IEC 61850-7-2

Edition 2.0 2010-08

INTERNATIONAL STANDARD

colour inside

Communication networks and systems for power utility automation – Part 7-2: Basic information and communication structure – Abstract communication service interface (ACSI)

IEC 61850-7-2:2010(E)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub

The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

IEC 61850-7-2

Edition 2.0 2010-08

INTERNATIONAL STANDARD

colour inside

Communication networks and systems for power utility automation – Part 7-2: Basic information and communication structure – Abstract communication service interface (ACSI)

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PRICE CODE XK

ICS 33.200

ISBN 978-2-88912-065-9

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMUNICATION NETWORKS AND SYSTEMS FOR POWER UTILITY AUTOMATION –

Part 7-2: Basic information and communication structure – Abstract communication service interface (ACSI)

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). 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 organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of 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 IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. 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 systems management and associated information exchange.

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

FDIS	Report on voting
57/1065/FDIS	57/1083/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 second edition cancels and replaces the first edition published in 2003. It constitutes a technical revision.

Future standards in this series will carry the new general title as cited above. Titles of existing standards in this series will be updated at the time of the next edition.

The major technical changes with regard to the previous edition are as follows:

- class diagrams have been updated,
- data types not required have been removed,
- errors and typos haven been corrected,
- substitution model has been moved to IEC 61850-7-3,
- service tracking for control blocks have been added,
- the view concept will be according to the new work on role bases access (RBA),
- security issues are solved by the IEC 62351 series, and
- several terms have been harmonized with those in the other parts.

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

In this document, the following print types are used:

– **bold** is used to highlight defined terms,

– Tahoma is used where the difference between a capital i (I) and a small L (l) is important to see.

A list of all parts of the IEC 61850 series, under the general title: *Communication networks and systems for power utility automation*, can be found on the IEC website.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. 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.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

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

The IEC 61850 series is intended to provide interoperability between a variety of 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 IEC 61850-7-x.

This part of IEC 61850 defines the abstract communication service interface (ACSI) for use in the utility application domain that requires 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[1)] (SCSM) are specified in IEC 61850-8-x and IEC 61850-9-x.

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.

NOTE 3 Examples use names of classes (for example XCBR for a class of a logical node) defined in IEC 61850-7-4 and IEC 61850-7-3. The normative names are defined in IEC 61850-7-4 and IEC 61850-7-3 only.

---

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

COMMUNICATION NETWORKS AND SYSTEMS FOR POWER UTILITY AUTOMATION –

Part 7-2: Basic information and communication structure – Abstract communication service interface (ACSI)

1 Scope

This part of IEC 61850 applies to the ACSI communication for utility automation. The ACSI provides the following abstract communication service 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,
   - setting group control,
   - 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/sub-scriber) and for transmission of sampled measured values (publisher/subscriber).

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 61850-6, *Communication networks and systems for power utility automation – Part 5: Configuration description language for communication in electrical substations related to IEDs*

IEC 61850-7-1, *Communication networks and systems for power utility automation – Part 7-1: Basic communication structure – Principles and models*[2)]

IEC 61850-7-3, *Communication networks and systems for power utility automation – Part 7-3: Basic communication structure – Common data classes*[2)]

IEC 61850-7-4, *Communication networks and systems for power utility automation – Part 7-4: Basic communication structure – Compatible logical node classes and data object classes*

---

2) To be published.

IEC 61850-8-1, *Communication networks and systems for power utility automation – Part 8-1: Specific communication service mapping (SCSM) – Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3*[3)]

IEC 61850-9-2, *Communication networks and systems for power utility automation – Part 9-2: Specific communication service mapping (SCSM) – Sampled values over ISO/IEC 8802-3*[3)]

ISO 4217, *Codes for the representation of currencies and funds*

ISO 9506 (all parts), *Industrial automation systems – Manufacturing Message Specification*

IEEE 754, *Standard for Floating-Point Arithmetic*

3 Terms and definitions

For the purposes of this document, the terms and definitions provided in IEC 61850-2 and the following 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 generation, transport, or distribution functions.

3.4
external equipment
entity that is stand-alone, or interfaces to an automation system, and that performs energy generation, transport, or distribution 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) To be published.

3.6
logical device
entity that represents a set of typical automation, protection or other functions

3.7
logical node
entity that represents a typical automation, protection or other 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.

4 Abbreviated terms

AA	application association
ACSI	abstract communication service interface
BRCB	buffered report control block
CB	control block
CDC	common data class (IEC 61850-7-3)
CT	current transformer
DA	data attribute
DataRef	data reference
dchg	data change trigger option
DS	data set
dupd	data-update trigger option
FC	functional constraint
FCD	functional constrained data
FCDA	functional constrained data attribute
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 (in this part of IEC 61850, the generic logical device class is defined (genLogicalDevice)
LN	logical node (in this part of IEC 61850, the generic logical node class is defined (genLogicalNode)
MC	multicast
MCAA	multicast application association
MMS	manufacturing message specification (ISO 9506)

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)
SGCB	setting group control block
SoE	sequence-of-events
SVC	sampled value control
TP	two party
TPAA	two party application association
TrgOp	trigger option
UCA™ 4)	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 Conceptual model of IEC 61850

The models of the ACSI provide

– the definition of the basic model of the utility information models contained in IEC 61850-7-3 (common data classes for utility automation applications), IEC 61850-7-4 (compatible logical node classes and compatible data classes for utility automation applications) and IEC 61850-6 (substation configuration language), and

– the definition 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 first level of the definitions is a list of base types and rules how to build hierarchical structures (meta-meta model) defined in 5.2.

4) UCA™ is a registered trade mark of EPRI, Palo Alto (USA). This information is given for the convenience of users and does not constitute an endorsement by the IEC of the product named.

The generic models (meta model) for example, for logical nodes and data classes including their services, are defined 5.3 and applied in IEC 61850-7-3 and IEC 61850-7-4 to define many specialized information models for utility automation models or in IEC 61400-25-2 to define specialized models for wind power plant applications (domain type models; see 5.4).

The part IEC 61850-6 (SCL) defines the instances to be implemented (configured) in real devices (data instance model; see 5.5).

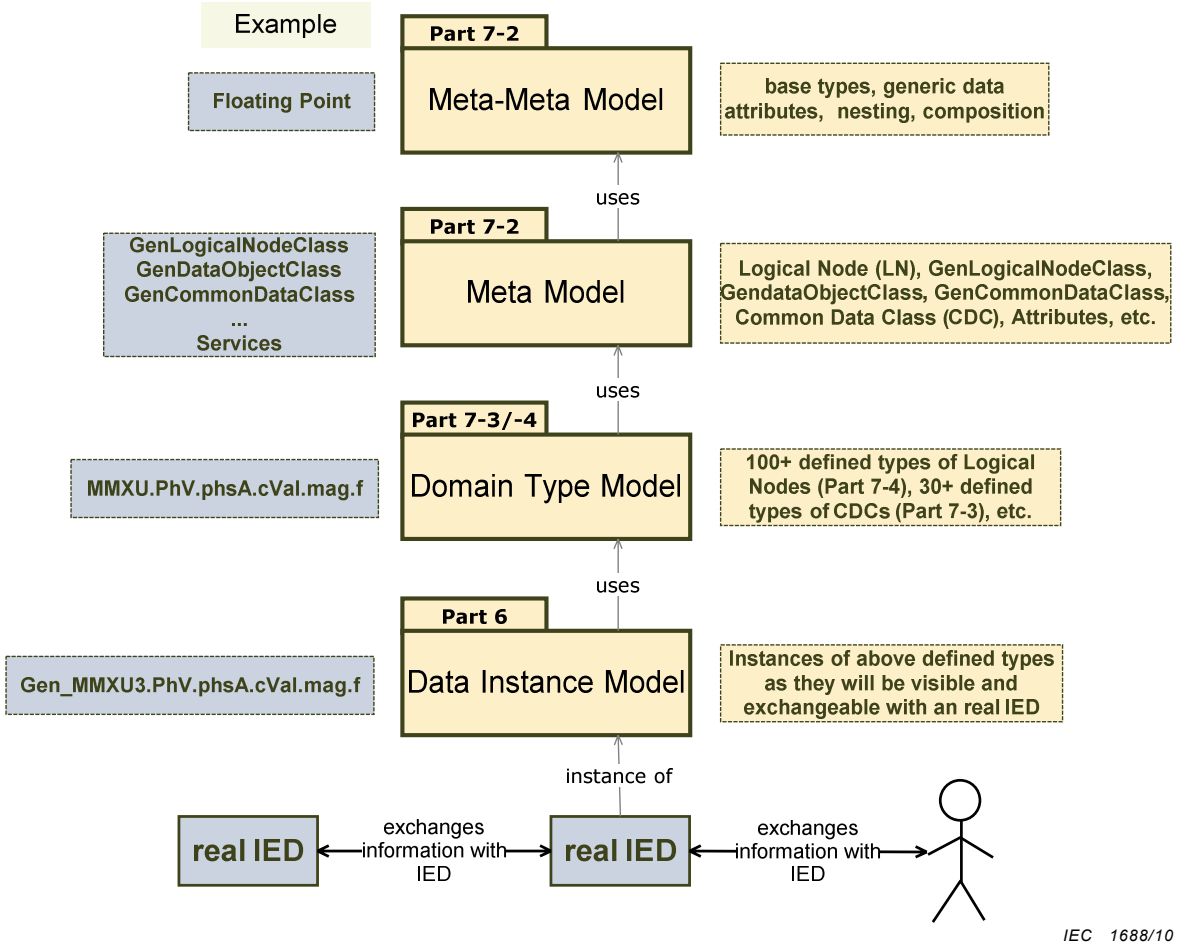


Figure 1 – Excerpt of conceptual model of IEC 61850

5.2 The meta-meta model

The meta-meta model is defined in Clause 6. It defines the basic data type classes to be used in the meta model with the exception of the recursion (nesting) of components to define hierarchical data models. The recursion is defined in the meta model.

5.3 The meta model

5.3.1 General

This part of IEC 61850 defines mainly the meta model for the whole IEC 61850 standard series. The meta model comprises classes for the description of a device with regard to data models and information exchange.

5.3.2 Information modelling classes

The following overall classes are defined:

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

NOTE 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)** – represents the information produced and consumed by a group of domain-specific application functions.

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

d) **Data objects** – provide means to define typed information, for example, position of a switch with quality information and timestamp, contained in logical nodes.

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

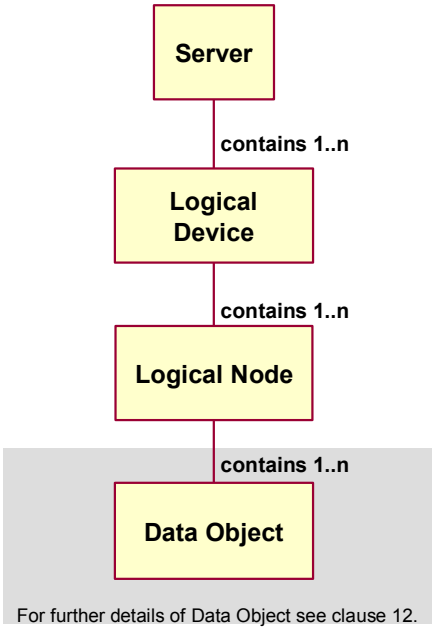


Figure 2 – Basic conceptual class model of the ACSI

Each of the following classes has a name and a reference: logical device, logical node, and data object.

EXAMPLE In an implementation, the logical device, logical node, data objects, 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 object	Data attribute
Object name	"Atlanta_HV5"	"**XCBR1**"	"**Pos**"	"**stVal**"
Description	High-voltage station 5	Circuit-breaker 1	Position	Status value

5.3.3 Information exchange modelling classes

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

a) **Data Set** – permits the grouping of data objects and data attributes. Used for direct access, for reporting, logging, GOOSE messaging and sampled value exchange.

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

c) **Setting group control** – defines how to switch from one set of setting values to another one and how to edit setting groups.

d) **Report control** and **logging** – describe the conditions for generating reports and logs based on parameters set by configuration or by a 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 system** – defines the exchange of large data blocks such as programs.

j) **Tracking** – provides a diagnosis interface to track services (control, configuration, exchange).

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

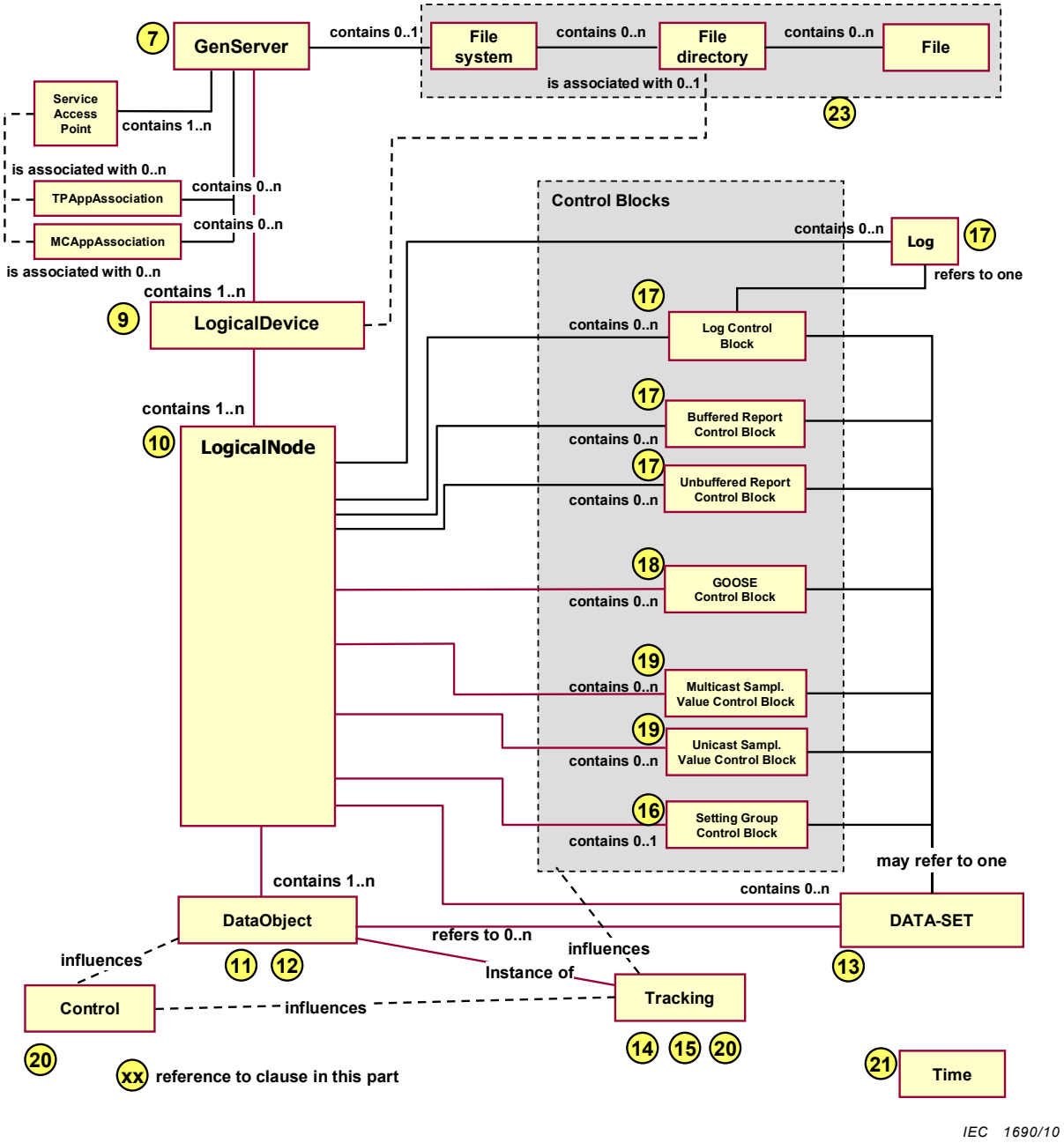


Figure 3 – Conceptual service model of the ACSI

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 logical node is one of the major building blocks that have associations to most of the other information exchange models, for example, report control, log control, and setting control. In this part of IEC 61850, the generic logical node class is defined (**GenLogicalNode**).

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.

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

Table 1 – ACSI model classes with related services

GenServer model GetServerDirectory **Association model** Associate Abort Release **GenLogicalDeviceClass model** GetLogicalDeviceDirectory **GenLogicalNodeClass model** GetLogicalNodeDirectory GetAllDataValues **GenDataObjectClass model** GetDataValues SetDataValues GetDataDirectory GetDataDefinition **DATA-SET model** GetDataSetValues SetDataSetValues CreateDataSet DeleteDataSet GetDataSetDirectory **SETTING-GROUP-CONTROL-BLOCK model** SelectActiveSG SelectEditSG SetEditSGValue ConfirmEditSGValues GetEditSGValue GetSGCBValues **REPORT-CONTROL-BLOCK and LOG-CONTROL-BLOCK model** **BUFFERED-REPORT-CONTROL-BLOCK**: Report[a] GetBRCBValues SetBRCBValues **UNBUFFERED-REPORT-CONTROL-BLOCK:** Report[a] GetURCBValues SetURCBValues	**LOG-CONTROL-BLOCK model:** GetLCBValues SetLCBValues QueryLogByTime QueryLogAfter GetLogStatusValues **Generic substation event model – GSE** **GOOSE** SendGOOSEMessage[a] GetGoReference GetGOOSEElementNumber GetGoCBValues SetGoCBValues **Transmission of sampled values model** **MULTICAST-SAMPLE-VALUE-CONTROL-BLOCK:** SendMSVMessage[a] GetMSVCBValues SetMSVCBValues **UNICAST-SAMPLE-VALUE-CONTROL-BLOCK:** SendUSVMessage[a] GetUSVCBValues SetUSVCBValues **Control model** Select SelectWithValue Cancel Operate CommandTermination TimeActivatedOperate **Time and time synchronization** TimeSynchronization **FILE transfer model** GetFile SetFile DeleteFile GetFileAttributeValues
a) All the services for spontaneous sending are limited to one access point per control block instance.	

