify the ObjectReference of the **GsCB**.

The service parameter **GsCBReference** shall be **LDName/LLN0.GsCBName**.

15.3.3.5.1.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to filter the respective instances of attributes of a **GsCB**.

The service parameter **FunctionalConstraint** shall be **GS** (gsse control).

15.3.3.5.2 Response+

The parameter **Response+** shall indicate that the service request succeeded.

15.3.3.5.2.1 GsEnable

The parameter **GsEnable** shall contain the value of the corresponding attribute **GsEna** of the referenced **GsCB**.

15.3.3.5.2.2 ApplicationID

The parameter **ApplicationID** shall contain the value of the corresponding attribute **AppID** of the referenced **GsCB**.

15.3.3.5.2.3 DataLabel [1..n]

The parameter **DataLabel** shall contain the **DataLabel** of the **Collection**.

15.3.3.5.2.4 LastSentData [1..n]

The parameter **LastSentData** shall contain the value of the attribute **LSentData** of the **GsCB**.

15.3.3.5.3 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

15.3.3.6 SetGsCBValues

A client shall use the **SetGsCBValues** service to set attribute values of **GsCB** made visible and thus accessible to the requesting client by the referenced **LLN0**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
GsCBReference
FunctionalConstraint
GsEnable [0..1]
ApplicationID [0..1]
Response+
Response–
ServiceError

15.3.3.6.1 Request

15.3.3.6.1.1 GsCBReference

The parameter **GsCBReference** shall specify the ObjectReference of the **GsCB**.

The service parameter **GsCBReference** shall be **LDName/LLN0.GsCBName**.

15.3.3.6.1.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to filter the respective instances of attributes of a **GsCB**.

The service parameter **FunctionalConstraint** shall be **GS** (gsse control).

15.3.3.6.1.3 GsEnable [0..1]

The parameter **GsEnable** shall contain the value for the corresponding attribute **GsEna** of the referenced **GsCB**.

15.3.3.6.1.4 ApplicationID [0..1]

The parameter **ApplicationID** shall contain the value for the corresponding attribute **AppID** of the referenced **GsCB**.

15.3.3.6.2 Response+

The parameter **Response+** shall indicate that the service request succeeded.

15.3.3.6.3 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

This service shall return a failure if the service has been issued for any attribute of a **GsCB** other than **GsEnable** while **GsCB** is enabled.

15.3.4 Generic substation state event (GSSE) message

15.3.4.1 Syntax

The abstract GSSE message format shall specify the information to be included in the GSSE message. The structure of the GSSE message shall be as specified in Table 31.

A GSSE message shall at least be sent each time when a value from one or more of the **LSentData** change (for example, a change of status value is detected).

Table 31 – GSSE message definition

GSSE message		
Parameter name	**Parameter type**	**Value/value range/explanation**
AppID	VISIBLE STRING65	Value from the instance of GsCB
T	EntryTime	
SqNum	INT32U	
StNum	INT32U	
Test	BOOLEAN	(TRUE) test | (FALSE) no-test
PhsID	INT16U	
GSSEData [1..n]		
Value	CODED ENUM	Invalid or transient (0) | false or closed (1) | true or open (2) | invalid (3)

15.3.4.2 AppID – application identifier

The parameter **AppID** shall contain the identifier **LOGICAL-DEVICE** (taken from the **GsCB**) in which the **GsCB** is located.

15.3.4.3 T – time stamp

The parameter **T** shall contain the time at which the **StNum** attribute was incremented.

15.3.4.4 SqNum – sequence number

The parameter **SqNum** shall contain the counter that shall increment each time a GSSE message has been sent.

The initial value for **STNum** shall be 1. The value of 0 shall be reserved.

15.3.4.5 StNum – state number

The parameter **StNum** shall contain a counter that increments each time a GSSE message has been sent and a value change has been detected within the data values of **LSentData**.

The initial value for **StNum** shall be 1. The value of 0 shall be reserved.

15.3.4.6 Test – test

The parameter **Test** shall indicate with the value of TRUE that the values of the message shall not be used for operational purposes.

15.3.4.7 PhsID – phase identification

The parameter PhsID shall indicate faulted phases.

15.3.4.8 GSSEData [1..n]

The parameter **GSSEData** shall be a status value of 4 values coded as CODED ENUM. The defined values are invalid or transient (0) | false or closed (1) | true or open (2) | invalid (3).

The size of the array [1..n] is determined by the size of the LSentData attribute of the associated **GsCB**.

16 Transmission of sampled value class model

16.1 Overview

The transmission of sampled values requires special attention with regard to the time constraints. The model provides transmission of sampled values in an organized and time-controlled way so that the combined jitter of sampling and transmission is minimized to a degree that an unambiguous allocation of the samples, times, and sequence is provided.

The model applies to the exchange of values of a **DATA-SET**. The **DATA** of the **DATA-SET** are of the common data class SAV (sampled analogue value as defined in IEC 61850-7-3). A buffer structure shall be defined for the transmission of the sampled values.

The information exchange shall be based on a publisher/subscriber mechanism. The publisher shall write the values in a local buffer at the sending side; the subscriber shall read the values from a local buffer at the receiving side. A time stamp shall be added to the values, so that the subscriber can check the timeliness of the values. The communication system shall be responsible to update the local buffers of the subscribers. A sampled value control (**SVC**) in the publisher shall be used to control the communication procedure.

Figure 28 gives an overview on the classes and services of the model.

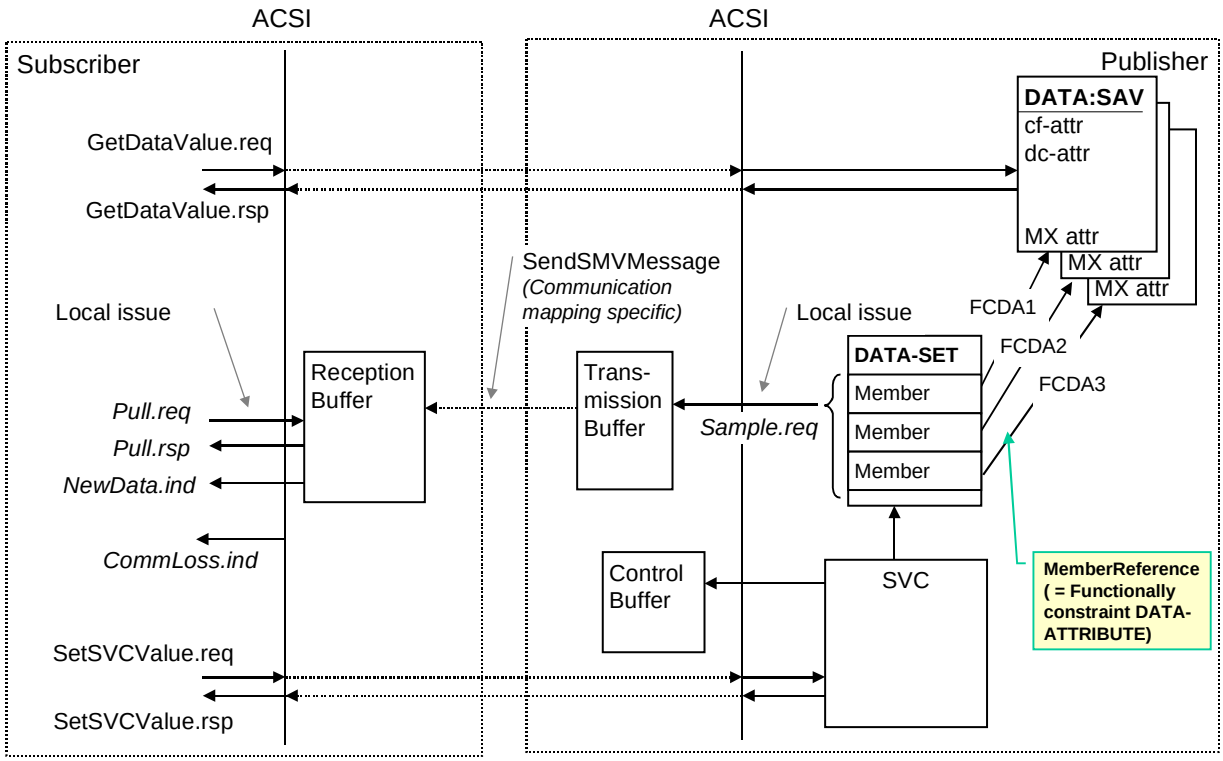


Figure 28 – Model for transmission of sampled values

There shall be two methods to exchange sampled values between a publisher and one or more subscriber. One method shall use the **MULTICAST-APPLICATION-ASSOCIATION** (multicast sampled value control, MSVCB), the other method shall use the **TWO-PARTY-APPLICATION-ASSOCIATION** (unicast sampled value control, USVCB).

The producer shall sample the inputs with the specified sample rate. The synchronization of this sampling may be done internal or over the network. The samples shall be posted in the transmission buffer.

The network embedded scheduler shall send the content of the buffer over the network to the subscribers. The rate may be a mapping specific parameter. Then the samples shall be placed into the receive buffers of the subscribers. The arrival of a new series of samples in the receive buffer shall be signalled to the application.

The model shall provide mechanisms that the subscriber can detect lost samples. If samples are not be transmitted due to problems in the communication network, the publisher shall delete these samples.

16.2 Transmission of sampled values using multicast

The transmission of sampled values using multicast (**MULTICAST-SAMPLE-VALUE-CONTROL-BLOCK** – **MSVCB**) shall be based on configured configuration in the producer device. The data exchange shall be based on the multicast application association. To support self-descriptive capabilities, any client may read the attributes of the sampled value control instance. Authorized clients may modify attributes of the sampled value control.

16.2.1 MSVCB class definition

The **MSVCB** shall be as defined in Table 32.

Table 32 – MSVCB class definition

MSVCB class				
Attribute name	**Attribute type**	**FC**	**TrgOp**	**Value/value range/explanation**
MsvCBName	ObjectName	-	-	Instance name of an instance of MSVCB
MsvCBRef	ObjectReference	-	-	Path-name of an instance of MSVCB
SvEna	BOOLEAN	MS	dchg	Enabled (TRUE) | disabled (FALSE), DEFAULT FALSE
MsvID	VISIBLE STRING65	MS	-	
DatSet	ObjectReference	MS	dchg	
ConfRev	INT32U	MS	dchg	
SmpRate	INT16U	MS	-	(0..MAX)
OptFlds	PACKED LIST	MS	dchg	
refresh-time	BOOLEAN			
sample-synchronized	BOOLEAN			
sample-rate	BOOLEAN			
Services SendMSVMessage GetMSVCBValues SetMSVCBValues				

16.2.1.1 MsvCBName – multicast sampled value control name

The attribute **MSVCBNam** shall unambiguously identify a **MSVCB** within the scope of an **LLN0**.

16.2.1.2 MsvCBRef – multicast sampled value control reference

The attribute **MSVCBRef** shall be the unique path-name of a **MSVCB** within an **LLN0**.

The ObjectReference **MSVCBRef** shall be:

LDName/LLN0.MSVCBName

16.2.1.3 SvEna – sampled value enable

The attribute **SvEna** (if set to TRUE) shall indicate that the **MSVCB** is currently enabled to send values of the **MSVCB**. If set to FALSE the **MSVCB** shall stop sending values.

While being TRUE (**MSVCB** enabled), no changes of attribute values of the **MSVCB** other than disabling shall be allowed.

16.2.1.4 MsvID – multicast sampled value identifier

The attribute **MSVID** shall be a unique identification of the sampled value buffer related to the update of the sampled values.

16.2.1.5 DatSet

The attribute **DatSet** shall specify the reference of the **DATA-SET** whose values of members are to be transmitted in the **MSVCB** message.

16.2.1.6 ConfRev – configuration revision

The parameter **ConfRev** shall contain a count of the number of times that the configuration with regard to the **MSVCB** has been changed. Changes that shall be counted are:

– any deletion of a member of the **DATA-SET**,

– any reordering of members of the **DATA-SET**,

– any change of a value of the **DataAttribute** of the **DATA-SET** whose functional constraint equals CF,

– any change of a value of an attribute of **MSVCB** (functional constraint of attribute **MSVCB** equals **MS** (multicast sampled value control).

The counter shall be incremented when the configuration changes.

The initial value for **ConfRev** is outside the scope of this standard. The value of 0 shall be reserved. A restart of the IED shall not reset the value.

NOTE Configuration changes of **DATA-SETs** due to processing of services are not allowed (see **DATA-SET** model). Changes to be taken into account for the ConfRev are those made by local means like system configuration.

16.2.1.7 SmpRate

The attribute **SmpRate** shall specify the sample rate in units of samples per nominal period.

16.2.1.8 OptFlds – optional fields to include in SV message

The attribute **OptFlds** shall be the client-specified optional fields to be included in the **SV** message issued by this **MSVCB**. This attribute defines a subset of the optional header fields that shall be included in the **SV** message:

– RefrTm (Refresh time, time of refresh activity)

– SmpSynch (Samples synchronized, samples are synchronized by clock signals) , and

– SmpRate (sample rate from the instance of MSVCB)

16.2.2 Multicast sampled value class services

16.2.2.1 Overview

For the **MSVCB** the following services are defined:

Service	Description
SendMSVMessage	Send MSV message
GetMSVCBValues	Retrieve the attributes of an **MSVCB**
SetMSVCBValues	Write the attributes of an **MSVCB**

16.2.2.2 SendMSVMessage

16.2.2.2.1 SendMSVMessage parameter table

The **SendMSVMessage** service shall be used by an **MSVCB** to send sampled values from the server to the client over a MULTICAST-APPLICATION-ASSOCIATION.

Parameter name
Request
MSV message

16.2.2.2.2 Request

16.2.2.2.2.1 MSV message

The parameter **MSV message** shall specify the values of the membes of the referenced **DATA-SET** of the **MSVCB** as specified in the abstract sampled value format definition (see 16.4). The concrete format of the **MSV message** shall be defined in the SCSM.

16.2.2.3 GetMSVCBValues

A client shall use the **GetMSVCBValues** service to retrieve attribute values of **MSVCBB** made visible and thus accessible to the requesting client by the referenced **LLN0**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
MsvCBReference
FunctionalConstraint
Response+
SvEnable
MulticastSampleValueID
DataSetReference
ConfigurationRevision
SampleRate
Response–
ServiceError

16.2.2.3.1 Request

16.2.2.3.1.1 MsvCBReference

The parameter **MsvCBReference** shall specify the ObjectReference of the **MSVCB**.

The service parameter **MsvCBReference** shall be **LDName/LLN0.MsvCBName**.

16.2.2.3.1.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to filter the respective instances of attributes of a **MSVCB**.

The service parameter **FunctionalConstraint** shall be **MS** (multicast sampled value control).

16.2.2.3.2 Response+

The parameter **Response+** shall indicate that the service request succeeded.

16.2.2.3.2.1 SvEnable

The parameter **SvEnable** shall contain the value of the corresponding attribute **SvEna** of the referenced **MSVCB**.

16.2.2.3.2.2 MulticastSampleValueID

The parameter **MulticastSampleValueID** shall contain the value of the corresponding attribute **MsvID** of the referenced **MSVCB**.

16.2.2.3.2.3 DataSetReference

The parameter **DataSetReference** shall contain the value of the corresponding attribute **DatSet** of the referenced **MSVCB**.

16.2.2.3.2.4 ConfigurationRevision

The parameter **ConfigurationRevision** shall contain the value of the corresponding attribute **ConfRev** of the **MSVCB**.

16.2.2.3.2.5 SampleRate

The parameter **SampleRate** shall contain the value of the corresponding attribute **SmpRate** of the **MSVCB**.

16.2.2.3.3 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

16.2.2.4 SetMSVCBValues

A client shall use the **SetMSVCB Values** service to set attribute values of **MSVCB** made visible and thus accessible to the requesting client by the referenced **LLN0**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
MsvCBReference
FunctionalConstraint
SvEnable [0..1]
MulticastSampleValueID [0..1]
DataSetReference [0..1]
SampleRate [0..1]
Response+
Response–
ServiceError

16.2.2.4.1 Request

16.2.2.4.1.1 MsvCBReference

The parameter **MsvCBReference** shall specify the ObjectReference of the **MSVCB**.

The service parameter **MsvCBReference** shall be **LDName/LLN0.MsvCBName**.

16.2.2.4.1.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to filter the respective instances of attributes of a **MSVCB**.

The service parameter **FunctionalConstraint** shall be **MS** (multicast sampled value control).

16.2.2.4.1.3 SvEnable [0..1]

The parameter **SvEnable** shall contain the value for the corresponding attribute **SvEna** of the referenced **MSVCB**.

16.2.2.4.1.4 MulticastSampleValueID [0..1]

The parameter **MulticastSampleValueID** shall contain the value for the corresponding attribute **MsvID** of the referenced **MSVCB**.

16.2.2.4.1.5 DataSetReference [0..1]

The parameter **DataSetReference** shall contain the value for the corresponding attribute **DatSet** of the referenced **MSVCB**.

16.2.2.4.1.6 SampleRate [0..1]

The parameter **SampleRate** shall contain the value for the corresponding attribute **SmpRate** of the **MSVCB**.

16.2.2.4.2 Response+

The parameter **Response+** shall indicate that the service request succeeded.

16.2.2.4.3 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

This service shall return a failure if the service has been issued for any attribute of a **MSVCB** other than **SvEnable** while **MSVCB** is enabled.

16.3 Transmission of sampled values using unicast

The transmission of sampled values using unicast (**UNICAST-SAMPLE-VALUE-CONTROL-BLOCK** – **USVCB**) shall be based on two-party application associations. The subscriber shall establish the association with the producer. The subscriber may then configure the class and enable the transmission of the sampled values with the attribute **SvEna**. When the association is released, the transmission of the sampled values shall stop and the instance of the control class shall be released.

The samples shall be sent using the two-party application association.

16.3.1 USVCB class definition

The **USVCB** shall be as defined in Table 33.

Table 33 – USVCB class definition

USVCB class				
Attribute name	**Attribute type**	**FC**	**TrgOp**	**Value/value range/explanation**
UsvCBNam	ObjectName	-	-	Instance name of an instance of UNICAST-SVC
UsvCBRef	ObjectReference	-	-	Path-name of an instance of UNIICAST-SVC
SvEna	BOOLEAN	US	dchg	Enabled (TRUE) | disabled (FALSE), DEFAULT FALSE
Resv	BOOLEAN	US	-	
UsvID	VISIBLE STRING65	US	-	
DatSet	ObjectReference	US	dchg	
ConfRev	INT32U	US	dchg	
SmpRate	INT16U	US	dchg	(0..MAX)
OptFlds	PACKED LIST	US	dchg	
refresh-time	BOOLEAN			
sample-synchronized	BOOLEAN			
sample-rate	BOOLEAN			
Services SendUSVMessage GetUSVCBValues SetUSVCBValues				

16.3.1.1 UsvCBName – unicast sampled value control name

The attribute **USVCBNam** shall unambiguously identify a **USVCB** within the scope of a **LLN0**.

16.3.1.2 UsvCBRef – unicast sampled value control reference

The attribute **USVCBRef** shall be the unique path-name of a **USVCB** within a **LLN0**.

The ObjectReference **USVCBRef** shall be:

LDName/LLN0.UsvCBName

16.3.1.3 SvEna – sampled value enable

The attribute **SvEna** (if set to TRUE) shall indicate that the **USVCB** is currently enabled to send values of the **USVCB**. If set to FALSE the **USVCB** shall stop issuing reports.

While being TRUE (**USVCB** enabled), no changes of attribute values of **USVCB** other than disabling shall be allowed.

If the **TWO-PARTY-APPLICATION-ASSOCIATION** to the client that has enabled the **USVCB** is lost, the **USVCB** shall set the attribute to FALSE.

16.3.1.4 Resv – reserve USVCB

The attribute **Resv** (if set to TRUE) shall indicate that the **USVCB** is currently exclusively reserved for the client that has set the value to TRUE. Other clients shall not be allowed to a set any attribute of that **USVCB**.

If the **TWO-PARTY-APPLICATION-ASSOCIATION** to the client that has set this attribute to TRUE is lost, the **USVCB** shall set the attribute to FALSE.

NOTE The attribute **Resv** functions as a semaphore for the configuration, enabling and disabling of the **USVCB**.

16.3.1.5 UsvID

The attribute **USVID** shall be a unique identification of the sampled value buffer related to the update of the sampled values.

16.3.1.6 DatSet

The attribute **DatRef** shall specify the reference of the **DATA-SET** whose values of members are to be transmitted in the **USVCB** message.

16.3.1.7 ConfRev – configuration revision

The parameter **ConfRev** shall contain a count of the number of times that the configuration with regard to the **USVCB** has been changed. Changes that shall be counted are:

– any deletion of a member of the **DATA-SET**,

– any reordering of members of the **DATA-SET**,

– any change of a value of the **DataAttribute** of the **DATA-SET** whose functional constraint equals cf.

– any change of a value of an attribute of **USVCB** (functional constraint of attribute **USVCB** equals **US**).

The counter shall be incremented when the configuration changes.

The initial value for **ConfRev** is outside the scope of this standard. The value of 0 shall be reserved. A restart of the IED shall not reset the value.

NOTE Configuration changes of **DATA-SETs** due to processing of services are not allowed (see **DATA-SET** model). Changes to be taken into account for the ConfRev are those made by local means like system configuration.

16.3.1.8 SmpRate

The attribute **SmpRate** shall specify the sample rate in units of samples per nominal period.

16.3.1.9 OptFlds – optional fields to include in SV message

The attribute **OptFlds** shall be the client-specified optional fields to be included in the **SV** message issued by this **USVCB**. This attribute defines a subset of the optional header fields that shall be included in the **SV** message:

– RefrTm (Refresh time, time of refresh activity)

– SmpSynch (Samples synchronized, samples are synchronized by clock signals), and

– SmpRate (sample rate from the instance of USVCB).

16.3.2 Unicast sampled value services

16.3.2.1 Overview

For the **USVCB** the following services are defined:

Service	Description
SendUSVMessage	Send USV message
GetUSVCBValues	Retrieve the attributes of a **USVCB**
SetUSVCBValues	Write the attributes of a **USVCB**

16.3.2.2 SendUSVMessage

16.3.2.2.1 SendUSVMessage parameter table

The **SendUSVCMessage** service shall be used by a **UMVC** to send sampled values from the server to the client over a **TWO-PARTY-APPLICATION-ASSOCIATION**.

Parameter name
Request
USV message

16.3.2.2.2 Request

USV message

The parameter **USV message** shall specify the values of the members of the referenced **DATA-SET** of the **USVCB** as specified in the abstract sampled value format definition (see 16.4). The concrete format of the **USV message** shall be defined in the SCSM.