NOTE All services defined in this part of IEC 61850 operate on instances of classes only. The service GetDataValues for example operates on an instantiated data object class implemented in a real device. The service parameters in the abstract service tables and the definition of the services refer to those instances and not to the generic classes defined in this part of IEC 61850.

5.3.4 Relations between classes

The crucial relations of the meta model classes are shown in Figure 4. The figure shows the crucial building rule for data objects (the recursions). The abstract model in the ACSI uses the generic common data class model to define any hierarchical model of domain specific information. The class diagram uses two recursions: the GenCommonDataClass and the GenConstructedAttributeClass. These two recursions allow the definition of any common data class defined in IEC 61850-7-3.

Figure 4 shows some examples of the definitions of IEC 61850-7-3 (common data classes WYE, CMV; attributes like cVal, etc.) and IEC 61850-7-4 (logical node MMXU and data object PhV). The various types, names and identifiers (IDs) are shown in coloured boxes with the

part number indicated as “6” for IEC 61850-6, “7-3” for names and identifiers defined in IEC 61850-7-3, and “7-4” for logical node and data object names defined in IEC 61850-7-4.

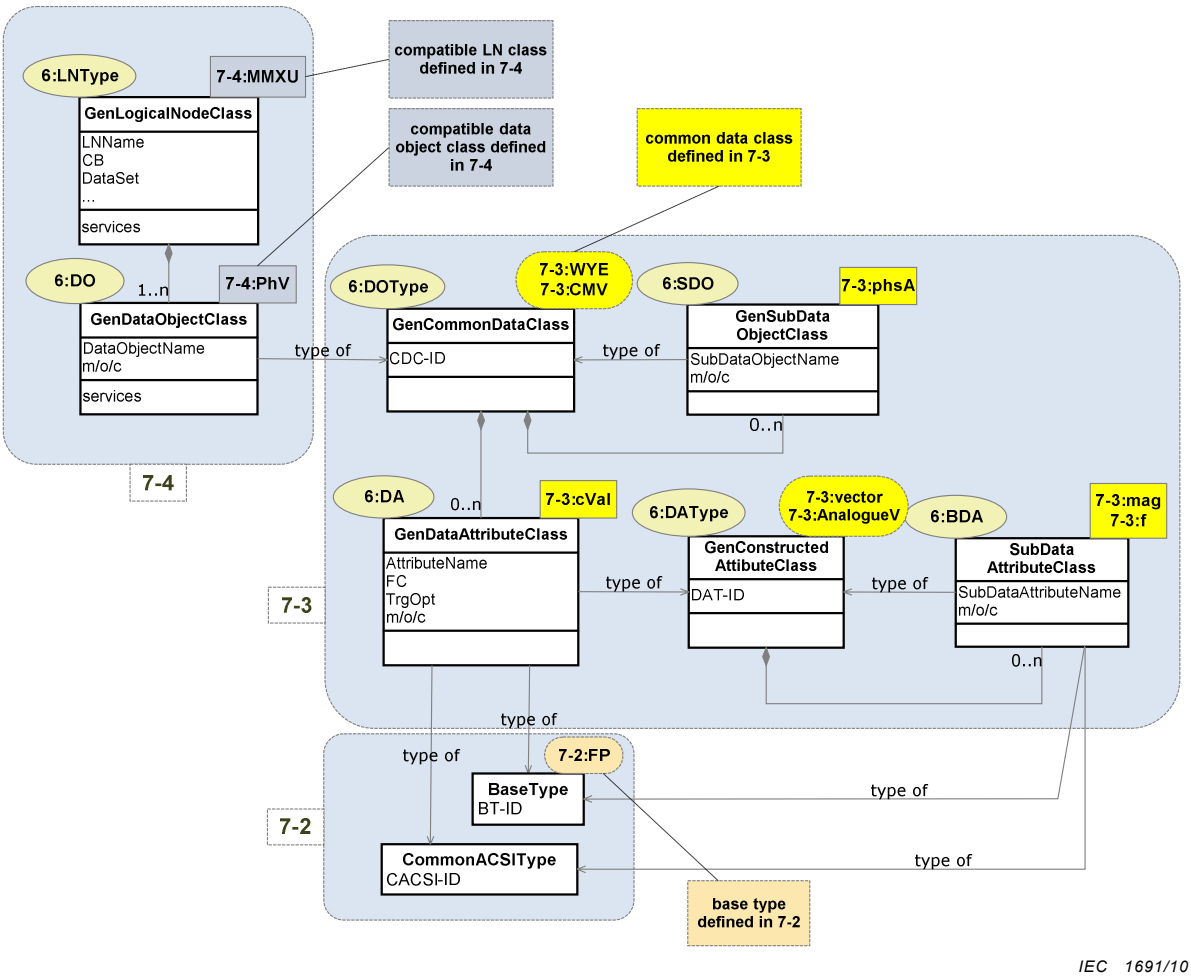


Figure 4 – Core of the conceptual meta model and relationship

The **GenCommonDataClass** is one of the crucial models to build the information models. The **GenCommonDataClass** model is used as a rule to define (build) common data classes (common for many domains like SPC for single point status or specific for a domain like WYE for electrical applications).

5.4 The domain type model

The domain type model of IEC 61850 defines lists of common data classes (CDC, IEC 61850-7-3), data objects (typed by common data classes) and logical node classes (IEC 61850-7-4) aggregating data objects. These classes are used to build data models for real IEDs.

IEC 61850-6 defines a method to describe the data instance model based on these classes, that can be used to describe the complete model implemented (programmed or configured) in a real IED. The data instance model is introduced in 5.5.

5.5 The data instance model

The data instance model describes instances of the classes defined in IEC 61850-7-x (see Figure 5). IEC 61850-6 defines by means of an XML schema (the SCL schema) a language to describe the configuration of IEDs. The SCL schema uses the element DOType to describe the common data class instantiation of a specific data object (DO element) in the logical node type (LNType). IEC 61850-6 defines an IED element that has logical devices (LD) composed of

logical nodes (LN). The logical nodes are typed by instantiable LNTypes listed in the DataTypeTemplate section of an SCL document. Data objects in a LNType become a data object instance (DOI) in the corresponding logical node.

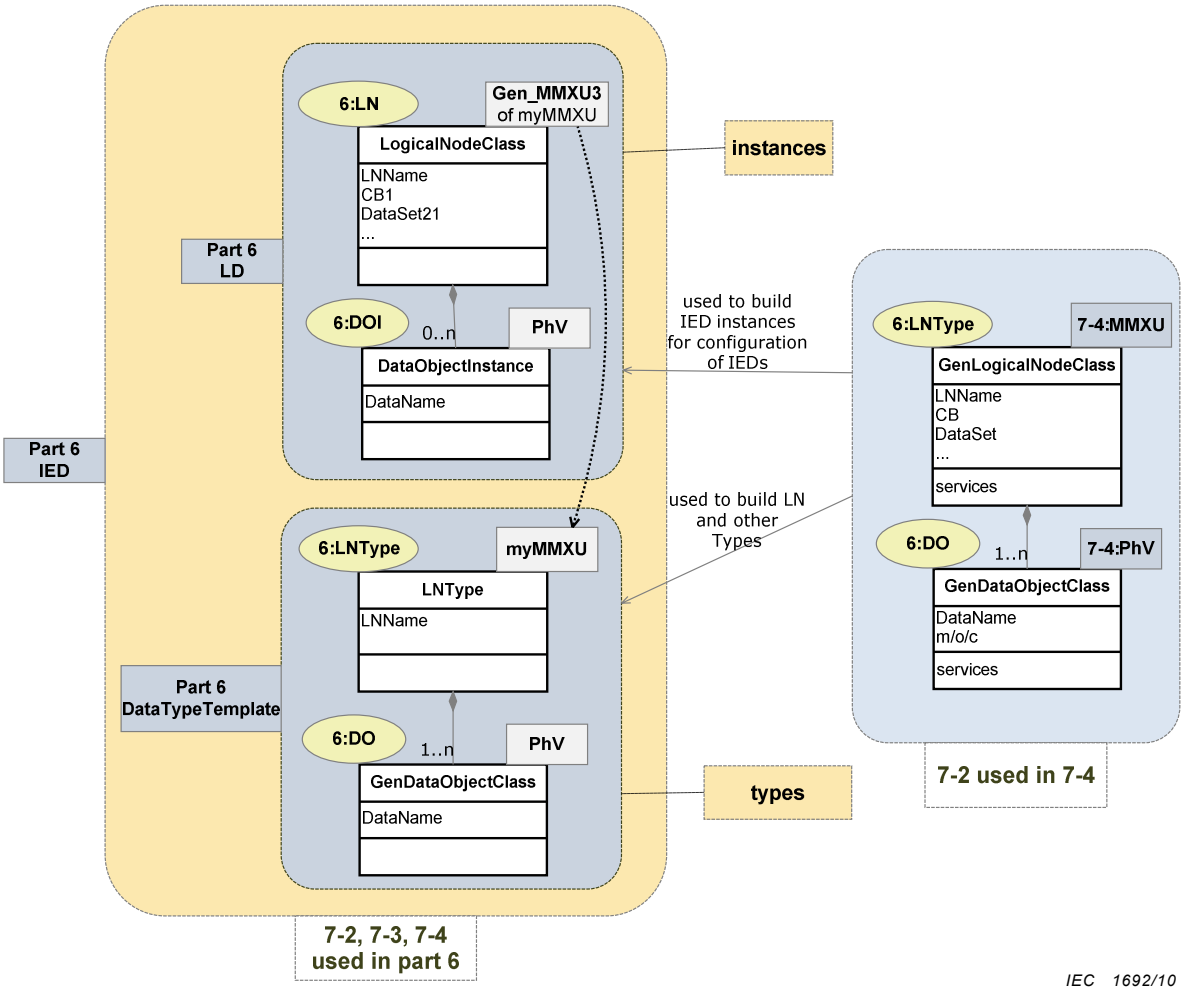


Figure 5 – Data instance model (conceptual)

NOTE The mapping of these instances to application layer protocols like MMS (manufacturing message specification, ISO 9506) are defined in the SCSMs. The logical device is mapped in MMS to a MMS domain class and the logical node and data objects are mapped to MMS NamedVariables as part of a domain.

6 TypeDefinitions

6.1 General

TypeDefinitions shall be composed of BasicTypes (6.1.1) and of CommonACSITypes (6.1.2).

6.1.1 BasicTypes

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

Table 2 – BasicTypes

BasicTypes			
Name	**Value range**	**Remark**	**Used by**
BOOLEAN	0,1 (false, true)		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
INT32	–2 147 483 648 to 2 147 483 647		IEC 61850-7-3 IEC 61850-7-2
INT64	–2**63 to (2**63) –1		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
INT24U	Unsigned integer, 0 to 16 777 215	Only used for TimeStamp type	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
ENUMERATED	Ordered set of values, defined where type is used. Values shall be assigned in the SCSMs.	Custom extensions are allowed	IEC 61850-7-3 IEC 61850-7-2
CODED ENUM	Ordered set of values, defined where type is used. Values shall be assigned in the SCSMs.	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 [a]	The NULL OCTET STRING is implemented by an empty OCTET STRING	IEC 61850-7-3 IEC 61850-7-2
VISIBLE STRING	Max. length shall be defined where type is used [a]	The NULL VISIBLE STRING is implemented by an empty VISIBLE STRING	IEC 61850-7-3 IEC 61850-7-2
UNICODE STRING	Unicode coding is defined in the SCSM. Max. length shall be defined where type is used [a]	The NULL UNICODE STRING is implemented by an empty UNICODE STRING	IEC 61850-7-3
Currency	A currency identification code based on ISO 4217 3-character currency code. The concrete coding shall be defined by the SCSMs.		IEC 61850-7-3
NOTE The data type INT128 is deprecated and replaced by INT64.			
[a] The length suffix shall have the format "...STRINGnn" where "nn" is the length in characters.			

6.1.2 CommonACSITypes

6.1.2.1 General

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

The following types are defined:

- ObjectName (see 6.1.2.2),
- ObjectReference (see 6.1.2.3),
- PHYCOMADDR type (see 6.1.2.4),
- ARRAY type (see 6.1.2.5),
- ServiceError type (see 6.1.2.6),
- EntryID type (see 6.1.2.7),
- Packed list type (see 6.1.2.8),
- TimeStamp type (see 6.1.2.9),
- EntryTime type (see 6.1.2.10),
- TriggerConditions type (see 6.1.2.11),
- ReasonCode (ReasonForInclusion) (see 6.1.2.12).

6.1.2.2 ObjectName

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

Table 3 – ObjectName type

ObjectName type			
Attribute name	**Attribute type**	**Value/value range/explanation**	**Used by**
ObjectName	VISIBLE STRING64	Name of an instance of a class of a single hierarchy level	IEC 61850-7-4 IEC 61850-7-3 IEC 61850-7-2

The constraints defined in Clause 22 on the use of the type ObjectReference shall be applied.

6.1.2.3 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 defined in Table 4.

Table 4 – ObjectReference type

ObjectReference type			
Attribute name	**Attribute type**	**Value/value range/explanation**	**Used by**
ObjectReference	VISIBLE STRING129	ObjectReference comprises the whole path-name of an instance of a class that identifies the instance uniquely. The ObjectReference shall be composed of two parts: up to 64 characters for the LD name followed by one separator "/" followed by up to 64 characters for the path below the LD name. The NULL ObjectReference is an empty ObjectReference (i.e. empty VISIBLE STRING129)	IEC 61850-7-3 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 "[.]" indicates an option.
– The "[. ...]" indicates further names of recursively nested definitions.
– The "(…)" shall indicate an array element

NOTE 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.

The constraints defined in Clause 22 on the use of the type ObjectReference shall be applied.

6.1.2.4 PHYCOMADDR type

The type **phycomaddr** shall represent a physical communication address (for example media access address, priority, and other information) as defined by a SCSM.

6.1.2.5 ARRAY type

The type **array** shall be defined as follows:

ARRAY 0..m OF p

with m ≥ 0
p = **GenCommonDataClass** that does not contain an **array** type or
GenDataAttributeClass or
GenConstructedAttributeClass or
TypeDefinitions (except **ARRAY** type)

shall represent a list of elements numbered from 0 to "m". The type of the elements shall be as specified by "p".

6.1.2.6 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 | no-error	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.

6.1.2.7 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 of the **EntryID** value is a local issue outside the scope of this standard. However, the NULL entryID used in the standard must be the OCTET STRING whose octets have all the value 0 (zero).

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

6.1.2.8 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

6.1.2.9 TimeStamp type

6.1.2.9.1 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 21.

NOTE 1 The TimeStamp type relies on requirements specified in Clause 21. 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.

6.1.2.9.2 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	INT32U	(0…MAX)	M
FractionOfSecond	INT24U	Value = SUM from i=0 to 23 of b_i*2**–(i+1); Order = b_0, b_1, b_2, b_3, …	M
TimeQuality	TimeQuality		M

6.1.2.9.3 TimeStamp attributes

6.1.2.9.3.1 SecondSinceEpoch

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

6.1.2.9.3.2 FractionOfSecond

The attribute **FractionOfSecond** 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 b_i*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.

6.1.2.9.3.3 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		M
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 contains 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. Instead the seconds since start of the epoch are calculated from the current date assuming a constant day length of 86 400 s.

NOTE If a UTC time master clock is used and accessable, LeapSecondsKnown should always be true.

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

ClockNotSynchronized: When set, the attribute **clockNotSynchronized** shall indicate that the time source of the sending device is not synchronized with the external UTC time; otherwise **clockNotSynchronized** is FALSE.

TimeAccuracy: The attribute **TimeAccuracy** shall represent the time accuracy class of the time respective time stamp relative to the external UTC time. The exact meaning depends on the usage of the time stamp respective on the time stamped object. 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 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)

The NULL TimeStamp is a TimeStamp whose fields are all set to 0 (zero).

6.1.2.10 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.

The time base for **EntryTime** shall be GMT. The epoch for EntryTime shall be 01. January 1984 (MJD 40 587).

NOTE 1 The TimeStamp type is used for common data classes in IEC 61850-7-3 and definition of compatible data object classes in IEC 61850-7-4. 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.

6.1.2.11 TriggerConditions type

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

Table 10 – TriggerConditions type

TriggerConditions type			
Attribute name	Attribute type	Value / Value range	M/O/C
	PACKED LIST		M
data-change	BOOLEAN	See Clause 17	M
quality-change	BOOLEAN	See Clause 17	M
data-update	BOOLEAN	See Clause 17	M
integrity	BOOLEAN	See Clause 17	M
general-interrogation	BOOLEAN	See Clause 17	M

NOTE Details on the use of **TriggerConditions** are defined in Clause 17.

6.1.2.12 ReasonCode (ReasonForInclusion)

The values conveyed by ReasonForInclusion shall be a PACKEDLIST as defined in Table 11.

Table 11 – ReasonForInclusion

ReasonForInclusion			
Attribute Name	Attribute Type	Value/Value Range	M/O/C
	PACKEDLIST		
data-change	BOOLEAN	may only be TRUE if TrgOp.dchg = TRUE	M
quality-change	BOOLEAN	may only be TRUE if TrgOp.qchg = TRUE	M
data-update	BOOLEAN	may only be TRUE if TrgOp.dupd = TRUE	M
integrity	BOOLEAN	may only be TRUE if IntgPd is non-zero and TrgOp.integrity = True	M
general-interrogation	BOOLEAN	may only be TRUE if there has been a SetBRCBValues of GI=TRUE and TrgOp.general-interrogation=TRUE	M
application-trigger	BOOLEAN	may only be TRUE if the trigger comes from an application function	AC_LOG_M
AC_LOG_M: The attribute shall only be present when ReasonForInclusion is used within the scope of LOG.			

General-interrogation, integrity and application-trigger reasons for inclusions are mutually exclusive of all other reasons for inclusion (see 17.2.3.2.3.5 and 17.3.3.2.7.3).

7 GenServerClass model

7.1 GenServerClass definition

7.1.1 GenServerClass syntax

The **GenServerClass** shall represent the externally visible behaviour of a device. The **GenServerClass** shall be a composition as defined in Table 12.

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

Table 12 – GenServerClass definition

GenSERVER class		
Attribute name	**Attribute type**	**Value/value range/explanation**
ServiceAccessPoint [1..n]	(*)	(*) Type is SCSM specific
LogicalDevice [1..n]	GenLogicalDeviceClass	
FileSystem [0..1]	FILE-SYSTEM	
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.

7.1.2 GenServerClass attributes

7.1.2.1 ServiceAccessPoint [1..n]

The attribute **ServiceAccessPoint** shall identify a server within the scope of a subnetwork.

NOTE The **ServiceAccessPoint** is an abstraction of an address used to identify the server and a profile of 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.

7.1.2.2 LogicalDevice [1..n]

The attribute **LogicalDevice** shall identify a logical device that is contained in a **GenServer**.

7.1.2.3 FileSystem [0..n]

The attribute **FileSystem** shall identify a File System contained in a **GenServer**.

7.1.2.4 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 8.

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

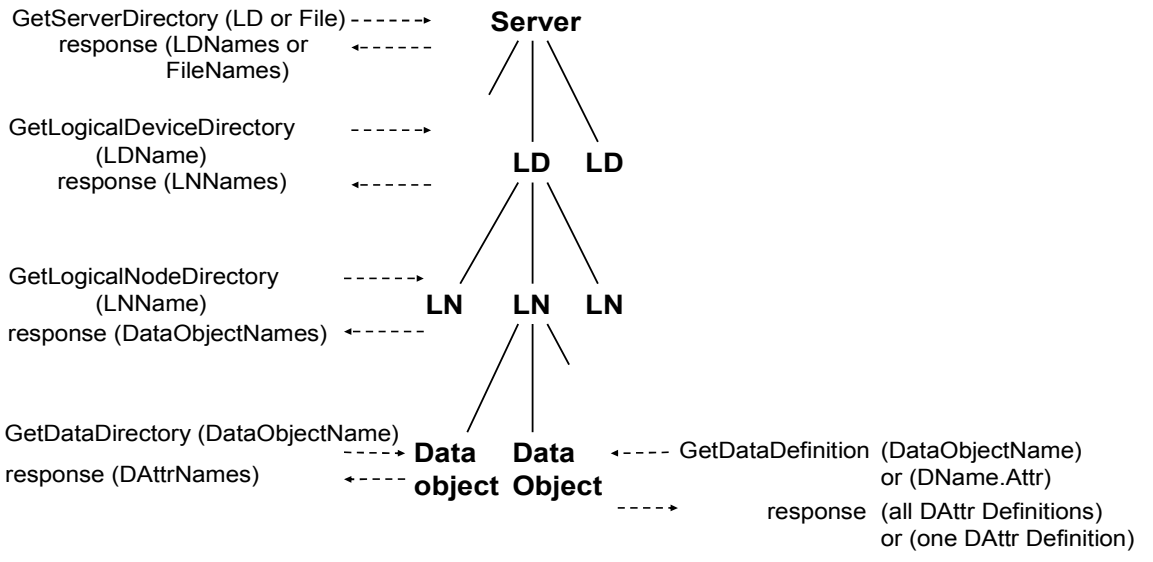
The attribute **MCAppAssociation** shall identify a subscriber with which a server (publisher) maintains a multicast application association.

NOTE Details can be found in Clause 8.

7.2 Server class services

7.2.1 Overview of directory and GetDefinition services

To support self-description of a device, several GetXXDirectory and GetXXDefinition services as shown in Figure 6 are specified in this part of IEC 61850.



IEC 400/03

Figure 6 – 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.

7.2.2 GetServerDirectory

7.2.2.1 GetServerDirectory parameter table

A client shall use the **GetServerDirectory** service to retrieve a list of the names of all logical devices and file systems made visible and thus accessible to the requesting client by the addressed server.

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

7.2.2.2 Request

7.2.2.2.1 ObjectClass

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

– **logical-device**

– **file –system**

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

7.2.2.3 Response+

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

7.2.2.3.1 Reference [0..n]

The parameter **Reference** shall contain the ObjectReference of the logical devices (when ObjectClass is set to LOGICAL-DEVICE) or shall contain the file name(s) present in the file system (when ObjectClass is set to FILE-SYSTEM).

NOTE The **FileName** type is a VISIBLE STRING255 and is defined in 23.1.1.

7.2.2.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned, for example, failed-due-to-server-constraint, parameter-value-inconsistent, or parameter-value-inappropriate.

8 Application association model

8.1 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 are defined in IEC 61850-5.

NOTE Security implementations are defined in the SCSMs.

8.2 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.

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

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

8.3.1.1 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.

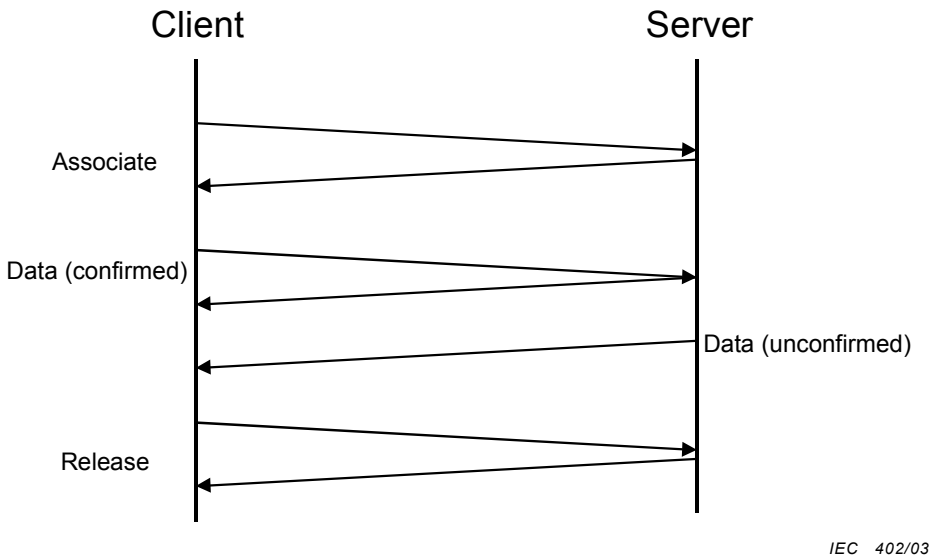


Figure 7 – Normal operation

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

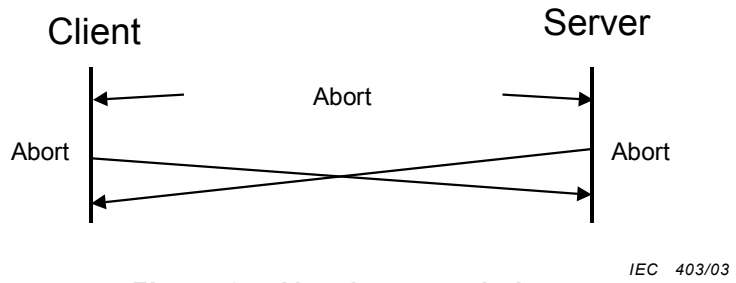


Figure 8 – Aborting association

The **two-party-application-association** (TPAA) class shall be defined as in Table 13.

Table 13 – 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”).		

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

8.3.1.2.1 AssociationId

The attribute **AssociationId** shall define 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.

8.3.1.2.2 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 and password. The details are defined in the SCSMs.

8.3.2 Two-party application association services

8.3.2.1 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

8.3.2.2 Associate

8.3.2.2.1 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

8.3.2.2.2 Request

8.3.2.2.2.1 ServerAccessPointReference

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

8.3.2.2.2.2 AuthenticationParameter

This parameter **AuthenticationParameter** shall define 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.

8.3.2.2.3 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.

8.3.2.2.4 Result

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

8.3.2.2.5 Response–

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

8.3.2.3 Abort

8.3.2.3.1 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

8.3.2.3.2 Request

8.3.2.3.2.1 AssociationId

The parameter **AssociationId** shall define 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.

8.3.2.3.2.2 Reason

The parameter **Reason** shall define 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.

8.3.2.3.3 Indication

8.3.2.3.3.1 AssociationId

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

8.3.2.3.3.2 Reason

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

8.3.2.4 Release

8.3.2.4.1 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

8.3.2.4.2 Request

AssociationId

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

8.3.2.4.3 Response+

8.3.2.4.3.1 AssociationId

The parameter **AssociationId** shall define the association that has been terminated.

8.3.2.4.3.2 Result

The parameter **Result** shall indicate, if the termination of the application association was successful.

8.3.2.4.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned, for example, instance-not-available, parameter-value-inappropriate, parameter-value-inconsistent, failed-due-to-communications-constraint, or failed-due-to-server-constraint.

In the case of a **Release** requested before the completion of (a) pending service(s), the Server shall answer with **Response–**. The application association shall not be terminated.

8.4 MULTICAST-APPLICATION-ASSOCIATION (MCAA) class

8.4.1 MULTICAST-APPLICATION-ASSOCIATION (MCAA) class definition

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.

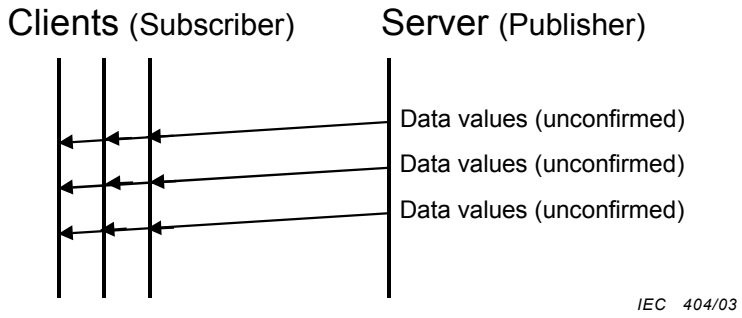


Figure 9 – Principle of multicast application association

The **multicast-application-association** (**MCAA**) shall be as defined in Table 14.

Table 14 – 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”).		

8.4.2 MULTICAST-Application-association (MCAA) class attributes

8.4.2.1 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.

9 GenLogicalDeviceClass model

9.1 GenLogicalDeviceClass definition

9.1.1 GenLogicalDeviceClass syntax

The **GenLogicalDeviceClass** (**GenLD**) shall be a composition of **GenLogicalNodeClass** as defined in Table 15.

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

Table 15 – GenLogicalDeviceClass (GenLD) class definition

GenLOGICAL-DEVICE class		
Attribute name	**Attribute type**	**Value/value range/explanation**
LDName	ObjectName	Instance name of an instance of GenLogicalDeviceClass
LogicalNode [1..n]	GenLogicalNodeClass	IEC 61850-7-4 specifies specialized classes of GenLogicalNodeClass
Services GetLogicalDeviceDirectory		

9.1.2 GenLogicalDeviceClass attributes

9.1.2.1 LDName – logical device name

The attribute **LDName** shall unambiguously identify an instance of a logical device within the scope of a subnetwork.

9.1.2.2 LogicalNode [1..n]

The attribute **LogicalNode [1..n]** shall be a list of all logical nodes of that are contained in a **GenLogicalDeviceClass**.

Each **GenLogicalDeviceClass** shall have one and only one **logical-node-zero** (**LLN0**) and it may have no or several other **LogicalNodes**.

NOTE The details of **LLN0** and other logical node classes are defined in IEC 61850-7-4 for utility automation applications.

9.2 GenLogicalDeviceClass services

9.2.1 GetLogicalDeviceDirectory

9.2.1.1 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.

Parameter name
Request
LDName
Response+
LNReference [1..n]
Response–
ServiceError

9.2.1.2 Request

9.2.1.2.1 LDname – logical device object name

The parameter **LDName** shall contain the object name of a logical device.

9.2.1.3 Response+

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

The parameter **LNReference [1..n]** shall contain ObjectReferences of all logical nodes from the referenced logical device. At least one reference shall be returned according to IEC 61850-7-4: for the logical node zero (**LLN0**).

9.2.1.4 Response–

The parameter **Response–** shall indicate that the service request failed. The appropriate ServiceError shall be returned, for example if the LogicalDevice specified with the LDName of the GetLogicalDevice request does not exist in the server.

10 GenLogicalNodeClass model

10.1 GenLogicalNodeClass definition

10.1.1 GenLogicalNodeClass diagram

The model of the GenLogicalNodeClass is depicted in Figure 10. The GenLogicalNodeClass (for example a class MMXU – measurements of a 3-phase electrical system) comprises several attributes and services. It is an aggregation of data objects (for example, PhV – phase voltages).

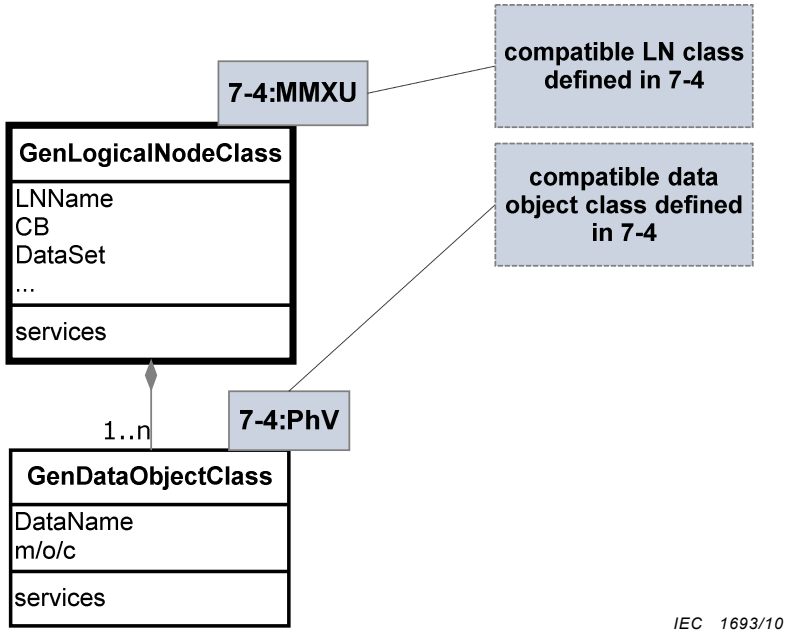


Figure 10 – Basic conceptual model of the GenLogicalNodeClass

The GenLogicalNodeClass details are defined in the following clauses.

10.1.2 GenLogicalNodeClass syntax

The **GenLogicalNodeClass** shall be a composition of **DataObjects**, **DATA-SET**, **BRCB**, **URCB**, **LCB**, **LOG**, **SGCB**, **GoCB**, **GsCB**, **MSVCB**, and **USVCB** as defined in Table 16.

Table 16 – GenLogicalNodeClass definition

GenLogicalNodeClass		
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
DataObject [1..n]	GenCommonDataClass	
DataSet [0..n]	DATA-SET	
BufferedReportControlBlock [0..n]	BRCB	
UnbufferedReportControlBlock [0..n]	URCB	
The following attributes shall only be available if their support is explicitly stated in the definition of a compatible LN class, for example, in IEC 61850-7-4.		
SettingGroupControlBlock [0..1]	SGCB	
Log [0..n]	LOG	
LogControlBlock [0..n]	LCB	
GOOSEControlBlock [0..n]	GoCB	
MulticastSampledValueControlBlock [0..n]	MSVCB	
UnicastSampledValueControlBlock [0..n]	USVCB	
Services GetLogicalNodeDirectory GetAllDataValues		
NOTE IEC 61850-7-4 defines specialized logical node classes – the compatible logical node classes, for example, XCBR representing circuit-breakers.		

The definitions of **GenDataObjectClasses** for the utility automation domain are refined by the definition of specific data objects 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 utility automation-domain-specific logical nodes.

NOTE IEC 61850-7-4 defines further attributes for logical nodes; for example, the mode (behaviour: ON, TEST, TEST-BLOCKED etc.) of the utility automation-specific logical node is defined in IEC 61850-7-4. The state model of a logical node is modelled as a specific data object (named **Mod**).

10.1.3 GenLogicalNodeClass attributes

10.1.3.1 LNName – Logical node name

The attribute **LNName** shall unambiguously identify a logical node within the scope of a logical device.

10.1.3.2 LNRef – Logical node ObjectReference

The attribute **LNRef** shall be the unique path-name of a logical node.

The ObjectReference **LNRef** shall be:

LDName/LNName

10.1.3.3 DataObject [1..n]

The attribute **DataObject** shall be a data object that is contained in a logical node. The **DataObject** details are defined in Clause 11.

NOTE IEC 61850-7-4 defines standardized data object classes.

10.1.3.4 DataSet [0..n]

The attribute **DataSet** shall be a list of all data sets that are contained in a logical node. The **DataSet** details are defined in Clause 13.

10.1.3.5 BufferedReportControlBlock [0..n]

The attribute **BufferedReportControlBlock** shall be a list of all buffered report control blocks that are contained in a logical node. The **BufferedReportControlBlock** details are defined in 17.2.2.

10.1.3.6 UnbufferedReportControlBlock [0..n]

The attribute **UnbufferedReportControlBlock** shall be a list of all unbuffered report control blocks that are contained in a logical node. The **UnbufferedReportControlBlock** details are defined in 17.2.4.

10.1.3.7 Log [0..n]

The attribute **Log** shall be a list of all logs that are contained in a logical node. The **Log** details are defined in 17.3.

10.1.3.8 LogControlBlock [0..n]

The attribute **LogControlBlock** shall be a list of all log control blocks that are contained in a logical node. The **LogControlBlock** details are defined in 17.3.2.

10.1.3.9 SettingGroupControlBlock [0..1]

The attribute **SettingGroupControl** shall be the setting group control block that is contained in a logical node. The **SettingGroupControl** details are defined in Clause 16.

10.1.3.10 GOOSEControlBlock [0..n]

The attribute **GOOSEControlBlock** shall be a list of all GOOSE control blocks that are contained in a logical node. The **GOOSEControlBlock** details are defined in 18.2.

10.1.3.11 MulticastSampledValueControlBlock [0..n]

The attribute **MulticastSampledValueControlBlock** shall be a list of all multicast sampled value control blocks that are contained in a logical node. The **MulticastSampledValueControlBlock** details are defined in 19.2.

10.1.3.12 UnicastSampledValueControlBlock [0..n]

The attribute **UnicastSampledValueControlBlock** shall be a list of all unicast sampled value control blocks that are contained in a logical node. The **UnicastSampledValueControlBlock** details are defined in 19.3.