16.3.2.3 GetUSVCBValues

A client shall use the **GetUSVCBValues** service to retrieve attribute values of **USVCB** made visible and thus accessible to the requesting client by the referenced **LLN0**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
UsvCBReference
FunctionalConstraint
Response+
SvEnable
CBReserved
UnicastSampleValueID
DataSetReference
ConfigurationRevision
SampleRate
Response–
ServiceError

16.3.2.3.1 Request

16.3.2.3.1.1 UsvCBReference

The parameter **UsvCBReference** shall specify the ObjectReference of the **USVCB**.

The service parameter **UsvCBReference** shall be **LDName/LLN0.UsvCBName**.

16.3.2.3.1.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to filter the respective instances of attributes of a **USVCB**.

The service parameter **FunctionalConstraint** shall be **US** (unicast sampled value control).

16.3.2.3.2 Response+

The parameter **Response+** shall indicate that the service request succeeded.

16.3.2.3.2.1 SvEnable

The parameter **SvEnable** shall contain the value of the corresponding attribute **SvEna** of the referenced **MSVCB**.

16.3.2.3.2.2 CBReserved

The parameter **CBReserved** shall contain the value of the corresponding attribute **Resv** of the referenced **MSVCB**.

16.3.2.3.2.3 UnicastSampleValueID

The parameter **UnicastSampleValueID** shall contain the value of the corresponding attribute **UsvID** of the referenced **USVCB**.

16.3.2.3.2.4 DataSetReference

The parameter **DataSetReference** shall contain the value of the corresponding attribute **DatSet** of the referenced **USVCB**.

16.3.2.3.2.5 ConfigurationRevision

The parameter **ConfigurationRevision** shall contain the value of the corresponding attribute **ConfRev** of the **USVCB**.

16.3.2.3.2.6 SampleRate

The parameter **SampleRate** shall contain the value of the corresponding attribute **SmpRate** of the **USVCB**.

16.3.2.3.3 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

16.3.2.4 SetUSVCBValues

A client shall use the **SetUSVCBValues** service to set attribute values of **USVCB** made visible and thus accessible to the requesting client by the referenced **LLN0**.

NOTE The visible instances are those that are defined within a given view (see Clause 7 for details on the view concept).

Parameter name
Request
UsvCBReference
FunctionalConstraint
SvEnable [0..1]
CBReserved [0..1]
UnicastSampleValueID [0..1]
DataSetReference [0..1]
SampleRate [0..1]
Response+
Response–
ServiceError

16.3.2.4.1 Request

16.3.2.4.1.1 UsvCBReference

The parameter **UsvCBReference** shall specify the ObjectReference of the **USVCB**.

The service parameter **MsvCBReference** shall be **LDName/LLN0.UsvCBName**.

16.3.2.4.1.2 FunctionalConstraint

The parameter **FunctionalConstraint** shall contain the value of the functional constraint parameter to filter the respective instances of attributes of a **USVCB**.

The service parameter **FunctionalConstraint** shall be **US** (unicast sampled value control).

16.3.2.4.1.3 SvEnable [0..1]

The parameter **SvEnable** shall contain the value for the corresponding attribute **SvEna** of the referenced **USVCB**.

16.3.2.4.1.4 CBReserved

The parameter **CBReserved** shall contain the value for the corresponding attribute **Resv** of the referenced **MSVCB**.

16.3.2.4.1.5 UnicastSampleValueID [0..1]

The parameter **UnicastSampleValueID** shall contain the value for the corresponding attribute **UsvID** of the referenced **USVCB**.

16.3.2.4.1.6 DataSetReference [0..1]

The parameter **DataSetReference** shall contain the value for the corresponding attribute **DatSet** of the referenced **USVCB**.

16.3.2.4.1.7 SampleRate [0..1]

The parameter **SampleRate** shall contain the value for the corresponding attribute **SmpRate** of the **USVCB**.

16.3.2.4.2 Response+

The parameter **Response+** shall indicate that the service request succeeded.

16.3.2.4.3 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

This service shall return a failure if the service has been issued for any attribute of a **USVCB** other than **SvEnable** while **USVCB** is enabled.

16.4 Sampled value format

The abstract sampled value format used for the sampled value message shall be as follows:

Table 34 – Sampled value (SV) format definition

Sampled value format		
Parameter name	**Parameter type**	**Value/value range/explanation**
MsvID or UsvID	VISIBLE STRING65	Value from the MSVCB or USVCB
OptFlds	[a]	Optional fields to be included in the SV message
DatSet	ObjectReference	Value from the MSVCB or USVCB
Sample [1..n]		
Value	(*)	(*) The value of the member of the instance of the DATA-SET. Type of the common data classes is SAV (sampled analogue value) as defined in IEC 61850-7-3
SmpCnt	INT16U	Sample counter
RefrTm	EntryTime	OPTIONAL; time of refresh activity
ConfRev	INT32U	Configuration revision number from the instance of MSVCB or USVCB
SmpSynch	BOOLEAN	Samples are synchronized by clock signals
SmpRate	INT16U	OPTIONAL; sample rate from the instance of MSVCB or USVCB

[a] The type and value of this parameter shall be derived from the attribute OptFlds of the respective **USVCB** or **MSVCB**.

16.4.1 MsvID or UsvID

The parameter **MsvID** or **UsvID** shall contain the values of the attributes **MsvID** or **UsvID** of the **MSVCB** or **USVCB** to be included in the sampled value message.

16.4.2 OptFlds

The parameter **OptFlds** shall specify which of the optional fields (**RefrTm**, **SmpSynch** and **SmpRate**) are included in the sampled value message. If refresh-time (sample-rate or sample-synchronized) is TRUE then the field **RefrTm** (**SmpSynch** or **SmpRate**) shall be contained in the sampled value message.

The parameter **OptFlds** shall be derived from the attribute **OptFlds** of the respective **USVCB** or **MSVCB**.

16.4.3 DatSet

The parameter **DatSet** (taken from the **MsvID** or **UsvID**) shall contain the ObjectReference of the **DATA-SET** whose values of the members are transmitted in the message.

16.4.4 Sample [1..n]

The parameter **Sample** shall contain the value of a member of **DATA-SET** sampled at a given time.

16.4.5 SmpCnt

The parameter **SmpCnt** shall contain the values of a counter, which is incremented each time a new sample of the analogue value is taken. The sample values shall be kept in the right order. If the counter is used to indicate time consistency of various sampled values, the counter shall be reset by an external synchronization event.

NOTE The external synchronization event is outside this part of the standard; details can be found in a SCSM.

16.4.6 RefrTm

The parameter **RefrTm** shall contain the time when the transmission buffer has been refreshed locally.

NOTE The semantic of the **RefrTm** is defined in the SCSM. This time may be used by the subscriber to check the validity of the data.

16.4.7 ConfRev

The parameter **ConfRev** shall contain the value of the attribute **ConfRev** of the **MSVCB** or **USVCB**.

16.4.8 SmpSynch

The parameter **SmpRate** shall indicate whether the sampled analogue values sent by the **MSVCB** or **USVCB** are synchronized by clock signals.

16.4.9 SmpRate

The parameter **SmpRate** shall contain the value of the attribute **SmpRate** of the **MSVCB** or **USVCB**.

17 CONTROL class model

17.1 Introduction

DATA related to external devices, control outputs, or other internal functions may require to be controlled by a client. The **control** model provides services to operate on **DATA** with **DataAttribute** having the functional constraint **FC** (=**CO** or **SP**). The **DATA** providing controllable **DataAttributes** shall be of one of the common **DATA** classes defined in IEC 61850-7-3, for example:

– Controllable single point (**SPC**)
– Controllable double point (**DPC**)
– Controllable integer status (**ISC**)
– Binary controlled step position information (**BSC**)
– Integer controlled step position information (**IST**)
– Controllable analogue set point (**APC**)

NOTE This clause makes use of the term “control object”. A control object can be any **DATA** based on one of the above-listed common data classes.

The control model consists of

– specification of services;
– a behaviour described with state machines.

The control model defines the following services:

– **Select** (Sel) / **SelectWithValue** (SelVal)
– **Cancel**
– **Operate** (Oper) / **TimeActivatedOperate** (TimOper)
– **CommandTermination** (CmdTerm)

NOTE The abbreviations for these services may be used in the SCSM.

The concept of the control model is depicted in the example in Figure 29.

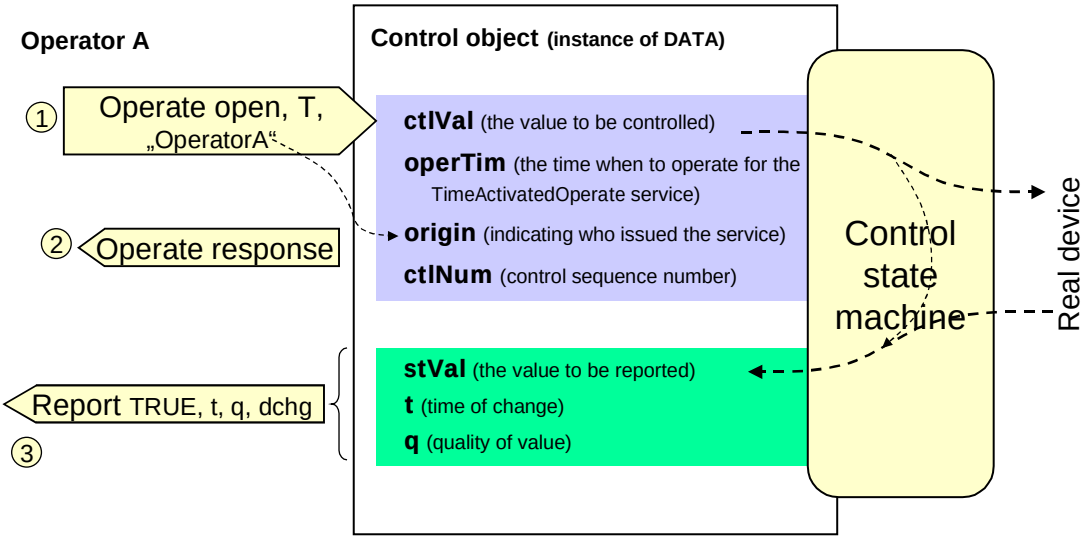


Figure 29 – Principle of the control model

The client (Operator A) issues the **Operate** service which is immediately confirmed by the **Operate** response. The new state change is reported by an independent **Report** indicating the final result of the control operation.

The services **Select**, **SelectWithValue**, **Cancel**, **Operate**, **TimeActivatedOperate**, and **CommandTermination** are related. The behaviour of these services shall be as defined in the state machines contained in this clause.

Depending on the application, different behaviours of a control object shall be used. Therefore, different state machines are defined. For a specific control object, the used model shall be defined in a configuration parameter. Four cases are defined:

Case 1: Direct control with normal security (**direct-operate**)

Case 2: SBO control with normal security (**operate-once** or **operate-many**)

Case 3: Direct control with enhanced security (**direct-operate**)

Case 4: SBO control with enhanced security (**operate-once** or **operate-many**)

As shown in the state diagrams, the change from one state to the next state shall be controlled by the parameter “check condition”. The check condition may be specified by a service parameter (for example, synchrocheck). Besides the check condition specified by the service parameter, the control object may perform additional checks.

17.2 Control with normal security

In the case of control with normal security there shall be no additional supervision of the status value by the control object. This means that for the negative case, if the status value did not change to the control value, the client will not get information about the failure from the control object.

17.2.1 Direct control with normal security

This model shall use the services **Operate** and **TimeActivatedOperate**. In addition, the change of the status of the control object may generate a report. The generation of this report is independent of the other services and therefore not included in the state machine behaviour.

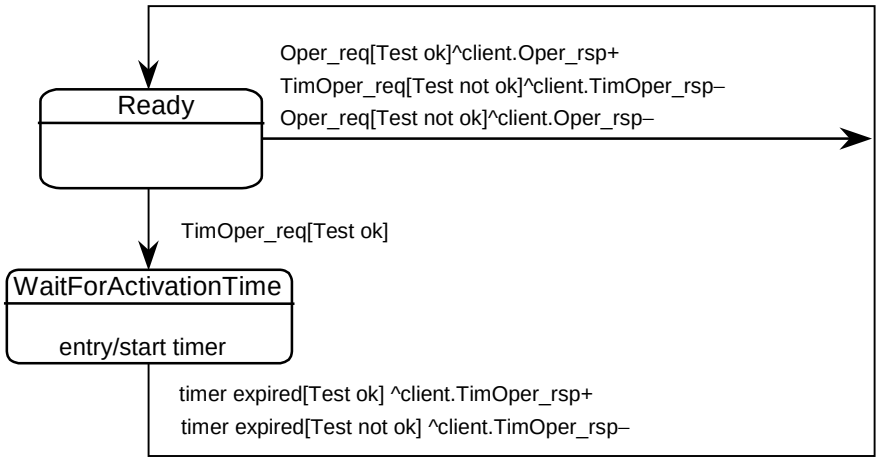


Figure 30 – State machine of direct control with normal security

Direct control with normal security should be used for operations that act either on local **DATA** (for example, a LED test) or on **DATA** that influence external devices where a return information is not supervised (for example, switch on a heating).

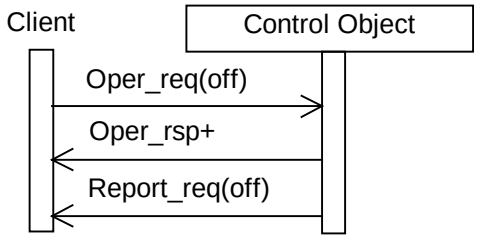


Figure 31 – Direct control with normal security

Procedure

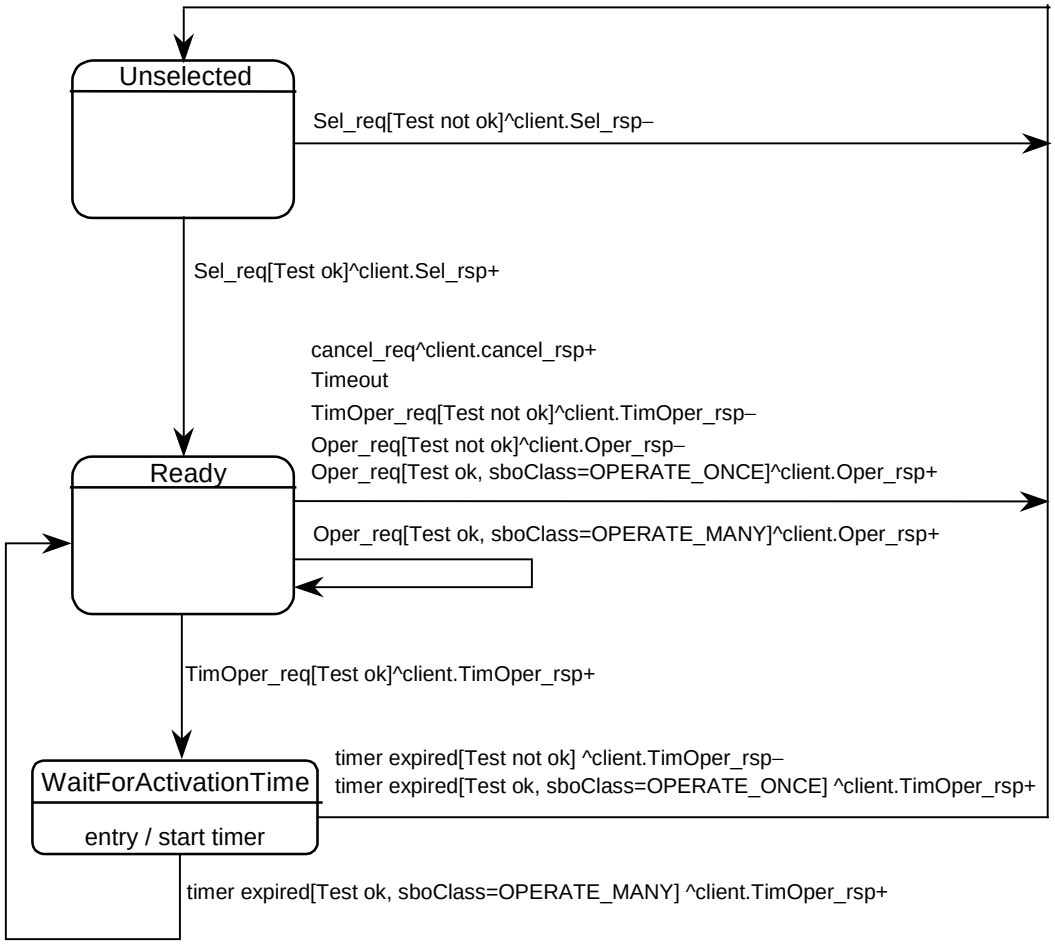
On receipt of an **Operate** request, the control object shall check validation of the control execution.

- If not successful, the control object shall issue a negative response to the requesting client.
- If successful, the control object shall issue a positive response to the requesting client and causes the requested action.

The new status may be reported by the **Report** service (see reporting model).

17.2.2 SBO control with normal security

This model shall use the services **Select**, **Cancel**, **Operate**, and **TimeActivatedOperate**. In addition, the change of the status of the control object may generate a **Report**. The generation of this **Report** is independent of the other services and therefore not included in the state machine behaviour.



NOTE This state machine is compatible to the SBO control model defined in UCA™.2.

Figure 32 – State machine of SBO control with normal security

Procedure

a) On receipt of a **Select** request, the control object shall determine if the client has appropriate access authority, that the control object is not currently selected by a different client, and that the device represented by the associated **LOGICAL-NODE** is operable and is not tagged so as to restrict operation.

   – If the **Select** operation is not valid, the control object shall issue a negative response to the requesting client.

   – If the **Select** operation is valid, the control object shall issue a positive response to the requesting client, shall change the state to ready and starts a deselect timer for either the interval defined by the SelTimOut attribute or, if unimplemented, some locally determined duration.

b) If the deselect timer expires before an **Operate** request on one or more of the other control components shall be requested by the selecting client, the control object shall change the state to unselected.

c) If an **Operate** request is received from the selecting client while the state is not Ready for that client, the operation shall be denied.

d) On receipt of an **Operate** request, the control object shall check validation of the control execution.

   – If not successful, the control object shall issue a negative response to the requesting client.
   – If successful, the control object shall issue a positive response to the requesting client and shall cause the requested action by activating a binary output (or sending an equivalent signal on a process bus). The control object shall turn to the state WaitForActivationTime.

17.3 Control with enhanced security

17.3.1 Introduction

In the case of control with enhanced security there shall be an additional supervision of the status value by the control object. Each command sequence shall be terminated by a **CommandTermination** service primitive.

17.3.2 Direct control with enhanced security

This model shall use the services **Operate**, **TimeActivatedOperate**, and **Command-Termination**. In addition, the change of the status of the control object may generate a **Report**. The generation of this **Report** is related to the other services and therefore included in the state machine behaviour.

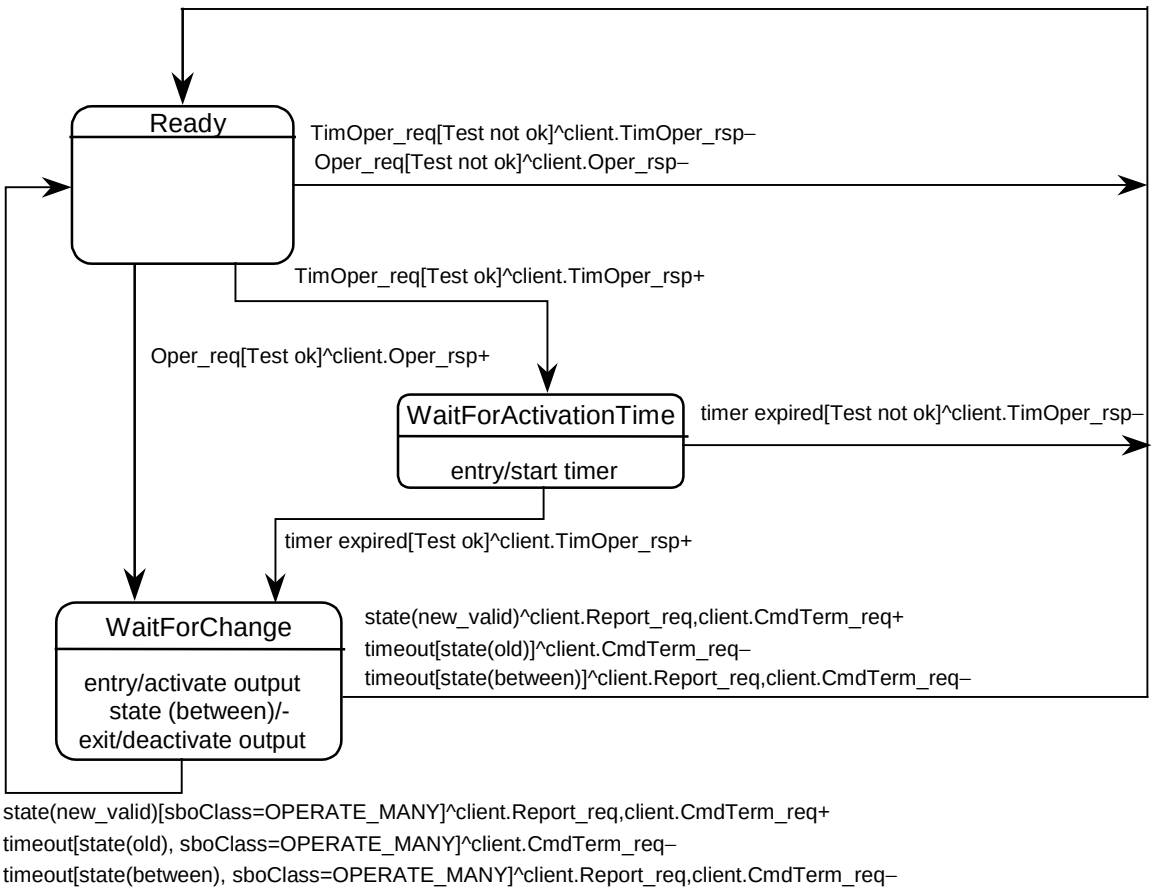


Figure 33 – State machine of direct control with enhanced security

17.3.3 SBO control with enhanced security

This model uses the services **SelectWithValue**, **Cancel**, **Operate**, **TimeActivatedOperate**, and **CommandTermination**. In addition, the change of the status of the control object may generate a **Report**. The generation of this **Report** is related to the other services and therefore included in the state machine behaviour.