10.2 GenLogicalNodeClass services

10.2.1 Overview

For **GenLogicalNodeClass**, 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 object contained in a logical node

10.2.2 GetLogicalNodeDirectory

10.2.2.1 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.

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

10.2.2.2 Request

10.2.2.2.1 LNReference

The parameter **LNReference** shall contain the ObjectReference of the logical node.

10.2.2.2.2 ACSIClass

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

The client shall select one identification for a class from the following ACSI class models:

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

NOTE GsCB is deprecated and kept only for backwards compatibility.

10.2.2.3 Response+

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

InstanceName [0..n]

If the ACSI class model requested equals DATA-SET, BRCB, URCB, LCB, LOG, SGCB, GoCB, GsCB, MSVCB, or USVCB, the parameter **InstanceName [0..n]** shall contain all names of all instances of the requested class.

If the ACSI class model requested equals **Data object**, the parameter **InstanceName [0..n]** shall contain all names of the highest level of the data object instance.

NOTE If the data object instance contains a data object (for example data object PhV containing a data object phsA according to the CDC WYE), the name of the underlying data object may be retrieved by the service GetDataDirectory (11.4.4).

In the case where the referenced logical node does not contain the requested ACSI class, the server shall indicate that no object for the requested ACSI class model exists in this logical node.

10.2.2.4 Response–

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

10.2.3 GetAllDataValues

10.2.3.1 GetAllDataValues parameter table

A client shall use the **GetAllDataValues** service to retrieve all data attribute values (having the same **FunctionalConstraint**) of all data objects made visible and thus accessible to the requesting client by the referenced logical node.

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

10.2.3.2 Request

10.2.3.2.1 LNReference

The parameter **LNReference** shall contain the ObjectReference of the logical node.

10.2.3.2.2 FunctionalConstraint [0..1]

The parameter **FunctionalConstraint** (**FC**) shall contain the functional constraint parameter (**FC**) to filter the respective data attributes of all data objects contained in the logical node. The **FC** shall be as defined in 12.3.3.2.

10.2.3.3 Response+

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

10.2.3.3.1 DataAttributeReference [1..n]

The parameter **DataAttributeReference** shall contain the ObjectReference of a data attribute 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 12.6.2.

10.2.3.3.2 DataAttributeValue [1..n]

The parameter **DataAttributeValue** shall contain the value of a data attribute of the data object contained in the referenced logical node. If the parameter **FunctionalConstraint** is present in the service request then only values of those data attributes that have the **FunctionalConstraint** as given in the service request shall be returned.

NOTE The syntax of the value of a data attribute is defined in an SCSM.

10.2.3.4 Response–

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

11 Generic data object class model

11.1 GenDataObjectClass diagram

The conceptual model of the GenDataObjectClass is depicted in Figure 11. Data objects are typed by common data classes from IEC 61850-7-3 (for example, WYE), or 7-2 (for example, CST).

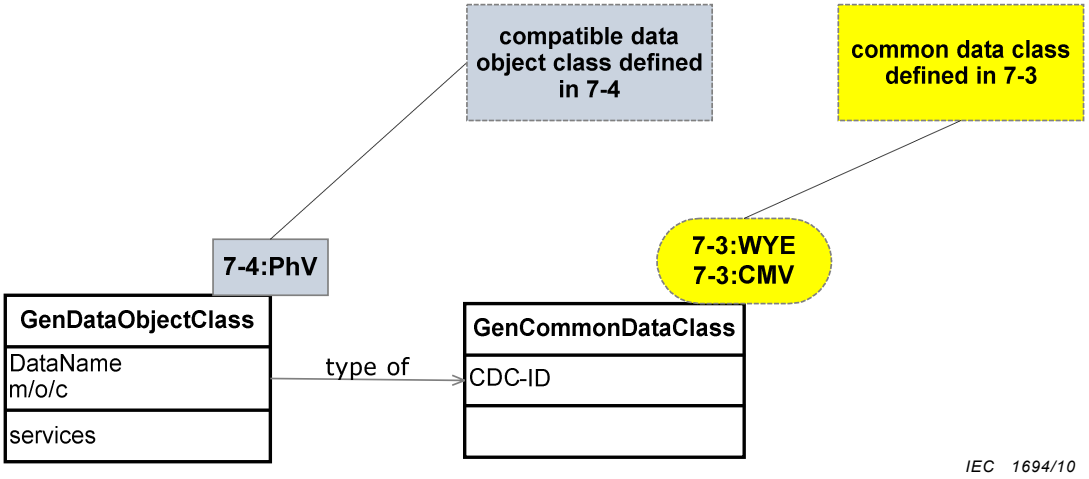


Figure 11 – Basic conceptual class model of the GenDataObjectClass

Data object classes represent meaningful information of applications located in an automation device. The values of data object instances can, for example, be written (**SetDataValues**), read (**GetDataValues**) and / or controlled (Operate). IEC 61850-7-4 specifies a list of common and utility-domain-specific – simple and complex – data object classes, for example, **Pos** for position, **OilFil** for oil filtration. The composition of data object classes in IEC 61850-7-4 is based on common templates (the common data classes, **CDC**) specified in IEC 61850-7-3. The common data classes in IEC 61850-7-3 are built according to the rules defined in this clause. The rules make use of the class diagram for **GenCommonDataClass**.

Any set of data objects instances (or parts of data object 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 writing values to or operating on instances of data objects is outside this part of IEC 61850. IEC 61850-7-3 and IEC 61850-7-4 define many utility-domain-specific data object classes. These definitions provide information on the actions to be taken by the receiving application, for example, changing (operating) the data object **Mod** from ON to TEST changes the state of the respective logical node instance to test mode behaviour as defined in IEC 61850-7-4.

NOTE 2 The client queries values from data objects (respectively **data-set**) from a server using the service **GetDataValues** (respectively **GetDataSetValues**). Services for unsolicited/spontaneous transmission of values from data objects 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.

11.2 GenDataObjectClass syntax

The GenDataObjectClass is a key element in IEC 61850. The class diagram in Figure 11 is intended as an introduction to the formal GenDataObjectClass definition. The syntax shall be as defined in Table 17.

Table 17 – GenDataObjectClass definition

GenDataObjectClass class		
Attribute name	**Attribute type**	**Value/value range/explanation**
DataObjectName	ObjectName	Instance name of an instance of a data object class, for example, **PhV** (1st level), **phsA** (2nd level). The 1st level shall start with an upper case letter, all lower levels with lower case letters.
DataObjectRef	ObjectReference	Path-name of an instance of a data object class, for example, myLD/**MMXU1.PhV** or for example, myLD/**MMXU1.PhV.phsA**
m/o/c	CODED ENUM	Indicates mandatory/optional/conditional
DataObjectType	GenCommonDataClass	For example, **CMV** class of IEC 61850-7-3
Services GetDataValues SetDataValues GetDataDirectory GetDataDefinition		

11.3 GenDataObjectClass attributes

11.3.1 DataObjectName

The attribute **DataObjectName** shall unambiguously identify a data object within the scope of a logical node.

11.3.2 DataObjectRef – data object reference

The attribute **DataObjectRef** shall be the unique path-name of a data object.

The ObjectReference **DataObjectRef** shall be:

LDName/LNName.DataObjectName[.SubDataObjectName[. …]]

NOTE Nesting depends on the CDC as defined in IEC 61850-7-3.

11.3.3 m/o/c

The attribute **m/o/c** of type **coded enum** shall define if a GenSubDataObjectClass, GenDataAttributeClass or DataAttributeClass is mandatory, optional, or conditional. Note that different conditions might apply.

11.3.4 DataObjectType

The **DataObjectType** shall be of type **GenCommonDataClass** as defined in Clause 12.

11.4 GenDataObjectClass services

11.4.1 General definitions and overview

For **GenDataObjectClass**, the following services are defined.

Service	Description
GetDataValues	Retrieve values of a data object contained in a logical node
SetDataValues	Write values of data object contained in a logical node
GetDataDirectory	Retrieve ObjectReferences of all **DataAttributes** contained in a data object
GetDataDefinition	Retrieve definitions of all **DataAttributes** contained in a data object

Excerpts of the four services are depicted in Figure 12.

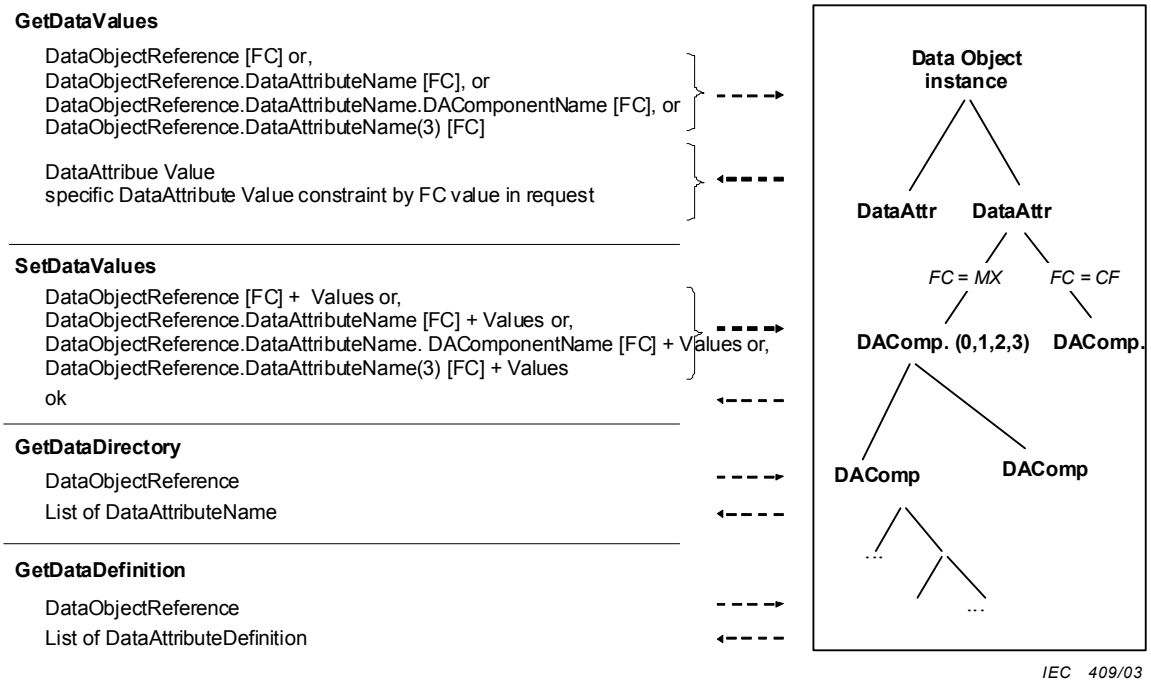


Figure 12 – Excerpt of GenDataObjectClass services

The **GetDataValues** and **SetDataValues** services allow accessing a complete **DataObject** or any part of it.

NOTE The services operate on instances of data objects implemented in a real IED. The description of the services uses parameter names that refer to data objects rather than to the **GenDataObjectClass**.

11.4.2 GetDataValues

11.4.2.1 GetDataValues parameter table

A client shall use the **GetDataValues** service to retrieve values of data attributes of the referenced data objects made visible and thus accessible to the requesting client by the referenced logical node.

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

11.4.2.2 Request

The parameter **Reference** shall define the **functional constrained data** (**FCD**) or **functional constrained data attributes** (**FCDA**) of the data object whose data attribute 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.

11.4.2.3 Response+

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

The parameter **DataAttributeValue [1..n]** shall contain

– the values of all data attributes of a data object referenced by **FCD**; or

– the value of a data attribute referenced by **FCDA**.

NOTE The syntax of the value of a data attribute is defined in an SCSM.

11.4.2.4 Response–

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

11.4.3 SetDataValues

11.4.3.1 SetDataValues parameter table

A client shall use the **SetDataValues** service to set value of data attributes of the referenced data objects made visible and thus accessible to the requesting client by the referenced logical node.

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

11.4.3.2 Request

11.4.3.2.1 Reference

The parameter **Reference** shall define the **functional constrained data** (**FCD**) or **functional constrained data attributes** (**FCDA**) of the data object whose data attribute values are to be set. The **Reference** shall be **FCD** or **FCDA**.

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

11.4.3.2.2 DataAttributeValue [1..n]

The parameter **DataAttributeValue [1..n]** shall contain

– the values of all data attributes of a data object referenced by **FCD**; or

– the value of a data attribute referenced by **FCDA**.

NOTE The syntax of the data attribute is defined in an SCSM.

11.4.3.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. The type for this parameter is SCSM specific.

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 data attribute of the data object requested by the service to the corresponding application.

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

11.4.3.4 Response–

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

The attempt to set a data attribute or an underlying component that is not available shall be interpreted as a service failure.

11.4.4 GetDataDirectory

11.4.4.1 GetDataDirectory parameter table

A client shall use the **GetDataDirectory** service to retrieve the list of all data attribute names of the referenced data object made visible and thus accessible to the requesting client by the referenced logical node.

Parameter name
Request
DataObjectReference
Response+
SubDataObjectName [0..n] or/and DataAttributeName [0..n]
Response–
ServiceError

11.4.4.2 Request

11.4.4.2.1 DataObjectReference – data object reference

The parameter **DataObjectReference** shall contain the object reference of a data object.

11.4.4.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. A successful result shall return the parameter **SubDataObjectName [0..n] DataAttributeName [0..n]**.

The parameter **SubDataObjectName [0...n] DataAttributeName [0..n]**, indicating a possibly empty list of **SubDataObjectName**, followed by a possibly empty list of **DataAttributeName**, shall contain the names of all components one level below the level the given by the **DataObjectReference** of the GetDataDirectory.request.

EXAMPLE A GetDataDirectory.request with (DataReference = myLD/MMXU1.PhV) could return DataObjectNames (phsA, phsB, phsC, neut, net, and res) and DataAttributeNames (angRef, d, dU). A GetDataDirectory.request with (DataObjectReference = myLD/MMXU1.PhV.phsA) could return DataAttributeNames (instCVal, cVal, range, rangeAng, q, t, …).

11.4.4.4 Response–

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

11.4.5 GetDataDefinition

11.4.5.1 GetDataDefinition parameter table

A client shall use the **GetDataDefinition** service to retrieve the complete list of all data attribute definitions of the referenced data object made visible and thus accessible to the requesting client by the referenced logical node. Complete means that the whole structure (the tree with all its branches and leaves) of each data attribute shall be retrieved, i.e., all nested DataAttribute.

Parameter name
Request
DataObjectReference
Response+
SubDataDefinition [0..n] or/and DataAttributeDefinition [0..n]
Response–
ServiceError

11.4.5.2 Request

The parameter **DataReference** shall contain the object reference of the data object.

NOTE An SCSM may bundle several data references into one message.

11.4.5.3 Response+

The parameter **SubDataDefinition [0..n] DataAttributeDefinition [0..n]** shall contain all data object names (data attribute names) and data attribute types (of GenConstructedAttributeClass, BaseType or CommonACSIType) of the first level and of all nested levels below of the referenced data object and the functional constraints of each data attribute (GenDataAttributeClass) where applicable.

11.4.5.4 Response–

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

12 Generic common data class model

12.1 General

The generic common data class model comprises the following definitions:

- generic common data class (see 12.2);
- generic sub data object class (see 12.2);
- generic data attribute class (see 12.3);
- generic constructed attribute class (see 12.4);
- generic sub data attribute class (see 12.5).

The details of these classes are defined in the following subclauses.

12.2 GenCommonDataClass

12.2.1 GenCommonDataClass diagram

The model of the **GenCommonDataClass** is depicted in Figure 13. Data objects in real IEDs or data object classes are typed by common data classes (for example, WYE CDC from IEC 61850-7-3). This clause defines the rules how to define common data classes (CDC) in IEC 61850-7-3.

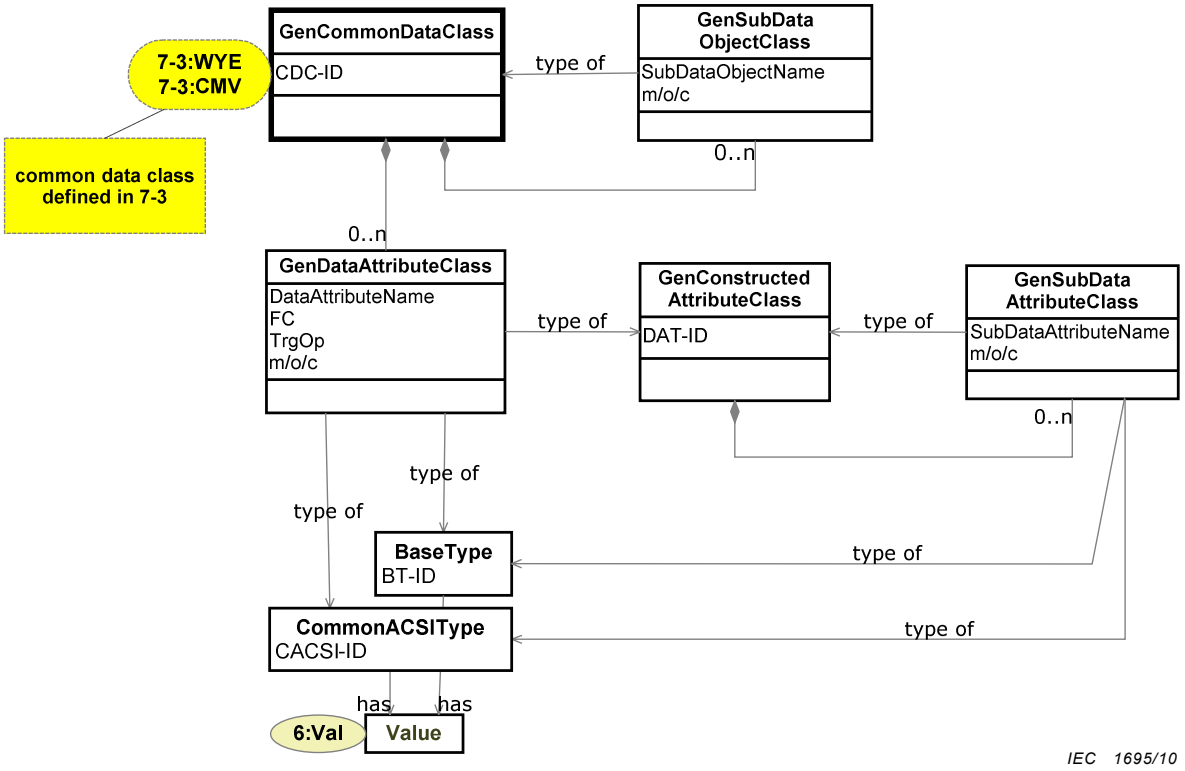


Figure 13 – Class diagram of the GenCommonDataClass

12.2.2 GenCommonDataClass syntax

The conceptual model of the **GenCommonDataClass** is depicted in Figure 14.