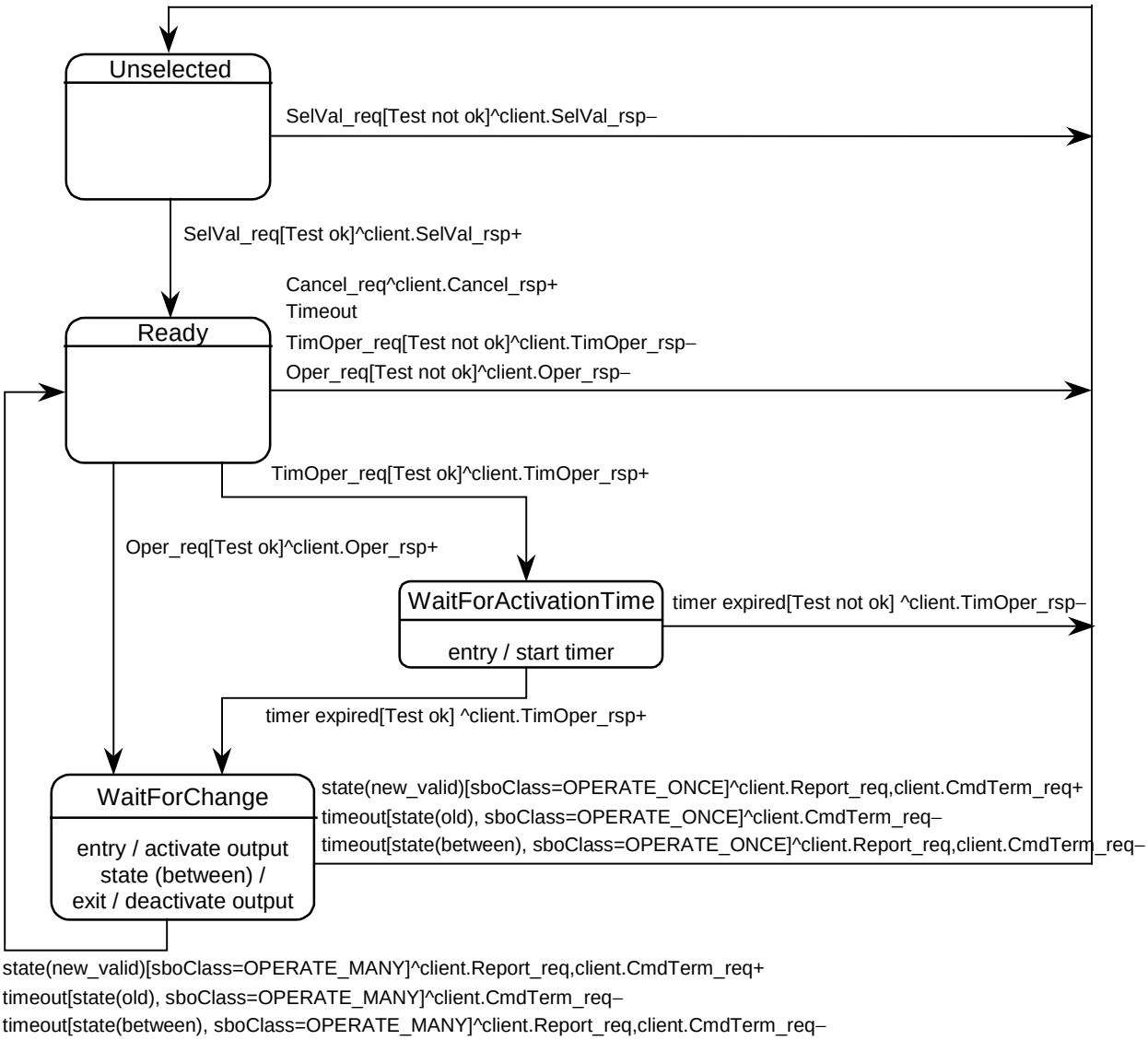


Figure 34 – State machine SBO control with enhanced security

Control with enhanced security should be used for control procedures that cause an important action outside the device containing the accessed control object.

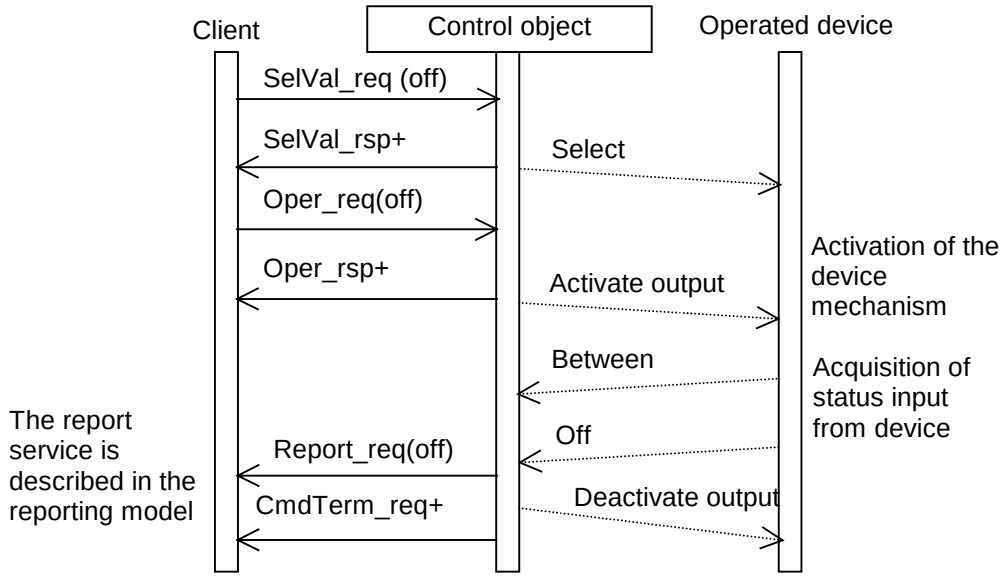


Figure 35 – Select before operate with enhanced security – positive case

NOTE The dashed lines in Figures 35 and 36 indicate that these "services" are local and not visible at the communication level.

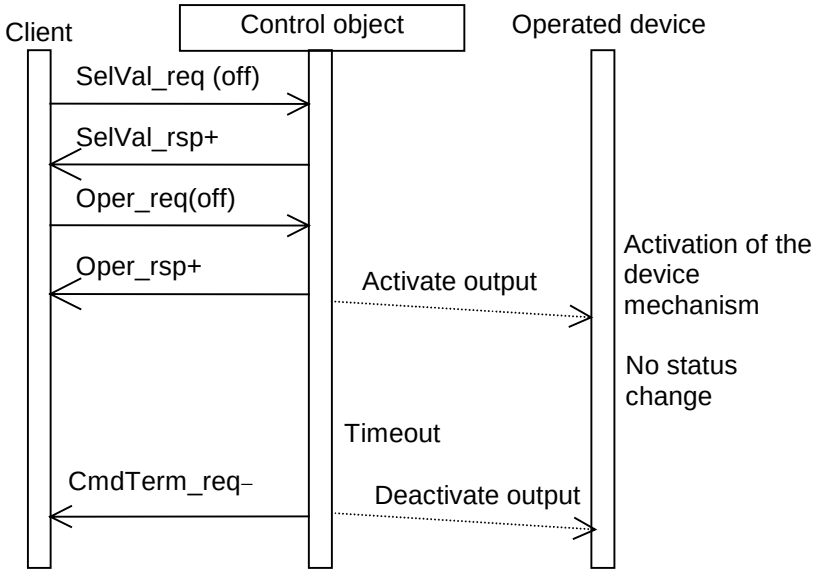


Figure 36 – Select before operate with enhanced security – negative case (no status change)

Procedure

a) On receipt of a **SelectWithValue** request, the control object shall determine if the client has appropriate access authority, that the control object is not currently selected by a different client, and that the device represented by the associated **LOGICAL-NODE** is operable and is not tagged so as to restrict operation.

   – If the **SelectWithValue** operation is not valid, the control object shall issue a negative response to the requesting client.

   – If the **SelectWithValue** operation is valid, the control object shall issue a positive response to the requesting client, shall change the state to ready and starts a deselect timer for either the interval defined by the SelTimOut attribute or, if unimplemented, some locally determined duration.

b) If the deselect timer expires before an **Operate** request on one or more of the other control components shall be requested by the selecting client, the control object shall change the state to unselected.

c) If an **Operate** request is received from the selecting client while the state is not Ready for that client, the operation shall be denied.

d) On receipt of an **Operate** request, the control object shall check validation of the control execution.

   – If not successful, the control object shall issue a negative response to the requesting client.

   – If successful, the control object shall issue a positive response to the requesting client and shall cause the requested action by activating a binary output (or sending an equivalent signal on a process bus). The control object shall turn to the state WaitForChange.

   – The control object supervises the change of the device status.

   – As soon as the status of the controlled device has changed, the control object shall report the new status using the report service of the reporting model.

   – If the status has not changed to the wanted value after a certain time, the control object shall issue a CommandTermination negative as soon as the output is deactivated.

   – When the object indicates the wanted position before expiration of a timer, the control object shall issues a CommandTermination positive as soon as the output is deactivated.

e) When leaving the WaitForChange state, one of the following procedures shall be performed based on the SBO-Select Class.

   – If the value of the sboClass attribute is **operate-once**, the new state shall be unselected.

   – If the value of the sboClass attribute is **operate-many**, the new state shall be Ready.

The last action shall be the command termination (**CmdTerm**) service.

17.4 Time-activated operate

Time-activated control shall consist of a **TimeActivatedOperate** request and response. The response shall inform the requesting client whether the command was successful, and had caused a time activation process, or unsuccessful.

This shall be an extension of the control model. To use the time-activated operate capability the service Operate in the control model shall be replaced by the service **TimeActivated-Operate**.

NOTE The example below is shown with the sboClass direct-operate. The use of select before operate mode is also possible. In that case, the control object must be in the state Ready before the service TimeActivatedOperate is supported.

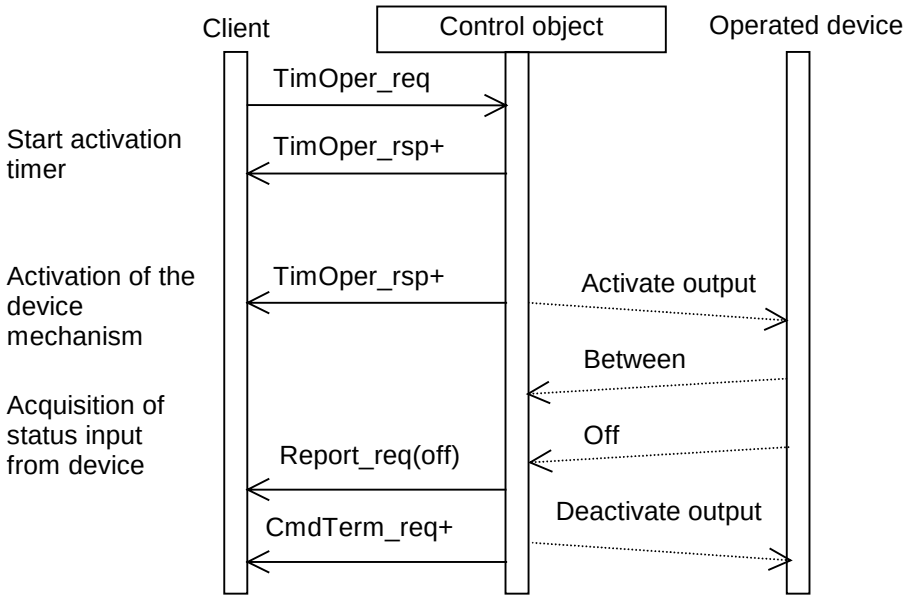


Figure 37 – Time-activated operate

Procedure

a) On receipt of a **TimeActivatedOperate** request the control object shall check the validity.

   – If not successful, the control object shall send a negative response to the requesting client.

   – If successful, the control object shall activate the timer and shall send a positive response with the information that the timer was started.

b) On expiration of the timer the wanted action shall be activated and a response shall be sent to the client.

c) All further information exchange shall be as described in the model for control with enhanced security.