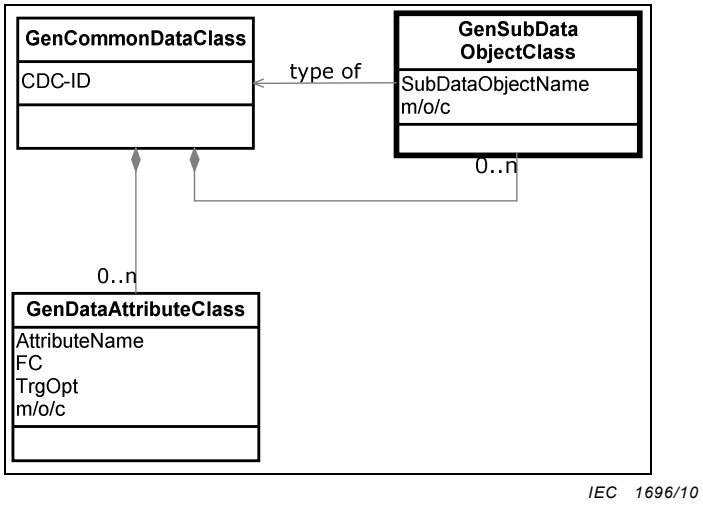


Figure 14 – Conceptual Class diagram of the GenCommonDataClass

The **GenCommonDataClass** shall be as defined in Table 18.

Table 18 – GenCommonDataClass definition

GenCommonDataClass		
Attribute name	**Attribute type**	**Value/value range/explanation**
CDC-ID	Visible String	Use only capital letters; example WYE
Options		
SubDataObject [0..n] or/and	**GenCommonDataClass**	Recursive class definition
DataAttribute [0..n]	GenDataAttributeClass	

12.2.3 GenCommonDataClass attributes

12.2.3.1 CDC-ID – Common data class identifier

The attribute **CDC-ID** shall unambiguously identify a common data class within the scope of either IEC 61850-7-2 or IEC 61850-7-3. It shall use capital letters only.

12.2.3.2 SubDataObjectClass [0..n]

The attribute **SubDataObjectClass [0..n]** shall be a component of the **GenCommonDataClass**. This is a recursive class definition. Observe that instances are not allowed to be recursive, i.e., no GenCommonDataClass is allowed to contain itself somewhere in its lower levels.

12.2.3.3 DataAttribute [0..n]

The attribute **DataAttribute [0..n]** shall be a component of the GenDataAttributeClass. Details are defined in 12.3.

12.3 GenDataAttributeClass

12.3.1 GenDataAttributeClass diagram

The conceptual model of the **GenDataAttributeClass** is depicted in Figure 15.

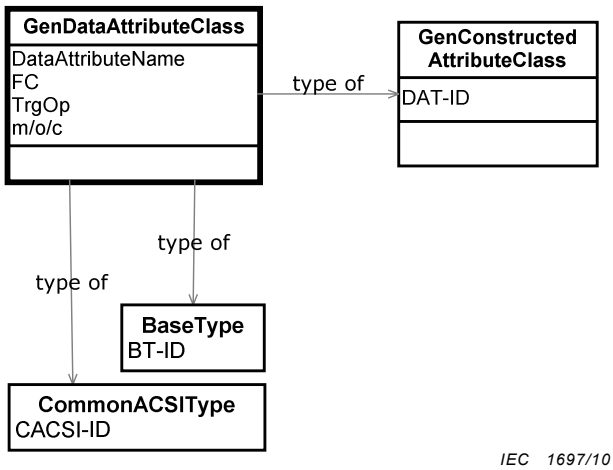


Figure 15 – Class diagram of the GenDataAttributeClass

12.3.2 GenDataAttributeClass syntax

The **GenDataAttributeClass** shall be as defined in Table 19.

Table 19 – GenDataAttributeClass definition

GenDataAttributeClass		
Attribute name	**Attribute type**	**Value/value range/explanation**
DataAttributeName	Visible String	Shall start with a lower case letter; example cVal in the CDC CMV
FunctionalConstraint	**FC**	
TrgOp	**TriggerConditions**	
M/O/C	CODED ENUM	
Type	GenConstructedAttributeClass OR BaseType OR CommonACSIType	

12.3.3 GenDataAttributeClass attributes

12.3.3.1 DataAttributeName

The attribute **DataAttributeName** shall unambiguously identify a data attribute within the scope of a CDC (at the same hierarchy level).

12.3.3.2 FunctionalConstraint (FC)

12.3.3.2.1 General

From an application point of view, the data attributes 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 data attribute.

The **FunctionalConstraint** (**FC**) shall be a property of the **GenDataAttributeClass** characterizing the specific use of the data attribute. The **FunctionalConstraint** is used in the definition of data objects (contained in logical nodes).

NOTE The FunctionalConstraint could be understood as a filter of the data attributes. The common data classes in IEC 61850-7-3 use the FunctionalConstraint values defined in this clause.

The **FunctionalConstraint** is used in various definitions in this part of IEC 61850. The **FunctionalConstraint** shall indicate the services that are allowed to be operated on a specific data attribute. The **FunctionalConstraints** shall be as specified in Table 20.

Table 20 – Functional constraint values

FunctionalConstraint values			
FC	**Semantic**	**Services allowed**	**Initial values/storage/ explanation**
ST	Status information	**DataAttribute** shall represent status information whose value may be read, substituted, reported, and logged but shall not be writeable.	Initial value of the **DataAttribute** shall be taken from the process.
MX	Measurands (analogue values)	**DataAttribute** shall represent measurand information whose value may be read, substituted, reported, and logged but shall not be writeable.	Initial value of the **DataAttribute** shall be taken from the process.
SP	Setting (outside setting group)	**DataAttribute** shall represent setting parameter information whose value is read and may be written. Changes of values shall become effective immediately, and may be reported.	Initial value of the **DataAttribute** shall be as configured; value shall be non-volatile.
SV	Substitution	**DataAttribute** shall represent 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.
CF	Configuration	**DataAttribute** shall represent 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. Value changes may be reported.	Initial value of the **DataAttribute** shall be as configured; value shall be non-volatile.
DC	Description	**DataAttribute** shall represent description information whose value may be written and read.	Initial value of the **DataAttribute** shall be as configured; value shall be non-volatile.
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**. **DataAttributes** with functional constraint SG shall be the current active value (for details see Clause 16). **DataAttributes** with **FC=SG** shall not be writeable.	Initial value of the **DataAttribute** shall be as configured; value shall be non-volatile.
SE	Setting group editable	**DataAttribute** that can be edited by **SGCB** services. Defines the edit buffer for the value sets belonging to attributes with fc=SG.	Value of the **DataAttribute** shall be available after **SelectEditSG** service has been processed.
SR	Service response	**DataAttribute** shall represent data from different process objects with the same tracking object whose values can be used to be reported and logged; the values shall not be writeable. These attributes are used for service tracking (see 15.3.2).	Initial values of the **DataAttribute** are a private issue, for example, all zero (except for time stamp).
OR	Operate received	**DataAttribute** shall represent the result of an Operate request at the data object receiving the Operate request, even if the execution of the Operate is blocked.	Initial value is irrelevant / arbitrary
BL	Blocking	**DataAttribute** is used for blocking value updates	If the value of the **DataAttribute** is volatile then the initial value shall be FALSE, else the value should be as set or configured.
EX	Extended definition (application name space)	**DataAttribute** shall represent an application name space. Application name spaces are used to define the semantic definitions of **LNs**, **data object class**, and **DataAttributes** as specified in 61850-7-3 and IEC 61850-7-4. **DataAttributes** with **FC=EX** shall not be writeable, Note that private extensions of Control Blocks may use the FC EX at SCSM level.	Value of the **DataAttribute** shall be as configured; value shall be non-volatile.

Table 20 *(continued)*

XX	Representing all **DataAttributes** as a service parameter	Shall represent all **DataAttributes** of a data object (of any **FC**) to be accessed, for example, to be written and read. The FC value "XX" shall only be used in the functional 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 may be further constrained by a view or an implementation.			

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

12.3.3.2.2 Functional constrained data (FCD)

The reference to an ordered collection of data attributes of a data object having the same **functional constraint** (**FC**) value shall be called **functional constrained data** (**FCD**). The order of the collection of the **FCD** shall be the order of the appearance of the data attributes in a data object. The reference includes beneath the FC the data object reference.

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

EXAMPLE 1 Figure 19 shows a [**MX**] **FCD** in the second line in the box of the bottom.

12.3.3.2.3 Functional constrained data attribute (FCDA)

A reference to a single data attribute, a sub data object or sub data attribute of a data object having a specific **functional constraint** (**FC**) value shall be called **functional constrained data attribute** (**FCDA**).

EXAMPLE 1 Figure 19 shows a [**MX**] **FCDA** in the fifth line in the box of the bottom. Another example is [MX] myLD/MMXU1.PhV.phsA.

If the data attribute or sub data atttribute is an element of an array, then the **FCDA** shall be accompanied by the NumArrayElement (value between 0 and m according to the instance of the array). The syntax for a **FCDA** of an array element shall be **FCDA(NumArrayElement)**.

EXAMPLE 2 The HWYE CDC in IEC 61850-7-3 uses the ARRAY type. The [MX] FCDA phsAHar(3).cVal references the value of cVal of the array element number 3.

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

12.3.3.3 TrgOp [0..2] – trigger option

The attribute **TrgOp** of type **TriggerConditions** (see 6.1.2.11) shall define the trigger conditions (associated to a data attribute of a data object) that may cause a report to be sent or a log entry to be stored into a log (report model; see Clause 17). If two TrgOp values are defined, the application data object shall define which one shall be taken at the data object instance. The services associated with the **TrgOp** shall be as specified in Table 21.

Table 21 – TrgOp

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 associated data attribute
qchg	quality-change	A report or a log entry shall be generated due to a change of the value of the associated quality data attribute **q**
dupd	data value update	A report or a log entry shall be generated due to updating the value of a data attribute. An updated value may have the same value as the old value. An example is freezing the value of a freezable data attribute updating the value of another data attribute, which could lead to the same value it already has.
NOTE The trigger conditions integrity and general-interrogation of the TriggerConditions type (see Table 10) are used independent of instances of a data object; they can be set from remote by services and thus trigger sending reports or placing log entries into logs. If a data attribute is a composite component, the change or update of a data attribute shall be understood as the change or update of one or more of the primitive components of the data attribute.		

As depicted in Figure 16, the value of a data attribute that provides a specific **TrgOp** (trigger option) shall be monitored for reporting respectively logging if the report control block respectively the log control block has enabled the specific trigger option (**TrgOp**). In the upper example of Figure 16, the **TrgOp** is **dchg**; the **TrgOp** of the data attributes is **dchg** for the first, **dupd** for the second, and **qchg** for the last data attribute. Reports of the first example 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.

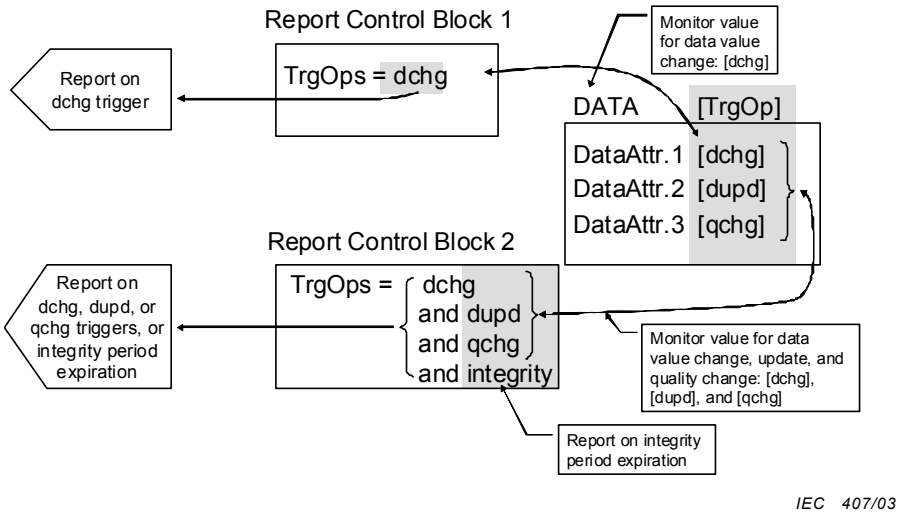


Figure 16 – Relation of TrgOp and Reporting

Data objects whose data attribute 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 17.

12.3.3.4 M/O/C

The attribute **M/O/C** of type **coded enum** shall define if a component of the **GenDataAttributeClass** is mandatory, optional, or conditional.

12.3.3.5 Type

The attribute **Type** shall identify the attribute type to be used for building this component of the CDC to be defined. It shall be a **GenConstructedAttributeClass**, **BaseType** or **CommonACSIType**.

12.4 GenConstructedAttributeClass

12.4.1 GenConstructedAttributeClass diagram

The conceptual model of the **GenConstructedAttributeClass** is depicted in Figure 17.

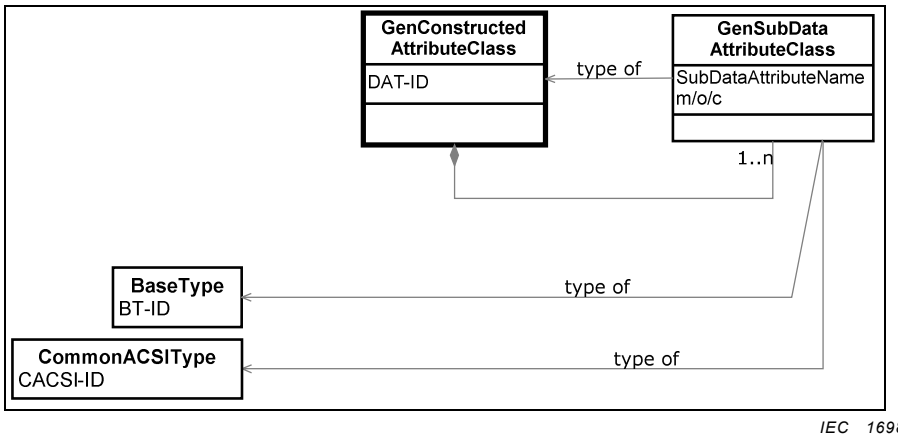


Figure 17 – Class diagram of the GenConstructedAttributeClass

12.4.2 GenConstructedAttributeClass syntax

The **GenConstructedAttributeClass** shall be as defined in Table 22.

Table 22 – GenConstructedAttributeClass definition

GenConstructedAttributeClass		
Attribute name	**Attribute type**	**Value/value range/explanation**
DAT-ID	Visible String	Example: attribute instCVal (DAT-ID=Vector) of CDC CMV.
SubDataAttribute [1..n]	GenSubDataAttrributeClass	

12.4.3 GenConstructedAttributeClass attributes

12.4.3.1 DAT-ID – Data attribute class identifier

The attribute **DAT-ID** shall unambiguously identify a common data attribute type within the scope of IEC 61850-7-2 and IEC 61850-7-3.

12.4.3.2 SubDataAttribute [1..n]

The attribute **SubDataAttributeClass [1..n]** shall be a component of the **GenSubDataAttributeClass**. Details are defined in 12.5.

12.5 GenSubDataAttributeClass

12.5.1 SubDataAttributeClass diagram

The conceptual model of the **SubDataAttributeClass** is depicted in Figure 17.

12.5.2 SubDataAttributeClass syntax

The **SubDataAttributeClass** shall be as defined in Table 23.

Table 23 – GenSubDataAttributeClass definition

GenSubDataAttributeClass		
Attribute name	**Attribute type**	**Value/value range/explanation**
SubDataAttributeName	Visible String	Example mag of the instCVal of the CDC CMV
m/o/c	CODED ENUM	
Type	GenConstructedAttributeClass OR BaseType OR CommonACSIType	Identified by DAT-ID, for example, mag (Type=AnalogueValue) of the instCVal of the CMV CDC

12.5.3 GenSubDataAttributeClass attributes

12.5.3.1 SubDataAttributeName

The attribute **SubDataAttributeName** shall unambiguously identify a sub data attribute component within the scope of a common data class (at the same hierarchy level). The name shall start with a lower case letter (some predefined exceptions exist).

12.5.3.2 m/o/c

The attribute **m/o/c** of type **coded enum** shall define if a **SubDataAttributeClass** is mandatory, optional, or conditional.

12.5.3.3 Type

The attribute **Type** shall identify the constructed attribute class to be used for building this component of the common data class. The type shall be defined by GenConstructedAttributeClass, BaseType, or CommonACSIType.

12.6 Referencing data objects and their components

12.6.1 General

The following clauses provide:

- The general syntax of referencing data objects and the underlying components of the hierarchical structure (see 12.6.2),
- A list of the crucial base types and their relation (see 12.6.3), and
- An example of a complex structure of a data object (see 12.6.4).

12.6.2 Reference syntax

The syntax of the various references shall be as follows:

If the SubDataObject is not an Array element: The ObjectReference containing the path through all naming levels of a data object shall be:

LDName/LNName.
DataObjectName[.SubDataObjectName[. ...]].DataAttributeName[(NumArrayElement)]
[.SubDataAttributeName[. ...]]

If the SubDataObject is an Array element: The ObjectReference containing the path through all naming levels of a data object shall be:

LDName/LNName.
DataObjectName[.SubDataObjectName(NumArrayElement)[. ...]].
DataAttributeName[.SubDataAttributeName[. ...]]

Observe that the (NumArrayElement) shall appear at maximum once in the whole path, either at a SubDataObjectName or at a DataAttributeName. This restricts the data model to one-dimensional ARRAYs within each CDC.

The ObjectReference for a data attribute shall be one of the two definitions:

If the DataAttribute is not an Array element: The ObjectReference **DataAttributeReference** shall be:

LDName/LNName.
DataObjectName[.SubDataObjectName[. ...]].DataAttributeName

If the DataAttribute is an Array element: The ObjectReference **DataAttributeReference** shall be:

LDName/LNName.
DataObjectName[.SubDataObjectName[. ...]].DataAttributeName(NumArrayElement)

NOTE 1 Nesting depends on the concrete definition of a data object.

NOTE 2 In each path within a data object, there is one and only one data attribute (level). Only data attributes have the functional constraint (FC) and trigger option (TrgOp).

The NumArrayElement (value between 0 and m according to the instance of the array) shall be the array element number in case a single array element is to be referenced.

12.6.3 Base types and their relation

A data object is defined as a list of data attributes and optionally a list of SubDataObjects (see Figure 18). Data attributes are named and structured. The attribute names are shown on the right in the left box (db, instMag, …). Each of the attributes has a well defined structure. The structure is defined in the corresponding data attribute type.

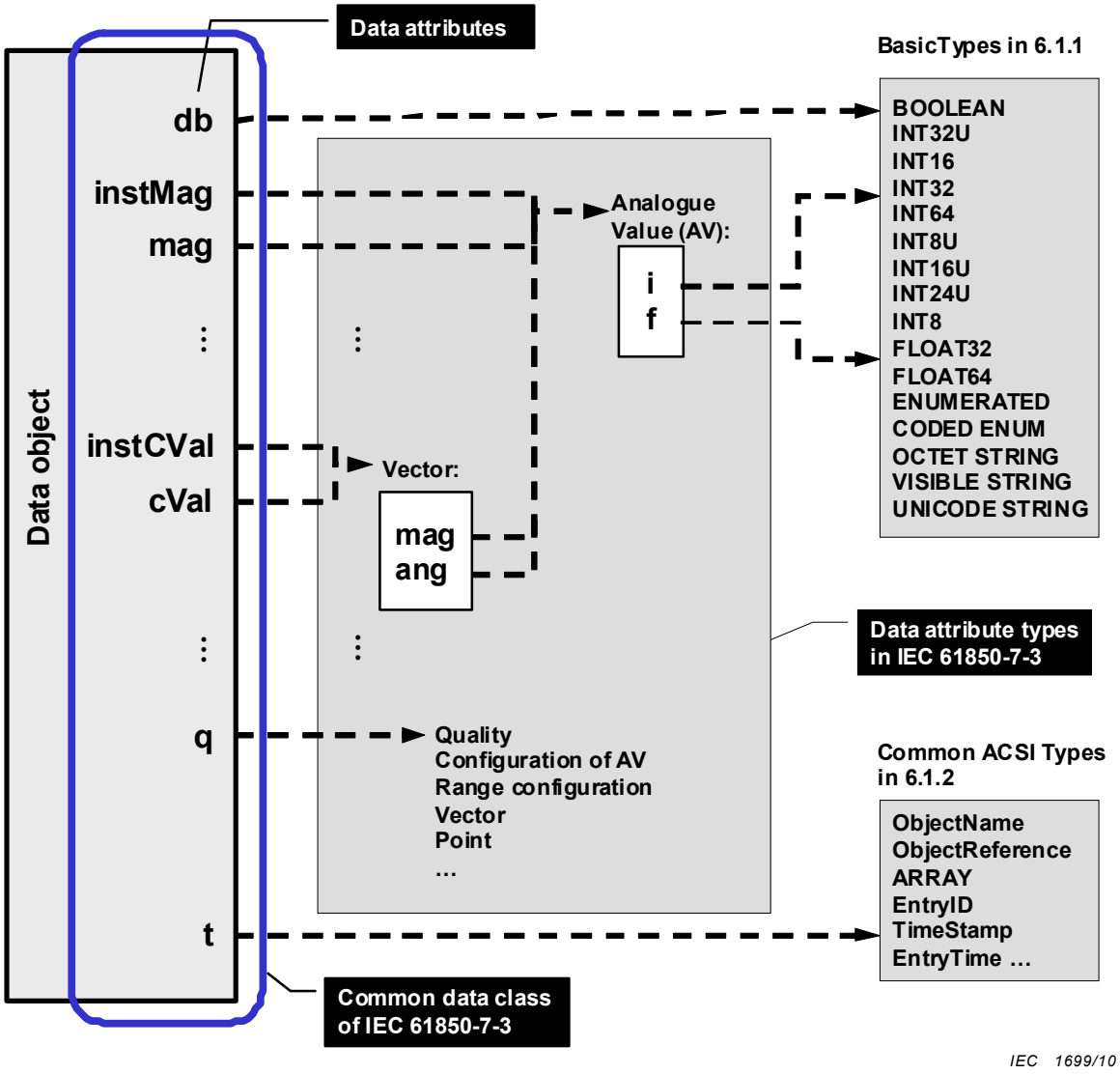


Figure 18 – Relation of types (example)

12.6.4 Example of using references

Figure 19 depicts an excerpt of a data object instance (contained in the logical node MMXU1). The instance of the logical node with the name MMXU1 (instantiated from the compatible logical node class **MMXU defined** in IEC 618590-7-4) is composed of the instance of the data object phase voltage named **PhV** (instantiated from the CDC **WYE**), which is composed of phase A voltage **phsA** (instantiated from CDC **CMV**), which is composed of complex value **cVal** (of type **Vector**), which is composed of voltage **mag** (of type **AnalogueValue**), which is composed of floating-point value **f** (of type **FLOAT32**). The data attribute has additionally the functional constraint **FC** = **MX** (measurand) and the trigger option **TrgOp** = **dchg** (data-change).

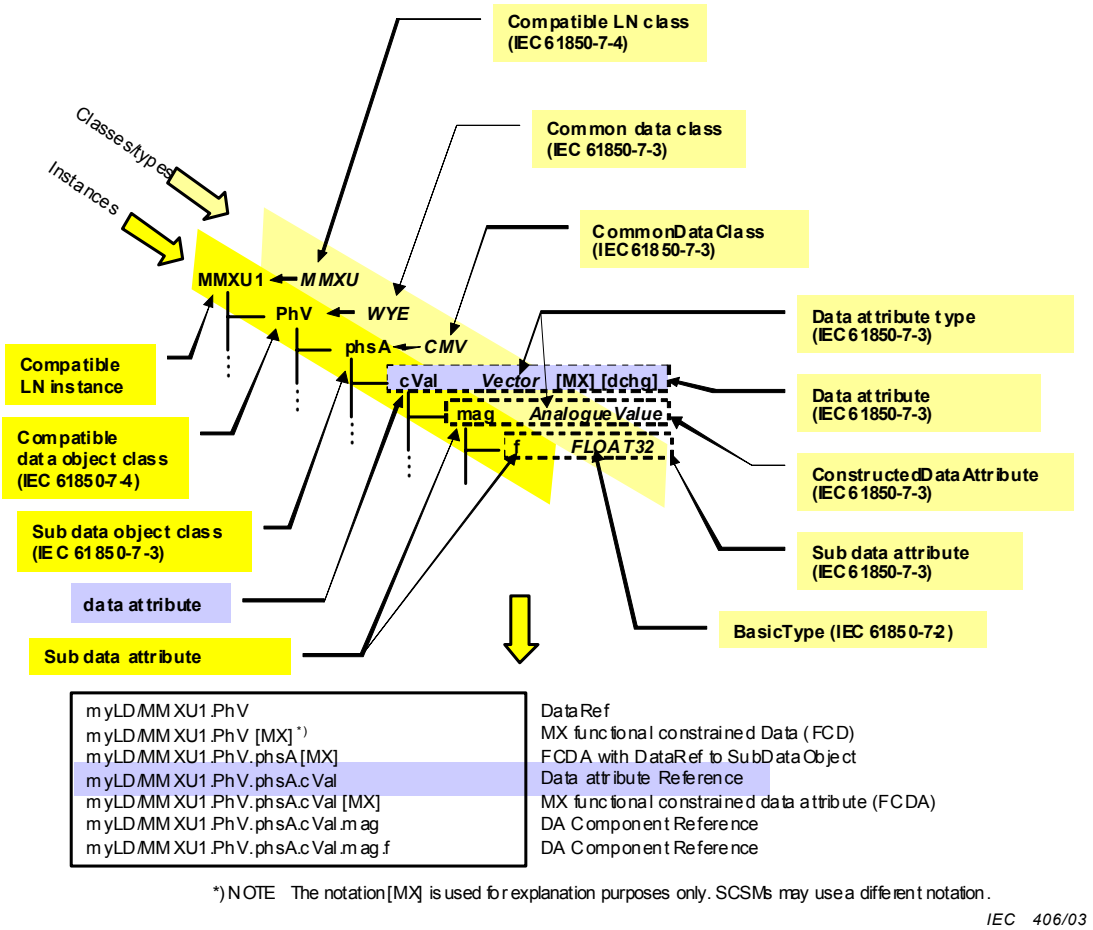


Figure 19 – Example of a data object

13 DATA-SET class model

13.1 General

A **data-set** is an ordered group of **ObjectReferences** of functional constraint **DataObjects**, functional constraint **SubDataObjects**, functional constraint **DataAttributes**, and functional constraint **SubDataAttributes**, in short of **FCDs** or **FCDAs** (see 12.3.3.2), organized as a single collection for the convenience of the client. The references are called the members of the data set. 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 **DataObjects**, **SubDataObjects**, **DataAttributes**, and **SubDataAttributes** need to be transmitted. This capability thus permits more efficient use of the communications bandwidth.

NOTE 1 The membership and order of the **DataObjects**, **SubDataObjects**, **DataAttributes**, and **SubDataAttributes** in a **DATA-SET** can be retrieved with the **GetDataSetDirectory** service. The persistent existence of data objects and DataAttributes 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 different models of spontaneous sending such as reporting, logging, and **GOOSE**. **Data-sets** are used, for example, to define the values of **DataObjects, SubDataObjects**, **DataAttributes**, and **SubDataAttributes** 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 **DataObject**, **DataAttribute** 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 20). 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.

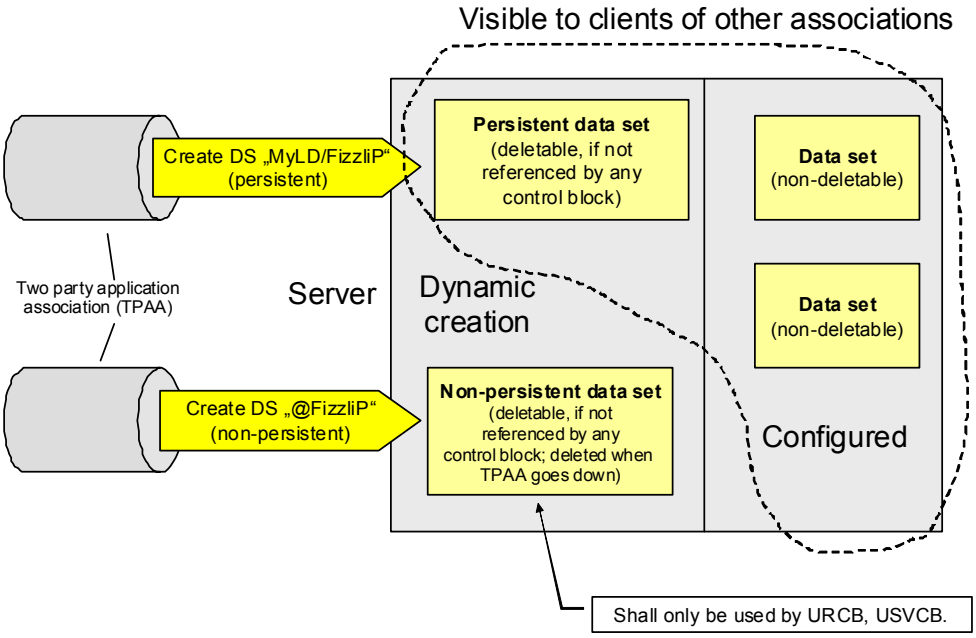


Figure 20 – Dynamic creation of data set instances

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 any 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).

13.2 DATA-SET class definition

13.2.1 DATA-SET class syntax

The **data-set** shall have the structure as defined in Table 24.

Table 24 – 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]	(*)	(*) Functional constrained data (**FCD**) or functional constrained data attribute (**FCDA**)
Services GetDataSetValues SetDataSetValues CreateDataSet DeleteDataSet GetDataSetDirectory		

13.2.2 DATA-SET class attributes

13.2.2.1 DSName

The attribute **DSName** shall unambiguously identify **data-set** within the scope of a **logical-node** or within a two-party-application-association.

13.2.2.2 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**

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

The attribute **DSMemberRef** shall define the functional constrained data (**FCD**) or functional constrained data attribute (**FCDA**). Any data object on the same server as the data set definition can be referenced by an **FCD** or **FCDA**.

An IED which claims to support dynamic creation of datasets (CreateDataSet) shall be able to receive (as a server), and process (as a server) any valid FCD or FCDA definition contained in the CreateDataSet request. An IED which claims to support configuration of datasets (via SCL) shall be able to process (as a server or as a client) any valid FCD or FCDA definition contained in the corresponding SCL file.

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

NOTE A **data-set** does not contain data objects. A **data-set** contains references, the functional constrained data (**FCD**) or functional constrained data attribute (**FCDA**). A **data-set** may contain references to functional constrained data (**FCD**) or functional constrained data attribute (**FCDA**) contained in different **logical-node**s.

13.3 DATA-SET class services

13.3.1 Overview

For **data-set**, the following services are defined.

Service	Description
GetDataSetValues	Retrieve the values of all data objects referenced by the members of the data set
SetDataSetValues	Write the values of all data objects referenced by the members of the data set
CreateDataSet	Create a data set by providing the **FCD** (**FCDA**) references
DeleteDataSet	Delete a data set
GetDataSetDirectory	Retrieve **FCD (FCDA)** references of all members referenced in a data set

13.3.2 GetDataSetValues

13.3.2.1 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**.

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

13.3.2.2 Request

The parameter **DataSetReference** shall define 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**.

13.3.2.3 Response+

The parameter **DataSetReference** of the request shall be mirrored in the response.

The parameter **DataSetMemberValue [1..n]** shall contain values of a member of the **data-set**. The value may be simple or complex depending on the definition of the member.

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

NOTE The syntax of the value is defined in an SCSM.

13.3.2.4 Response–

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

13.3.3 SetDataSetValues

13.3.3.1 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**.

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

13.3.3.2 Request

13.3.3.2.1 DataSetReference – data set ObjectReference

The parameter **DataSetReference** shall define 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**.

13.3.3.2.2 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 object may be simple or complex depending on the definition of the data object. For complex structures, the values of all **DataAttributes** of all nesting levels shall be contained.

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

13.3.3.3 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.

13.3.3.4 Result

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

13.3.3.5 Response–

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

13.3.4 CreateDataSet

13.3.4.1 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 functional constrained data (**FCD**) or functional constrained data attribute (**FCDA**) made visible and thus accessible to the requesting client.

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

13.3.4.2 Request

13.3.4.2.1 DataSetReference – data set ObjectReference

The parameter **DataSetReference** shall define 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**.

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

The parameter **DSMemberRef [1..n]** shall define all functional constrained data (**FCD**), functional constrained data attributes (**FCDA**) of a data object.

13.3.4.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. If one of the members to be defined is not available to that client then the service shall fail.

13.3.4.4 Response–

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

13.3.5 DeleteDataSet

13.3.5.1 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.

Parameter name
Request
DataSetReference
Response+
Response–
ServiceError

13.3.5.2 Request

13.3.5.2.1 DataSetReference – data set ObjectReference

The parameter **DataSetReference** shall define 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**.

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 GetDataSetDirectory

13.3.6.1 GetDataSetDirectory parameter table

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

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

13.3.6.2 Request

The parameter **DataSetReference** shall define 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**.

13.3.6.3 Response+

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

The parameter **DSMemberRef [1..n]** shall contain the **ObjectReferences** of the members of the **data-set**.

NOTE The syntax of the **DSMemberRef** 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.

14 Service tracking

14.1 General

The reporting and logging functions based on the process and function related data model as defined in IEC 61850-7-4 and IEC 61850-7-3 allow to track what happens in the process. For tracking what happens communicationwise in the system itself, this part of IEC 61850 provides the possibility to track all services, even those with negative responses. For this purpose the services are classified as follows:

– Control block related services (see 15.3);
– Command related services (see 20.6);
– Other services (see 14.2).

The base concept consists in defining within the data model where to store the values of the service parameters used by a service. Therefore, this part of IEC 61850 defines common data classes for each type of service to be reported respectively logged. For a given **Server**, there shall be a single data object instance (tracking data object) available in the object directory for each kind of service, that will mirror the value of the service parameters and its acceptance by the server. This allows that a service can be logged or reported to any client, as soon as the tracking data object is a **data-set** member of the **data-set** associated to a **LCB** or to a **BRCB / URCB**.

No (0) or one (1) instance of a needed tracking data object class (derived from the common service tracking data class) shall be defined in a logical device. IEC 61850-7-4 defines the data object name per tracking CDC.

The tracking concerns only the data and services available in the abstract services, not any extensions done within a SCSM.

14.2 Common service tracking (CST)

The common service tracking common data class CST shall be as defined in Table 25. It offers a logistic for tracking any service, and is the base from which all other service tracking classes are derived.

Table 25 – Common service tracking common data class (CST) definition

CST Class						
Attribute name	**Attribute type**	**FC**	**TrgOp**	**r/w**	**Value/Value range**	**M/O/C**
DataName	Inherited from Data Class					
Specific to the CST						
objRef	ObjectReference	SR	dupd	r	Reference of the object that is used in the tracking: either a control block that is being accessed or a control object that is being controlled.	M
serviceType	ServiceType	SR		r	Type of the tracked service – see Table 26	M
errorCode	ServiceError	SR		r	See error associated to the service that is specified by serviceType; value *no-error* for successful service	M
originatorID	OCTET STRING64	SR		r	Originator of the service	O
t	TimeStamp	SR		r	TimeStamp of the completion of the service.	M
d	VISIBLE STRING255	DC		r		O
dU	UNICODE STRING255	DC		r		O
cdcNs	VISIBLE STRING255	EX		r		AC_DLNDA_M
cdcName	VISIBLE STRING255	EX		r		AC_DLNDA_M
dataNs	VISIBLE STRING255	EX		r		AC_DLN_M

The CDC for common service tracking shall be inherited by all special service tracking CDCs.