17.5 CONTROL class service definitions

17.5.1 Overview

For **CONTROL** the following services listed in Table 35 are defined.

Table 35 – Control services

ACSI control service
Select (Sel)
SelectWithValue (SelVal)
Cancel (Cancel)
Operate (Oper)
CommandTermination (CmdTerm)
TimeActivatedOperate (TimOper)

17.5.2 Service parameter definition

The following service parameters shall be applied in the service definitions.

NOTE A pass-through parameter is a parameter that will not be interpreted by the service procedure. The value received in a pass-through parameter is available for any server application. These parameters are outside the scope of this standard; for example, the interpretation of Test/noTest is outside the scope of this standard.

17.5.2.1 ControlObjectReference

The parameter **ControlObjectReference** shall contain the ObjectReference of the controllable **DATA** (defined in IEC 61850-7-4) to be accessed, for example **Pos**, which represents the **DATA "Position"**.

17.5.2.2 Value

The parameter **Value** shall include values for all implemented **DataAttributes** of a controllable common **DATA** class that are accessed by various control services.

NOTE Common **DATA** classes and their **DataAttributes** are defined in IEC 61850-7-3.

EXAMPLE For the case of an **Operate** request, the value may include the following parameters:
– control value (on, off),
– originator category (remote, station, bay...),
– control sequence number.

17.5.2.3 T – control time-stamp

The parameter **T** shall be the time when the client sends the control request.

Table 36 – Control time-stamp definition

Control time-stamp type		
Attribute name	**Attribute type**	**Value/value range/explanation**
T	EntryTime	

17.5.2.4 Test – test status [pass-through parameter]

The parameter **Test** shall define whether the information is caused by normal operation or by test.

Table 37 – Test status definition

Test status type		
Attribute name	**Attribute type**	**Value/value range/explanation**
Test	BOOLEAN	no-test (FALSE) | test (TRUE)

17.5.2.5 Check – check condition

The parameter **Check** shall specify the kind of checks a control object shall perform before issuing the control operation if common **DATA** class is **DPC** (double-point control – see IEC 61850-7-3).

Table 38 – Check condition definition

Check condition type			
Attribute name		**Attribute type**	**Value/value range/explanation**
Check		PACKED LIST	
	synchrocheck	BOOLEAN	TRUE means run synchrocheck
	interlock-check	BOOLEAN	TRUE means run interlock-check

17.5.2.6 AddCause – additional cause diagnosis

The parameter **AddCause** shall identify the reason for failure in a negative control service specific response.

Table 39 – Additional cause diagnosis definition

Additional cause diagnosis type		
Attribute name	**Attribute type**	**Value/value range/explanation**
AddCause	ENUMERATION	ServiceError type Blocked-by-switching-hierarchy Select-failed Invalid-position Position-reached Parameter-change-in-execution Step-limit Blocked-by-Mode Blocked-by-process Blocked-by-interlocking Blocked-by-synchrocheck Command-already-in-execution Blocked-by-health 1-of-n-control Abortion-by-cancel Time-limit-over Abortion-by-trip

The description of the values shall be as defined in Table 40.

Table 40 – AddCause semantic

Value	Explanation
ServiceError type	All errors as defined in Table 5
Blocked-by-switching-hierarchy	Not successful since one of the downstream **Loc** switches like in **CSWI** has the value TRUE
Select-failed	Cancelled due to an unsuccessful selection (**select service**)
Invalid-position	Control action is aborted due to invalid switch position (**Pos** in **XCBR** or **XSWI**)
Position-reached	Switch is already in the intended position (**Pos** in **XCBR** or **XSWI**)
Parameter-change-in-execution	Control action is blocked due to running **parameter change**
Step-limit	Control action is blocked, because tap changer has reached the limit (**EndPosR** or **EndposL** in **YLTC**)
Blocked-by-Mode	Control action is blocked, because the LN (**CSWI** or **XCBR**/**XSWI**) is in a mode (**Mod**) which does not allow any switching
Blocked-by-process	Control action is blocked due to some external even at process level that prevents a successful operation, for example, blocking indication (**EEHealth** in **XCBR** or **XSWI**)
Blocked-by-interlocking	Control action is blocked due to interlocking of switching devices (in **CILO** attribute **EnaOpn.stVal="FALSE"** or **EnaCls.stVal="FALSE"**
Blocked-by-synchrocheck	Control action with synchrocheck is aborted due to the exceeding of the **time limit** and missing synchronism condition
Command-already-in-execution	Control service or cancel is rejected, because **control action** is already running
Blocked-by-health	Control action is blocked due to some internal event that prevents a successful operation (**Health**)
1-of-n-control	Control action is blocked, because another **control action** in a domain (for example, substation) is already **running** (in any **XCBR** or **XSWI**, the **DPC.stSeld**=**"TRUE"**).
Abortion-by-cancel	Control action is aborted due to **cancel service**
Time-limit-over	Control action is terminated due to exceed of some **time limit**
Abortion-by-trip	Control action is aborted due to a trip (**PTRC** mit **ACT.general="TRUE"**)

17.5.2.7 TimOperRsp – TimeActivatedOperate response

This parameter **TimOperRsp** shall specify the details of the positive response on the service **TimeActivatedOperate**

Table 41 – TimeActivatedOperate response definition

TimeActivatedOperate response type		
Attribute name	**Attribute type**	**Value/value range/explanation**
TimOperRsp	ENUMERATED	timer-activated | command-executed

17.5.3 Service specification

17.5.3.1 General

The services operate on several **DataAttributes** defined in common data classes of IEC 61850-7-3. Mainly the following **DataAttributes** defined in IEC 61850-7-3 are involved in control services:

– **ctlVal** (the value to be controlled);

– **operTm** (the time when to operate for the **TimeActivatedOperate** service);

– **origin** (indicating who issued the service);

– **ctlNum** (control sequence number).

These shall be set before the control services of this clause can be issued.

NOTE 1 The SCSM defines the subset of service parameters in the response service primitives. A communication stack that allows the client to assign a response to the relating request may not support all the service parameters that were also transmitted in the request.

NOTE 2 The additional cause diagnosis is a service parameter that is only transmitted in the response service primitives. The SCSM defines how this service parameter is included in the response PDU.

17.5.3.2 Select (Sel)

The **Select** service shall define the following service parameters.

Parameter name
Request
ControlObjectReference
Response+
ControlObjectReference
Response–
ControlObjectReference

NOTE The service parameters are defined in 17.5.2.

17.5.3.3 SelectWithValue (SelVal)

The **SelectWithValue** service shall define the following service parameters.

Parameter name
Request
ControlObjectReference
Value
T
Test
Check
Response+
ControlObjectReference
Value
T
Test
Response–
ControlObjectReference
Value
T
Test
AddCause

NOTE The service parameters are defined in 17.5.2.

17.5.3.4 Cancel

The **Cancel** service shall be used for the de-selection.

Parameter name
Request
ControlObjectReference
T
Test
Response+
ControlObjectReference
T
Test
Response–
ControlObjectReference
T
Test
AddCause

NOTE The service parameters are defined in 17.5.2.

17.5.3.5 Operate (Oper)

The **Operate** service shall define the following service parameters.

Parameter name
Request
ControlObjectReference
Value
T
Test
Check
Response+
ControlObjectReference
Value
T
Test
Response–
ControlObjectReference
Value
T
Test
AddCause

NOTE The service parameters are defined in 17.5.2.

17.5.3.6 CommandTermination (CmdTerm)

The **CommandTermination** service shall define the following service parameters.

Parameter name
Request+
ControlObjectReference
T
Test
Request–
ControlObjectReference
T
Test
AddCause

NOTE The service parameters are defined in 17.5.2.

17.5.3.7 TimeActivatedOperate (TimOper)

The **TimeActivatedOperate** service shall define the following service parameters.

Parameter name
Request
ControlObjectReference
Value
T
Test
Check
Response+
ControlObjectReference
Value
T
Test
TimOperRsp
Response–
ControlObjectReference
Value
T
Test
AddCause

NOTE The service parameters are defined in 17.5.2.

18 Time and time-synchronization model

18.1 General

The time and time-synchronization model shall provide the UTC synchronized time to applications located in server and client substation IEDs. The components of the time and time-synchronization model are depicted in Figure 38.

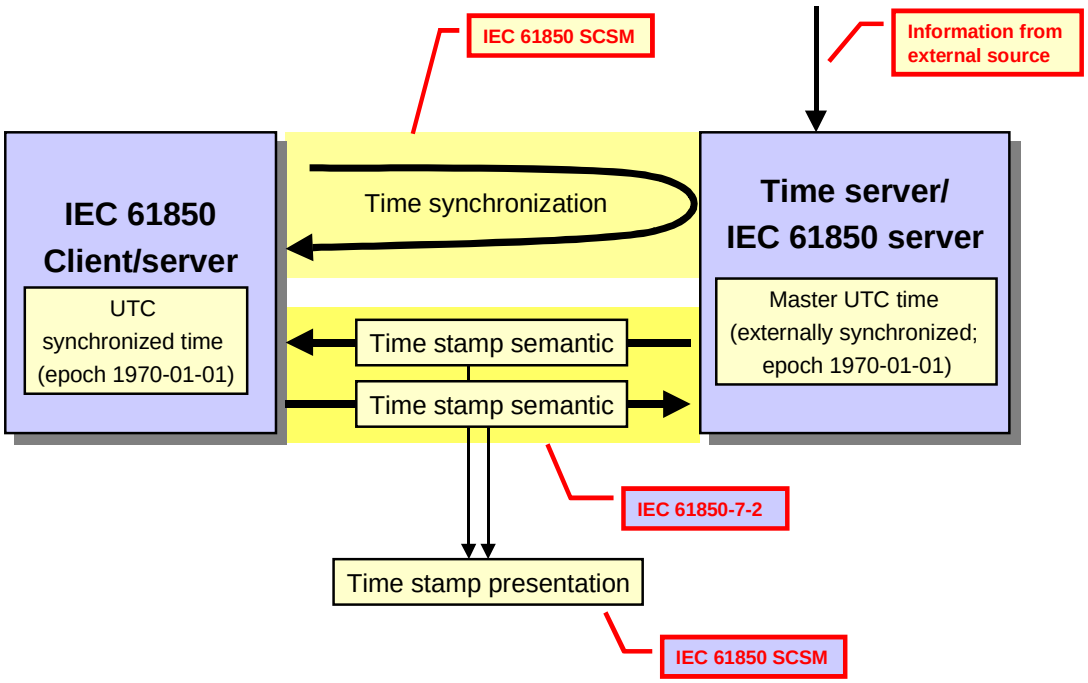


Figure 38 – Time model and time synchronization (principle)

The model shall comprise

– the **external information** required by the **time master** from an external source to synchronize other substation server or client IEDs (see 18.2);

– **time server** providing the source for the substation internal time synchronization and source for time stamping (in case the time server is implemented together with an IEC 61850 client/server in one physical device);

– **time synchronization** protocol providing time synchronization with other IEDs. Time synchronization shall meet the requirements of IEC 61850-5; the specification of time synchronization is defined in the SCSMs (for example, SNTP for IEC 61850-8-1);

– the **time stamp semantics** used for information exchange of the ACSI (see 5.5.3.6);

– the **presentation** of the time stamps according to the chosen SCSM;

– the **server** and **clients** that need substation-wide synchronized time.

18.2 External information

External information required for the time and time synchronization model shall provide the following.

a) Received external time

   – synchronized time to some known level of accuracy;

   – elapsed number of seconds since Epoch. If this count of seconds includes the leap seconds that have occurred since the epoch then the time produced by this time server shall have the LeapSecondsKnown quality attribute set to true, otherwise set to false.

b) **Epoch** (for example, GPS 6.1.1980).