The attribute **objRef** of type ObjectReference references the object that is being accessed by the service. For control block related services, this is the reference of the control block whose attributes shall be tracked. For control services it is the reference of the controlled data object.

The attribute **originatorID** of type OCTET STRING64 shall identify the client that issued the corresponding service.

The attributes **t**, **d**, **dU**, **cdcNs**, **cdcName**, and **dataNs** shall be as defined in IEC 61850-7-3.

The attribute ServiceType shall be as defined in Table 26.

Table 26 – ServiceType type

ServiceType definition			
Attribute name	**Attribute type**	**Value /value range/explanation**	**Used by**
serviceType	ENUMERATED	GetServerDirectory | Associate | Abort | Release | GetLogicalDeviceDirectory | GetAllDataVaues | GetDataValues | SetDataValues | GetDataDirectory | GetDataDefinition | GetDataSetValues | SetDataSetValues | CreateDataSet | DeleteDataSet | GetDataSetDirectory | SelectActiveSG | SelectEditSG | SetEditSGValue | ConfirmEditSGValues | GetEditSGValue | GetSGCBValues | Report | GetBRCBValues | SetBRCBValues | GetURCBValues | SetURCBValues | GetLCBValues | SetLCBValues | QueryLogByTime | QueryLogAfter | GetLogStatusValues | SendGOOSEMessage | GetGoCBValues | SetGoCBValues | GetGoReference | GetGOOSEElementNumber SendMSVMessage | GetMSVCBValues | SetMSVCBValues | SendUSVMessage | GetUSVCBValues | SetUSVCBValues | Select | SelectWithValue | Cancel | Operate | CommandTermination |TimeActivatedOperate | GetFile | SetFile | DeleteFile | GetFileAttributeValues | TimeSynchronization | InternalChange	IEC 61850-7-2

15 Modelling of control block classes

15.1 General

Control blocks as defined for reporting (17.2), logging (17.3), GSE (Clause 18) and SV (Clause 19) are modelled similar to data object classes. There is however a fundamental difference between data objects and control blocks. Data objects are used to interface to application level functions, while above-mentioned control blocks configure the communication services. The configuration language (SCL) separates the data objects for the application information completely from the communication service models (defined as control blocks).

This clause introduces a general model for control blocks (see 15.2), as used later on for service definitions of control block based services. Further it models control block attributes and service related attributes into common data classes, which allow using a data model for service tracking by providing service related events. The events can be monitored by using the already defined reporting and logging services (see 15.3). This reporting and logging of service related events is called service tracking.

15.2 Control block class models

Control block classes shall be defined using the table notation as shown in Table 27.

Table 27 – CB class definition

CB class			
Attribute name	**Attribute type**	**r/w**	**Value/value range/explanation**
CBName	ObjectName	r	Instance name of an instance of CB
CBRef	ObjectReference	r	Path-name of an instance of CB
Specific to report handler			
Attribute 1	Type for Attribute 1	r/w	
…	…	…	
Attribute n	Type for Attribute n	r/w	
Services Service1 Service2 …			

15.2.1 Control block attributes

15.2.1.1 CBName

The attribute **CBName** shall be the name of the **CB** that unambiguously identifies the **CB** within a logical node.

15.2.1.2 CBRef

The attribute **CBRef** shall be the unique path-name of a **CB**.

The **ObjectReference CBRef** shall be:

LDName/LNName.CBName

15.2.1.3 Attribute 1 to Attribute n

Further attributes as required for a special control block.

15.2.2 Control block services

Services as required for a special control block.

15.2.3 Attribute type

The **Attribute type** shall define the data type of a control block attribute.

15.3 Control block tracking services

15.3.1 General

The control block tracking services provide a means to keep track of issued services on or related to the control blocks. Control block value changes may be caused by services setting a specific value or by internal (local) events in the server.

NOTE The control block tracking service replaces the originally intended use of the FC and TrgOp of the control block attributes of the edition 1. The tracking service provides the same possibilities in a more consistent way, and with additional features. The model can be used to track any service request.

The concept of the control block service tracking is depicted in Figure 21. The base concept is to expose the attributes of any control block by a special data object that represents the control

block attributes and further information (BrcbTrk in our example). On a change of a value of a control block attribute we may log or report the data value of the special data object. This approach does not require any change in the service models nor in the configuration language. The control block definitions are even simpler than in edition 1.

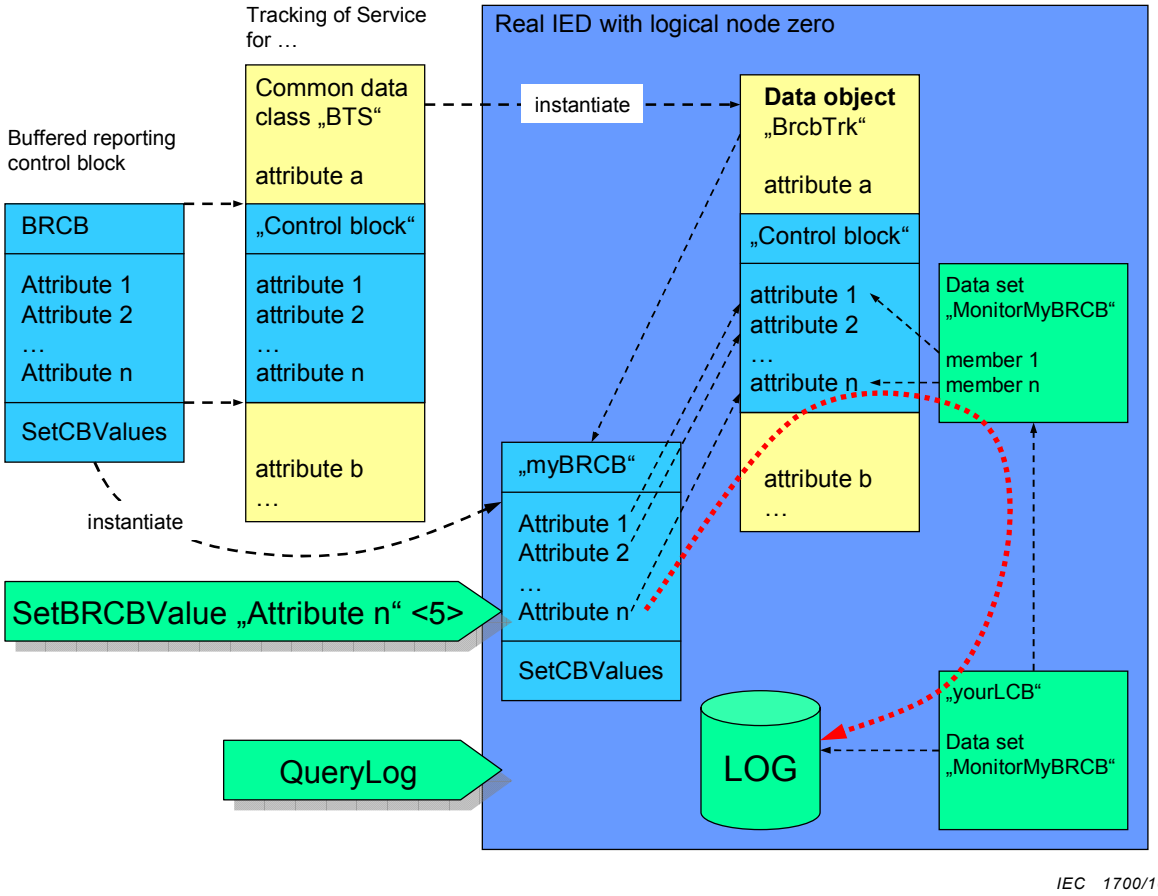


Figure 21 – Control block service mapping

15.3.2 Common data classes for control block service tracking

15.3.2.1 General

The specific Common data classes (CDC) for control block service tracking are defined in conjunction with the control block model, though the control block details are defined later.

NOTE To fully understand the service tracking, it is required to know the basics of the control blocks.

The common data classes for control block service tracking are:

- Buffered report Tracking Service (BTS) (see 15.3.2.2);
- Unbuffered report Tracking Service (UTS) (see 15.3.2.3);
- Log control block Tracking Service (LTS) (see 15.3.2.4);
- GOOSE control block Tracking Service (GTS) (see 15.3.2.6);
- MSVCB Tracking Service (MTS) (see 15.3.2.7);
- USVCB Tracking Service (NTS) (see 15.3.2.8);
- SGCB Tracking Service (STS) (see 15.3.2.9).

All these tracking related CDCs inherit the attributes from the base tracking CDC CST (14.2) and add the relevant attributes from the control blocks. To better distinguish these from the

control block attributes, the attributes within the tracking CDCs start with a lower case letter, while those used in the control block start with an upper case letter.

No (0) or one (1) instance of the data object class (derived from the above CDCs) shall be defined in a logical device. IEC 61850-7-4 defines a data object name per CDC. The service tracking function for a CB class can use a single instance of such a data object to drive the reporting or logging process for all control block objects of the same class in the order of their events.

15.3.2.2 Tracking of service for buffered reporting – Buffered report tracking service (BTS)

The tracking of service for buffered reporting (buffered report tracking service - **BTS**) shall be as defined in Table 28. It offers the logistic for tracking any service that is dedicated to a buffered report control block access. The **BTS CDC** shall be used by data objects for service tracking of the access of a **BRCB** instance. It shall inherit all attributes of the **CST CDC**.

Table 28 – Buffered report tracking service (BTS) definition

BTS Class						
Attribute name	**Attribute type**	**FC**	**TrgOp**	**r/w**	**Value/value range**	**M/O/C**
Shall inherit all the data attributes of the **CST CDC**						
Specific to the BTS						
rptID	VISIBLE STRING129	SR		r	Service parameter ReportIdentifier is mapped to rptID	M
rptEna	BOOLEAN	SR		r	Service parameter ReportEnable is mapped to rptEna	M
datSet	ObjectReference	SR		r	Service parameter DatSetReference is mapped to datSet	M
confRev	INT32U	SR		r	Service parameter ConfigurationRevision is mapped to confRev	M
optFlds	PACKED_LIST	SR		r	Service parameter OptionalFields is mapped to optFlds	M
bufTm	INT32U	SR		r	Service parameter BufferTime is mapped to bufTm	M
sqNum	INT16U	SR		r	Attribute SqNum of BRCB is mapped to sqNum	M
trgOps	TriggerConditions	SR		r	Service parameter TriggerOptionsEnabled is mapped to trgOps	M
intgPd	INT32U	SR		r	Service parameter IntegrityPeriod is mapped to intgPd	M
gi	BOOLEAN	SR		r	Service parameter GeneralInterrogation is mapped to gi	M
purgeBuf	BOOLEAN	SR		r	Service parameter PurgeBuf is mapped to purgeBuf	M
entryID	EntryID	SR		r	Service parameter EntryIdentifier is mapped to entryID	M
timeOfEntry	EntryTime	SR		r	Service parameter TimeOfEntry is mapped to timeOfEntry	M
resvTms	INT16	SR		r	Service parameter ReserveTimeSecond is mapped to resvTms	O

15.3.2.2.1 ServiceType = SetBRCBValues

The attributes **rptID**, **rptEna**, **datSet**, **optFlds**, **bufTm**, **trgOps**, **intPd**, **purgeBuf**, **entryID**, **resvTms** mirror the service parameters of the service **SetBRCBValues** of the **BRCB** referenced by the attribute **objRef**. The attribute **errorCode** returns the error that occurred during the service, or *no-error*, if it was successful. The attribute **originatorID** contains the ID of the client. In case one of the attributes of the **BRCB** is not written while using the **SetBRCBValues** service (for example **ReportIdentifier** [0..1], …), the value that is returned in the **BTS** reflects the value that would have been returned by a **GetBRCBValues** performed on the **BRCB** referenced by the attribute **objRef** at the completion of the **SetBRCBValues**.

15.3.2.2.2 ServiceType = InternalChange

The attributes **RptEna** (report enabled, at loss of association with the Client), and **ResvTms** (at expiration of Reservation) of the control block shall be tracked.

The value that is returned in the **BTS** reflects the value that would have been returned by a **GetBRCBValues** performed on the **BRCB** referenced by the attribute **objRef** at the completion of the **InternalChange**.

15.3.2.2.3 ServiceType = Report

Shall not be tracked, as it is contained already in a report.

15.3.2.2.4 ServiceType = GetBRCBValues

Shall not be tracked.

15.3.2.3 Tracking of service for unbuffered reporting – Unbuffered report tracking service (UTS)

The tracking of service for unbuffered reporting (unbuffered report tracking service - **UTS**) shall be as defined in Table 29. The **UTS CDC** shall be used by data objects for service tracking of the access of a **URCB** instance. It shall inherit all attributes of the **CST CDC**.

Table 29 – Unbuffered report tracking service (UTS) definition

UTS Class						
Attribute name	**Attribute type**	**FC**	**TrgOp**	**r/w**	**Value/value range**	**M/O/C**
Shall inherit all the data attributes of the **CST CDC**						
Specific to the UTS						
rptID	VISIBLE STRING129	SR		r	Service parameter ReportIdentifier is mapped to rptID	M
rptEna	BOOLEAN	SR		r	Service parameter ReportEnable is mapped to rptEna	M
resv	BOOLEAN	SR		r	Service parameter Reserve	M
datSet	ObjectReference	SR		r	Service parameter DatSetReference is mapped to datSet	M
confRev	INT32U	SR		r	Service parameter ConfigurationRevision is mapped to confRev	M
optFlds	PACKED_LIST	SR		r	Service parameter OptionalFields is mapped to optFlds	M

UTS Class						
Attribute name	**Attribute type**	**FC**	**TrgOp**	**r/w**	**Value/value range**	**M/O/C**
Shall inherit all the data attributes of the **CST CDC**						
Specific to the UTS						
bufTm	INT32U	SR		r	Service parameter BufferTime is mapped to bufTm	M
sqNum	INT8U	SR		r	Attribute SqNum of URCB is mapped to sqNum	M
trgOps	TriggerConditions	SR		r	Service parameter TriggerOptionsEnabled is mapped to trgOps	M
intgPd	INT32U	SR		r	Service parameter IntegrityPeriod is mapped to intgPd	M
gi	BOOLEAN	SR		r	Service parameter GeneralInterrogation is mapped to gi	M

15.3.2.3.1 ServiceType = SetURCBValues

The attributes **rptID**, **rptEna**, **datSet**, **optFlds**, **bufTm**, **trgOps**, **intPd**, and **resv** mirror the service parameters of the service **SetURCBValues** of the **URCB** referenced by the attribute **objRef**. The attribute **errorCode** returns the error that occurred during the service, or *no-error*, if it was successful. The attribute **originatorID** contains the ID of the client. In case one of the attribute is not written while using the **SetURCBValues** service (**ReportIdentifier** [0..1], …), the value that is returned in the **UTS** reflects the value that would have been returned by a **GetURCBValues** performed on the **URCB** referenced by the attribute **objRef** at the completion of the **SetURCBValues**.

15.3.2.3.2 ServiceType = InternalChange

The attributes **RptEna** (loss of association with the Client), **and Resv** (state of Reservation) of the **URCB** instances shall be tracked.

The value that is returned in the **UTS** reflects the value that would have been returned by a **GetURCBValues** performed on the **URCB** referenced by the attribute **objRef** at completion of the **InternalChange**.

15.3.2.3.3 ServiceType = Report

Shall not be tracked, as it is contained already in a report.

15.3.2.3.4 ServiceType = GetURCBValues

Shall not be tracked.

15.3.2.4 Tracking of service for log control block – Log control block tracking service (LTS)

The tracking of service for log control block access (log control block tracking service - **LTS**) shall be as defined in Table 30. The **LTS CDC** shall be used by data objects for service tracking of the access of a **LCB** instance. It shall inherit all attributes of the **CST CDC**.

Table 30 – Log control block tracking service (LTS) definition

LTS Class						
Attribute name	**Attribute type**	**FC**	**TrgOp**	**r/w**	**Value/value range**	**M/O/C**
Shall inherit all the data attributes of the **CST CDC**						
Specific to the LTS						
logEna	BOOLEAN	SR		r	Service parameter LogEnable is mapped to logEna	M
datSet	ObjectReference	SR		r	Service parameter DataSetReference is mapped to datSet	M
optFlds	PACKED LIST	SR		r	Service parameter OptionalFields is mapped to optFlds	M
bufTm	INT32U	SR		r	Service parameter BufferTime is mapped to bufTm	M
trgOps	TriggerConditions	SR		r	Service parameter TriggerOptions is mapped to trgOps	M
intgPd	INT32U	SR		r	Service parameter IntegrityPeriod is mapped to intgPd	M
logRef	ObjectReference	SR		r	Service parameter LogReference is mapped to logRef	M

15.3.2.4.1 ServiceType = SetLCBValues

The attributes **logEna**, **datSet**, **optFlds**, **bufTm**, **trgOps**, **intPd**, **logRef** mirror the service parameters of the service **SetLCBValues** of the **LCB** referenced by the attribute **objRef**. The attribute **errorCode** returns the error that occurred during the service. The attribute **originatorID** contains the ID of the client. In case one of the attribute is not written while using the **SetLCBValues** service (**LogEnable** [0..1], …), the value that is returned in the **LTS** reflects the value that is set in the **LCB** referenced by **objRef**.

15.3.2.4.2 ServiceType = GetLCBValues

Shall not be tracked.

15.3.2.5 Tracking of service for log – Log tracking service (OTS)

The tracking of service for log access (log tracking service - **OTS**) shall be as defined in Table 31. The **OTS CDC** shall be used by data objects for service tracking of the access of a **LOG** instance. It shall inherit all attributes of the **CST CDC**.