19 Naming conventions

19.1 Class naming and class specializations

The classes for DATA, common DATA, compatible DATA, and compatible LOGICAL-NODE defined in IEC 61850-7-x make use of the following specializations:

IEC 61850-7-3 common **DATA** classes (for example, **DPC**) are specializations of the class **DATA** of IEC 61850-7-2

IEC 61850-7-4 compatible **DATA** classes (for example, Pos – position) are specializations of IEC 61850-7-3 common **DATA** classes (for example, DPC – controllable double point)

IEC 61850-7-4 compatible **LOGICAL-NODE** classes (for example, XCBR) are specializations of the **LOGICAL-NODE** class of IEC 61850-7-2

Figure 39 shows an overview of the specializations.

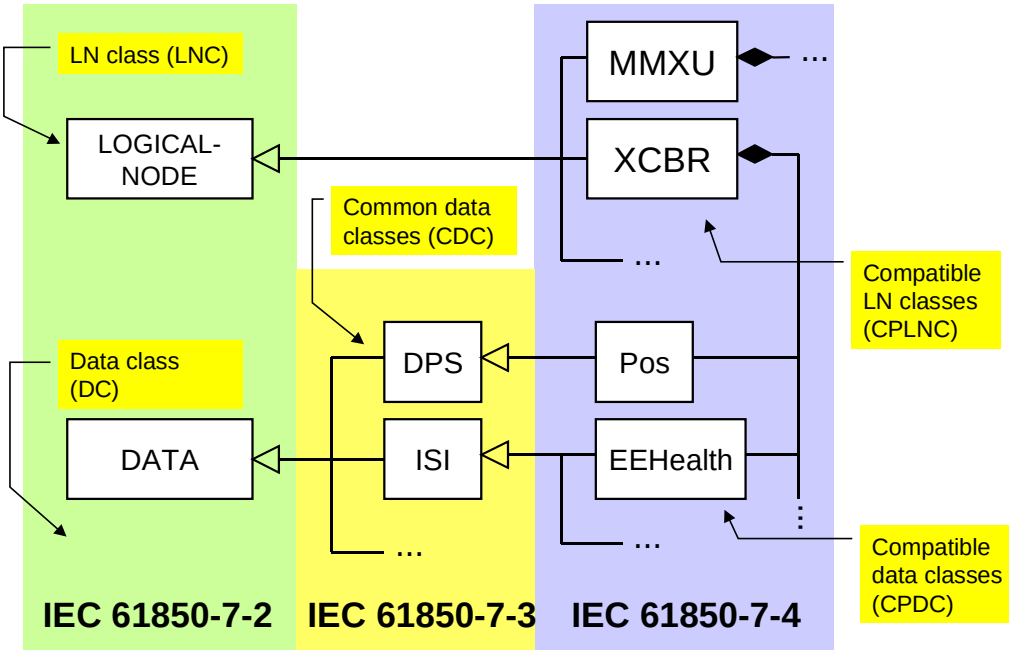


Figure 39 – Specializations

Each class in IEC 61850-7-x has its own class name. These class names shall be the basic building blocks when referencing class instances.

19.2 Referencing an instance of a class

The ObjectReferences and the abbreviations (used in class and service definitions) shall be as listed in Table 42.

Table 42 – List of ObjectReferences

ACSI class	ObjectReference of instance
LOGICAL-DEVICE	
LDRef (logical device reference)	LDName
LOGICAL-NODE	
LNRef (logical node reference)	LDName/LNName
DATA	
DataRef (data reference)	LDName/LNName.DataName[. DataName[. ...]]
DataAttribute	
DataAttributeReference (data attribute reference)	LDName/LNName. DataName[. DataName[. ...]]. DataAttributeName[.DAComponentName[. ...]]
DATA-SET	
DSRef (data set reference)	LDName/LNName.DataSetName (persistent), or @DataSetName (non-persistent)
SETTING-GROUP-CONTROL	
SGCB-Reference	LDName/LLN0.SGCB
BUFFERED-REPORT-CONTROL-BLOCK	
BRCBRef (buffered report control block reference)	LDName/LNName.BRCBName
UNBUFFERED-REPORT-CONTROL-BLOCK	
URCBRef (unbuffered report control block reference)	LDName/LNName.URCBName
LOG-CONTROL	
LCBRef (log control block reference)	LDName/LNName.LCBName
LOG	
LogRef (log reference)	LDName/LDName
GOOSE	
GoCBRef (GOOSE control block reference)	LDName/LLN0.GoCBName
GSSE	
GsCBRef (GOOSE control block reference)	LDName/LLN0.GsCBName
MSVCB	
MSVCBRef (multicast sampled value control block)	LDName/LLN0.MSVCBName
USVCB	
USVCBRef (multicast sampled value control block)	LDName/LLN0.USVCBName

Additionally, the following length definitions shall apply.

LDName/LNName.
DataName[.DataName[. ...]].DataAttributeName[.DAComponentName[. ...]]

The inner square bracket “[. ...]” shall indicate further recursive definitions of nested data attribute components.

LDName	=	up to 32 characters, application specific
LNName	=	[LN-Prefix] LN class name [LN-Instance-ID]
LN-Prefix	=	m characters (application specific)
LN class name	=	4 characters (for example, compatible logical node name as defined in IEC 61850-7-4)
LN-Instance-ID	=	n numeric characters (application specific)
m+n	≤	7 characters
DataName	=	up to 10 characters (as, for example, used in IEC 61850-7-4
DataName[.DataName[. ...]].DataAttributeName[.DAComponentName[. ...]]	≤	20 characters
FCD	≤	29 characters including all separators “.” (without the value of the **FC**)

The characters allowed shall be:

VisibleString (FROM

```
("A"|"a"|"B"|"b"|"C"|"c"|"D"|"d"|"E"|"e"|"F"|"f"|
"G"|"g"|"H"|"h"|"I"|"i"|"J"|"j"|"K"|"k"|"L"|"l"|
"M"|"m"|"N"|"n"|"O"|"o"|"P"|"p"|"Q"|"q"|"R"|"r"|
"S"|"s"|"T"|"t"|"U"|"u"|"V"|"v"|"W"|"w"|"X"|"x"|
"Y"|"y"|"Z"|"z"|"_"|"0"|"1"|"2"|"3"|"4"|"5"|"6"|
"7"|"8"|"9") )
```

EXAMPLE Figure 40 shows examples of object names and object references. The example at the top (first five lines) can be just five class definitions (not yet instantiated) or five instances of the classes E1.QA5/XCBR.Pos.ctlVal, ...stVal, ...q, ...t, ...ctlModel. The object references in this case do not indicate if object references refer to classes or instances. The context in which these references are used has to provide sufficient information to know what is meant (just class or instance).

The other examples refer to instances only.

NOTE The LD name E1.QA5 and its structure are outside the scope of IEC 61850. The functional constraint (FC) is not shown in the object reference. The FC information may be mapped into the ObjectReference in an SCSM; IEC 61850-8-1 maps the FC between LN and Data.

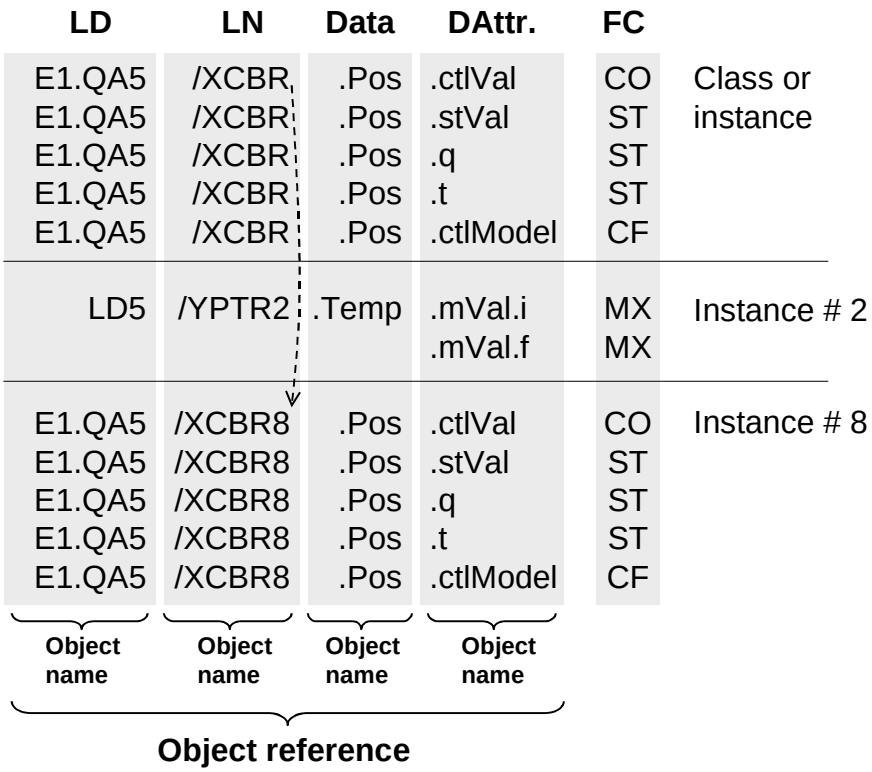


Figure 40 – Object names and object reference

19.3 Scope

Server specific scope (instances are defined outside of all LDs but in the server) shall be defined using the “/” and up to 32 characters to the right.

EXAMPLE /ABC.xyz

Logical device specific scope (instances are defined inside a specific LD) shall be defined as up to 32 characters, then “/” followed by up to 32 characters to the right.

EXAMPLE Atlanta_110/XCBR.Pos

TPAA specific scope (instances are defined inside a specific TPAAA) shall be defined using “@”, then “/” followed by up to 32 characters to the right.

EXAMPLE @/DataSet5 (for non-persistent **DATA-SETs).**

NOTE 1 The SCSMs may map the Reference to a flat numerical index or to a character string that is derived from the definition above. These character strings may comprise additional elements such as the functional constraint (FC).

NOTE 2 IEC 61850-6 gives additional definitions on how the application-specific character strings for logical devices can be built.

20 File transfer

20.1 File transfer model

The ACSI file transfer services shall provide the functionality for transferring files from and to file stores and for managing file stores.

NOTE The ACSI file services and the structure of the ACSI file store are intentionally limited in scope to simplify implementation in functionally restricted devices. The ACSI file store addresses a single file format – sequential unstructured binary – which may contain programs, data, or both. Any interpretation of the contents is by mutual agreement of the systems involved.

The **FILE** shall have the structure as defined in Table 43.

Table 43 – FILE class definition

FILE class		
Attribute name	**Attribute type**	**Value/value range/explanation**
FileName	VISIBLE STRING255	
FileSize [0...1]	INT32U	
LastModified	TimeStamp	
Services GetFile SetFile DeleteFile GetFileAttributeValues		

20.1.1 FileName

The attribute **FileName** shall be the name of the file in the ACSI file store.

NOTE File names may be structured to differentiate file types, for example, disturbance records, programs, and parameter and configuration data.

20.1.2 FileSize [0..1]

The attribute **FileSize** (in octets) shall be the length of a file in the file store.

NOTE In case the FileSize cannot be determined (for example, in the case of an on-the-fly created COMTRADE file) the meaning and interpretation of the FileSize is outside the scope of this standard.

20.1.3 LastModified

The attribute **LastModified** shall be the time when the file was last modified.

20.2 File services

20.2.1 GetFile

20.2.1.1 GetFile parameter

The **GetFile** service shall be used by a client to transfer the contents of a file from the server to the client.

Parameter name
Request
FileName
Response+
File-Data
Response–
ServiceError

20.2.1.2 Request

FileName

The parameter **FileName** shall specify the name of the file being transferred.

20.2.1.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. A successful result shall return the following parameter.

File-Data

The parameter **File-Data** shall contain the data transferred; the type of file-data is octet string.

20.2.1.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

20.2.2 SetFile

20.2.2.1 SetFile parameter

The SetFile service shall be used by a client to transfer the contents of a file from the client to the server.

Parameter name
Request
FileName
File-Data
Response+
Response–
ServiceError

20.2.2.2 Request

20.2.2.2.1 FileName

The parameter **FileName** shall specify the name of the file being transferred.

20.2.2.2.2 File-Data

The parameter **File-Data** shall contain the data transferred; the type of file-data is octet string.

20.2.2.3 Response+

The parameter **Response+** shall indicate that the service request succeeded.

20.2.2.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

20.2.3 DeleteFile

20.2.3.1 DeleteFile parameter

The **FileDelete** service shall be used by a client to delete a file in the file store of a server.

Parameter name
Request
FileName
Response+
Response–
ServiceError

20.2.3.2 Request

FileName

The parameter **FileName** shall specify the name of the file being deleted.

20.2.3.3 Response+

The parameter **Response+** shall indicate that the service request succeeded.

20.2.3.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

20.2.4 GetFileAttributeValues

20.2.4.1 GetFileAttributeValues parameter

The **GetFileAttributeValues** service shall be used by a client to obtain the name and attributes of a specific file in the server's file store.

Parameter name
Request
FileName
Response+
FileName
FileAttribute [1..n]
Response–
ServiceError

20.2.4.2 Request

FileName

The parameter **FileName** shall, when present, specify the name of the file whose attributes are requested to be returned to the client.

20.2.4.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. A successful result shall return the following parameters.

20.2.4.3.1 FileName

The parameter **FileName** shall provide the name of the file whose attributes are returned.

20.2.4.3.2 FileAttribute [1..n]

The parameter **FileAttribute** shall contain attribute information describing the selected file. This information consists of the size of the file and time of last modification.

20.2.4.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned.

Annex A
(normative)

ACSI conformance statement

A.1 General

The following ACSI conformance statements shall be used to provide an overview and details about a device claiming conformance with ACSI:

– ACSI basic conformance statement

– ACSI models conformance statement

– ACSI service conformance statement

to specify the communication features mapped to an SCSM.

NOTE 1 The conformance statements of this annex are abstract in the sense that the ACSI models and their services are mapped to application layer models, services, and protocols. Additional details on the conformance are defined in the SCSM.

NOTE 2 For several features the conformance requirement is implicitly defined with the common data class contained in IEC 61850-7-3 and the compatible **LOGICAL-NODE** classes and **DATA** classes contained in IEC 61850-7-4, for example, a TrgOp (trigger option) of the value qchg (quality change) of **DataAttribute** requires the support of the TrgOpEna (trigger option enabled) qchg of the **BRCB** or **URCB**.

A.1 ACSI basic conformance statement

The basic conformance statement shall be as defined in Table A.1.

Table A.1 – Basic conformance statement

		Client/ subscriber	Server/ publisher	Value/ comments
Client-server roles				
B11	**Server** side (of TWO-PARTY-APPLICATION-ASSOCIATION)	–	c1	
B12	**Client** side of (TWO-PARTY-APPLICATION-ASSOCIATION)	c1	–	
SCSMs supported				
B21	**SCSM**: IEC 6185-8-1 used			
B22	**SCSM**: IEC 6185-9-1 used			
B23	**SCSM**: IEC 6185-9-2 used			
B24	**SCSM**: other			
Generic substation event model (GSE)				
B31	**Publisher** side	–	O	
B32	**Subscriber** side	O	–	
Transmission of sampled value model (SVC)				
B41	**Publisher** side	–	O	
B42	**Subcriber** side	O	–	

c1 – shall be ‘M’ if support for **LOGICAL-DEVICE** model has been declared.

O – Optional

M – Mandatory

A.2 ACSI models conformance statement

The ACSI models conformance statement shall be as defined in Table A.2.

Table A.2 – ACSI models conformance statement

		Client/ subscriber	Server/ publisher	Value/ comments
If **Server** side (B1) supported				
M1	**Logical device**	c2	c2	
M2	**Logical node**	c3	c3	
M3	**Data**	c4	c4	
M4	**Data set**	c5	c5	
M5	**Substitution**	O	O	
M6	**Setting group control**	O	O	
	Reporting			
M7	**Buffered report control**	O	O	
M7-1	sequence-number			
M7-2	report-time-stamp			
M7-3	reason-for-inclusion			
M7-4	data-set-name			
M7-5	data-reference			
M7-6	buffer-overflow			
M7-7	entryID			
M7-8	BufTm			
M7-9	IntgPd			
M7-10	GI			
M8	**Unbuffered report control**	M	M	
M8-1	sequence-number			
M8-2	report-time-stamp			
M8-3	reason-for-inclusion			
M8-4	data-set-name			
M8-5	data-reference			
M8-6	BufTm			
M8-7	IntgPd			
	Logging	O	O	
M9	**Log control**	O	O	
M9-1	IntgPd			
M10	**Log**	O	O	
M11	**Control**	M	M	
If **GSE** (B31/32) is supported				
	GOOSE	O	O	
M12-1	entryID			
M12-2	DataRefInc			
M13	**GSSE**	O	O	

Table A.2 *(continued)*

		Client/ subscriber	Server/ publisher	Value/ comments
If **SVC** (41/42) is supported				
M14	Multicast SVC	O	O	
M15	Unicast SVC	O	O	
M16	**Time**	M	M	Time source with required accuracy shall be available
M17	**File Transfer**	O	O	

c2 – shall be 'M' if support for **LOGICAL-NODE** model has been declared.
c3 – shall be 'M' if support for **DATA** model has been declared.
c4 – shall be 'M' if support for **DATA-SET**, Substitution, Report, Log Control, or Time model has been declared.
c5 – shall be 'M' if support for Report, GSE, or SMV models has been declared.

M – Mandatory

A.3 ACSI service conformance statement

The ACSI service conformance statement shall be as defined in Table A.3 (depending on the statements in Table A.1).

Table A.3 – ACSI service conformance statement

	Services	AA: TP/MC	Client (C)	Server (S)	Comments
Server (Clause 6)					
S1	ServerDirectory	TP		M	
Application association (Clause 7)					
S2	Associate		M	M	
S3	Abort		M	M	
S4	Release		M	M	
Logical device (Clause 8)					
S5	LogicalDeviceDirectory	TP	M	M	
Logical node (Clause 9)					
S6	LogicalNodeDirectory	TP	M	M	
S7	GetAllDataValues	TP	O	M	
Data (Clause 10)					
S8	GetDataValues	TP	M	M	
S9	SetDataValues	TP	O	O	
S10	GetDataDirectory	TP	O	M	
S11	GetDataDefinition	TP	O	M	

Table A.3 *(continued)*

	Services	AA: TP/MC	Client (C)	Server (S)	Comments
Data set (Clause 11)					
S12	GetDataSetValues	TP	O	M	
S13	SetDataSetValues	TP	O	O	
S14	CreateDataSet	TP	O	O	
S15	DeleteDataSet	TP	O	O	
S16	GetDataSetDirectory	TP	O	O	
Substitution (Clause 12)					
S17	SetDataValues	TP	M	M	
Setting group control (Clause 13)					
S18	SelectActiveSG	TP	O	O	
S19	SelectEditSG	TP	O	O	
S20	SetSGValues	TP	O	O	
S21	ConfirmEditSGValues	TP	O	O	
S22	GetSGValues	TP	O	O	
S23	GetSGCBValues	TP	O	O	
Reporting (Clause 14)					
Buffered report control block (BRCB)					
S24	Report	TP	c6	c6	
S24-1	data-change (dchg)				
S24-2	qchg-change (qchg)				
S24-3	data-update (dupd)				
S25	GetBRCBValues	TP	c6	c6	
S26	SetBRCBValues	TP	c6	c6	
Unbuffered report control block (URCB)					
S27	Report	TP	c6	c6	
S27-1	data-change (dchg)				
S27-2	qchg-change (qchg)				
S27-3	data-update (dupd)				
S28	GetURCBValues	TP	c6	c6	
S29	SetURCBValues	TP	c6	c6	
c6 – shall declare support for at least one (BRCB or URCB).					
Logging (Clause 14)					
Log control block					
S30	GetLCBValues	TP	M	M	
S31	SetLCBValues	TP	O	M	
Log					
S32	QueryLogByTime	TP	c7	M	
S33	QueryLogAfter	TP	c7	M	
S34	GetLogStatusValues	TP	M	M	
c7 – shall declare support for at least one (QueryLogByTime or QueryLogAfter).					

Table A.3 *(continued)*

	Services	AA: TP/MC	Client (C)	Server (S)	Comments
Generic substation event model (GSE) (14.3.5.3.4)					
GOOSE-CONTROL-BLOCK					
S35	SendGOOSEMessage	MC	c8	c8	
S36	GetReference	TP	O	c9	
S37	GetGOOSEElementNumber	TP	O	c9	
S38	GetGoCBValues	TP	O	O	
S39	SetGoCBValues	TP	O	O	
GSSE-CONTROL-BLOCK					
S40	SendGSSEMessage	MC	c8	c8	
S41	GetReference	TP	O	c9	
S42	GetGSSEElementNumber	TP	O	c9	
S43	GetGsCBValues	TP	O	O	
S44	SetGsCBValues	TP	O	O	
c8 – shall declare support for at least one (SendGOOSEMessage or SendGSSEMessage). c9 – shall declare support if TP association is available.					

Transmission of sampled value model (SVC) (Clause 16)					
Multicast SVC					
S45	SendMSVMessage	MC	c10	c10	
S46	GetMSVCBValues	TP	O	O	
S47	SetMSVCBValues	TP	O	O	
Unicast SVC					
S48	SendUSVMessage	TP	c10	c10	
S49	GetUSVCBValues	TP	O	O	
S50	SetUSVCBValues	TP	O	O	
c10 – shall declare support for at least one (SendMSVMessage or SendUSVMessage).					

Control (16.4.8)					
S51	Select		M	M	
S52	SelectWithValue	TP	M	M	
S53	Cancel	TP	O	M	
S54	Operate	TP	M	M	
S55	Command-Termination	TP	M	M	
S56	TimeActivated-Operate	TP	O	O	

File transfer (Clause 20)					
S57	GetFile	TP	O	M	
S58	SetFile	TP	O	O	
S59	DeleteFile	TP	O	O	
S60	GetFileAttributeValues	TP	O	M	

Table A.3 *(continued)*

	Services	AA: TP/MC	Client (C)	Server (S)	Comments
Time (5.5)					
T1	Time resolution of internal clock				Nearest negative power of 2 in seconds
T2	Time accuracy of internal clock				T0
					T1
					T2
					T3
					T4
					T5
T3	Supported TimeStamp resolution				Nearest negative power of 2 in seconds

Bibliography

IEEE-SA TR 1550-1999 – Utility Communications Architecture (UCA™) Version 2 [4].

[4] UCA™ is a registered trade mark of EPRI, Palo Alto (USA).

Index

_____

ISBN 2-8318-XXXX-X

ICS 33.220

Typeset and printed by the IEC Central Office
GENEVA, SWITZERLAND