Table 31 – Log tracking service (OTS) definition

OTS Class						
Attribute name	**Attribute type**	**FC**	**TrgOp**	**r/w**	**Value/value range**	**M/O/C**
Shall inherit all the data attributes of the **CST CDC**						
Specific to the LTS						
oldEntrTm	TimeStamp	SR		r	Service parameter OldestEntryTime is mapped to oldEntrTm	M
newEntrTm	TimeStamp	SR		r	Service parameter NewestEntryTime is mapped to newEntrTm	M
oldEntr	INT32U	SR		r	Service parameter OldestEntry is mapped to oldEntr	M
newEntr	INT32U	SR		r	Service parameter NewestEntry is mapped to newEntr	M
rangeStrTm	TimeStamp	SR		r	Service parameter RangeStartTime is mapped to rangeStrTm	M
rangeStpTm	TimeStamp	SR		r	Service parameter RangeStopTime is mapped to rangeStpTm	M
entry	EntryID	SR		r	Service parameter Entry is mapped to entry	M

15.3.2.5.1 ServiceType = QueryLogByTime

Shall not be tracked.

15.3.2.5.2 ServiceType = QueryLogAfter

Shall not be tracked.

15.3.2.5.3 ServiceType = GetLogStatusValues

Shall not be tracked.

15.3.2.6 Tracking of service for GOOSE control block – GOOSE control block tracking service (GTS)

The tracking of service for **GOOSE** control block access (**GOOSE** control block tracking service - **GTS**) shall be as defined in Table 32. The **GTS CDC** shall be used by data objects for service tracking of the access of a **GOOSE** control block instance. It shall inherit all attributes of the **CST CDC**.

Table 32 – GOOSE Control block tracking service (GTS) definition

GTS Class						
Attribute name	**Attribute type**	**FC**	**TrgOp**	**r/w**	**Value/value range**	**M/O/C**
Shall inherit all the data attributes of the **CST CDC**						
Specific to the GTS						
goEna	BOOLEAN	SR		r	The service parameter GoEnable is mapped to goEna	M
goID	VISIBLE STRING 129	SR		r	The service parameter GOOSEID is mapped to goID	M
datSet	ObjectReference	SR		r	The service parameter DataSetReference is mapped to datSet	M
confRev	INT32U	SR		r	The service parameter ConfigurationRevision is mapped to confRev	M
ndsCom	BOOLEAN	SR		r	The service parameter NeedsCommissioning is mapped to ndsCom	M
dstAddress	PHYCOMADDR	SR		r	The service parameter DestinationAddress is mapped to dstAddress	M

15.3.2.6.1 ServiceType = SendGOOSEMessage

Shall not be tracked.

15.3.2.6.2 ServiceType = GetGoReference

Shall not be tracked.

15.3.2.6.3 ServiceType = GetGOOSEElementNumber

Shall not be tracked.

15.3.2.6.4 ServiceType = GetGoCBValues

Shall not be tracked.

15.3.2.6.5 ServiceType = SetGoCBValues

The attributes **goEna**, **goID**, and **datSet** mirror the service parameters of the service **SetGoCBValues** of the **GoCB** referenced by the attribute **objRef**. The attribute **errorCode** returns the error that occurred during the service, or the value *no-error* if it was successful. The attribute **originatorID** contains the ID of the client. In case one of the attributes is not written while using the **SetGoCBValues** service (**GoEnable** [0..1], …), the value that is returned in the **GTS** is the value that is set in the **GoCB** referenced by **objRef**.

15.3.2.7 Tracking of service for MSVCB control block – MSVCB tracking service (MTS)

The tracking of service for **MSVCB** control block (**MSVCB** tracking service - **MTS**) shall be as defined in Table 33. The **MTS CDC** shall be used by data objects for service tracking of the access of a **MSVCB** instance. It shall inherit all attributes of the **CST CDC**.

Table 33 – MSVCB tracking service (MTS) definition

MTS Class						
Attribute name	**Attribute type**	**FC**	**TrgOp**	**r/w**	**Value/value range**	**M/O/C**
Shall inherit all the data attributes of the **CST CDC**						
Specific to the MTS						
svEna	BOOLEAN	SR		r	The service parameter SvEnable is mapped to svEna	M
msvID	VISIBLE STRING 129	SR		r	The service parameter MulticastSampleValueID is mapped to msvID	M
datSet	ObjectReference	SR		r	The service parameter DataSetReference is mapped to datSet	M
confRev	INT32U	SR		r	The service parameter ConfigurationRevision is mapped to confRev	M
smpMod	ENUMERATED	SR		r	The service parameter SampleMode is mapped to smpMod	M
smpRate	INT16U	SR		r	The service parameter SampleRate is mapped to smpRate	M
optFlds	PACKED LIST	SR		r	The service parameter OptionalFields is mapped to optFlds	M
dstAddress	PHYCOMADDR	SR		r	The service parameter DestinationAddress is mapped to dstAddress	M

15.3.2.7.1 ServiceType = SendMSVMessage

Shall not be tracked.

15.3.2.7.2 ServiceType = GetMSVCBValues

Shall not be tracked.

15.3.2.7.3 ServiceType = SetMSVCBValues

The attributes **svEna**, **datSet**, **msvID**, **smpMod**, **smpRate**, and **optFlds** mirror the service parameters of the service **SetMSVCBValues** of the **MSVCB** referenced by the attribute **objRef**. The attribute **errorCode** returns the error that occurred during the service, or the value *no-error* if it was successful. The attribute **originatorID** contains the ID of the client. In case one of the attributes is not written while using the **SetMSVCBValues** service (**SvEnable** [0..1], …), the value that is returned in the **MTS** reflects the value that is set in the **MSVCB** referenced by **objRef**.

15.3.2.8 Tracking of service for USVCB control block – USVCB tracking service (NTS)

The tracking of service for **USVCB** control block (**USVCB** control block tracking service - **NTS**) shall be as defined in Table 34. The **NTS CDC** shall be used by data objects for service tracking of the access of a **USVCB** instance. It shall inherit all attributes of the **CST CDC**.

Table 34 – USVCB tracking service (NTS) definition

NTS Class						
Attribute name	**Attribute type**	**FC**	**TrgOp**	**r/w**	**Value/value range**	**M/O/C**
Shall inherit all the data attributes of the **CST CDC**						
Specific to the NTS						
svEna	BOOLEAN	SR		r	The service parameter SvEnable is mapped to svEna	M
usvID	VISIBLE STRING 129	SR		r	The service parameter UnicastSampleValueID is mapped to usvID	M
datSet	ObjectReference	SR		r	The service parameter DataSetReference is mapped to datSet	M
confRev	INT32U	SR		r	The service parameter ConfigurationRevision is mapped to confRev	M
smpMod	ENUMERATED	SR		r	The service parameter SampleMode is mapped to smpMod	M
smpRate	INT16U	SR		r	The service parameter SampleRate is mapped to smpRate	M
optFlds	PACKED LIST	SR		r	The service parameter OptionalFields is mapped to optFlds	M
dstAddress	PHYCOMADDR	SR		r	The service parameter DestinationAddress is mapped to dstAddress	M

15.3.2.8.1 ServiceType = SendUSVMessage

Shall not be tracked.

15.3.2.8.2 ServiceType = GetUSVCBValues

Shall not be tracked.

15.3.2.8.3 ServiceType = SetUSVCBValues

The attributes **svEna**, **datSet**, **usvID**, **smpMod**, **smpRate**, and **optFlds** mirror the service parameters of the service **SetUSVCBValues** of the **USVCB** referenced by the attribute **objRef**. The attribute **errorCode** returns the error that occurred during the service, or the value *no-error* if it was successful. The attribute **originatorID** contains the ID of the client. In case one of the attributes is not written while using the **SetUSVCBValues** service (**SvEnable** [0..1], …), the value that is returned in the **UTS** reflects the value that is set in the **USVCB** referenced by **objRef**.

15.3.2.9 Tracking of service for SGCB – SGCB tracking service (STS)

The tracking of service for **SGCB** control block (**SGCB** tracking service - **STS**) shall be as defined in Table 35. The **STS CDC** shall be used by data objects for service tracking of the access of a **SGCB** instance. It shall inherit all attributes of the **CST CDC**.

Table 35 – SGCB tracking service (STS) definition

STS Class						
Attribute name	**Attribute type**	**FC**	**TrgOp**	**r/w**	**Value/value range**	**M/O/C**
Shall inherit all the data attributes of the **CST CDC**						
Specific to the STS						
numOfSG	INT8U	SR		r	The service parameter NumberOfSettingGroup is mapped to numOfSG	M
actSG	INT8U	SR		r	The service parameter ActiveSettingGroup is mapped to actSG	M
editSG	INT8U	SR		r	The service parameters SettingGroupNumber resp. EditSettingGroup are mapped to editSG	M
cnfEdit	BOOLEAN	SR		r		M
lActTm	TimeStamp	SR		r	The service parameter LastActivateTime is mapped to lActTm	M
resvTms	INT16U	SR		r		O

15.3.2.9.1 ServiceType = SelectActiveSG

The attribute **actSG** mirrors the service parameter of the service **SelectActiveSG** of the **SGCB** referenced by **objRef**. The attribute **errorCode** returns the error that occurred during the service, or the value *no-error* if it was successful. The attribute **originatorID** contains the ID of the client. The value that is returned in the **STS** for the attributes (**numOfSG**, **editSG**, ...) reflects the value that is set in the **SGCB** referenced by **objRef**.

15.3.2.9.2 ServiceType = SelectEditSG

The attribute **editSG** mirrors the service parameters of the service **SelectEditSG** of the **SGCB** referenced by **objRef**. The attribute **errorCode** returns the error that occurred during the service, or the value *no-error* if it was successful. **originatorID** contains the ID of the client. The value that is returned in the **STS** for the attributes (**numOfSG**, **actSG**, ...) reflects the value that is set in the **SGCB** referenced by **objRef**.

15.3.2.9.3 ServiceType = SetEditSGValue

The service SetEditSGValue shall be tracked but not in the scope of the Tracking Service for SGCB, but in the scope of Tracking Services (see Clause 14).

15.3.2.9.4 ServiceType = ConfirmEditSGValues

The attribute **cnfEdit** mirrors the service parameters of the service **ConfirmEditSGValues** of the **SGCB** referenced by **objRef**. The attribute **errorCode** returns the error that occurred during the service, or the value *no-error* if it was successful. The attribute **originatorID** contains the ID of the client. The value that is returned in the **STS** for the attributes (**numOfSG**, **editSG**, ...) reflects the value that is set in the **SGCB** referenced by **objRef**.

15.3.2.9.5 ServiceType = GetEditSGValue

Shall not be tracked.

15.3.2.9.6 ServiceType = GetSGCBValues

Shall not be tracked.

16 SETTING-GROUP-CONTROL-BLOCK class model

16.1 General

An instance of a data object class 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 objects. Values that belong together build the setting group (**SettingGroup**).

Setting groups span a logical device, and therefore the SGCB, if it exists at all, shall be located in LLN0. A logical node zero may have one **setting-group-control-block**. The setting data objects are defined in IEC 61850-7-4.

The **SGCB** model provides services to handle different values for one or more data object. The **SettingGroup** whose values are currently used by the data object of a logical node shall be in the state “active”. The **SettingGroup** that can be edited shall be in the state “edit”.

The **SGCB** model is depicted in the example in Figure 22. The **logical-node “PVOC”** (voltage controlled/dependent time overcurrent according to IEC 61850-7-4) comprises eight data objects for settings (**logical-node PDIF** has one data object for settings) – (**MinOpTmms**, ..., **RstMode**). The **SGCB** “**SettingGroup** Control” provides three **SettingGroups** (#1, #2, and #3) each with independent values for the nine **data**. Each **SettingGroup** contains nine values (one for each of: **MinOpTmms**, ..., **RstMode**). The members of the active **SettingGroup** are referenced by the **ObjectReferences** of the data object with functional constraint **SG**. The members of the **SettingGroup** in the “edit buffer” are referenced by the **ObjectReferences** of the data object with functional constraint **SE**.

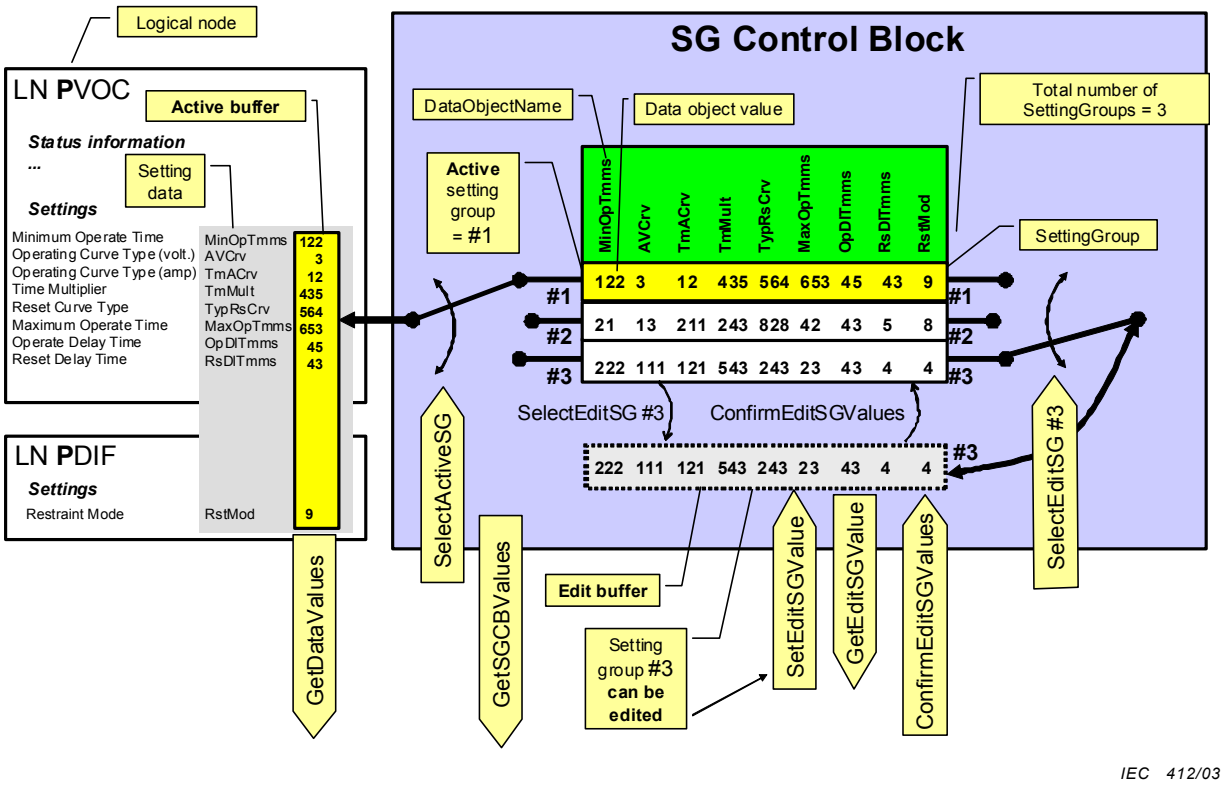


Figure 22 – Basic model of the settings model

The values of the data object of the logical node **PVOC** are derived from the values of one of the **SettingGroups**. This is accomplished by the multiplexer on the left. The service **SelectActiveSG** determines which values (of **SettingGroup** #1, #2, or #3) shall be copied to the "active buffer" and be used by **PVOC**. In the example, the **SettingGroup** #1 has been set to be in the active state.

A **SettingGroup** contains values for data objects that are contained in several logical nodes. The **SettingGroups** in the example provide values for data objects in two logical nodes (**PDIF** and **PVOC**).

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

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

The active value of a data object contained in the **SettingGroup** can be accessed directly with **GetDataValues**.

16.2 SGCB class definition

16.2.1 SGCB class syntax

The **SGCB** shall have the structure defined in Table 36.

Clients should use the existence of a **SGCB** to determine if the **logical-device** contains SettingGroups.

Table 36 – SGCB class definition

SGCB class				
Attribute name	**Attribute type**	**r/w**	**Value/value range/explanation**	**M/O/C**
SGCBName	ObjectName	--	Instance name of an instance of SGCB	M
SGCBRef	ObjectReference	--	Path-name of an instance of SGCB	M
NumOfSG	INT8U	r	n = NumOfSG	M
ActSG	INT8U	r/w	Allowable range: 1 ... n	M
EditSG	INT8U	r/w	Allowable range: 0 ... n	M
CnfEdit	BOOLEAN	w		M
LActTm	TimeStamp	r		M
ResvTms	INT16U	r		O
Services SelectActiveSG SelectEditSG SetEditSGValue ConfirmEditSGValues GetEditSGValue GetSGCBValues				

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

16.2.2 SGCB class attributes

16.2.2.1 SGCBName – setting group control name

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

16.2.2.2 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**.

16.2.2.3 NumOfSG – number of setting groups

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

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

16.2.2.4 ActSG – active setting group

The attribute **ActSG** shall identify the values of the **SettingGroup** that are in the active buffer. The attribute **ActSG** shall define the **SettingGroup** whose values shall be used by the

respective **LOGICAL-NODE** performing its function. The values of the **DataAttribute** of the active **SettingGroup** can be retrieved by the service **GetDataValues**.

16.2.2.5 EditSG – edit setting group

The attribute **EditSG** shall identify the values of the **SettingGroup** that shall be copied into the edit buffer. The values of the edit buffer can be set and retrieved by the services **SetEditSGValue** and **GetEditSGValue**. The current values in the **SettingGroup** shall be unchanged until the client has confirmed to overwrite the values with those values stored in the edit buffer (**ConfirmEditSGValues**). The values stored in the buffer shall be used to overwrite the current values in the **SettingGroup**.

If the value of **EditSG** is (= 0) then the use of services **SetEditSGValue** (with **FC**=**SE**) and **GetEditSGValue** shall cause a **Response–**.

16.2.2.6 CnfEdit – confirm editing

The attribute **CnfEdit** shall be used to confirm the editing process.

16.2.2.7 LActTm – last activation time

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

16.2.2.8 ResvTms – reservation of SettingGroup in seconds

The configuration attribute ResvTms defines the time interval in seconds a reservation of a SGCB is granted to a client. A reservation of a SGCB starts after a successful SettingGroup edition has been successfully processed with the service **SelectEditSG**.

16.3 SGCB class services

16.3.1 Overview

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

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

16.3.2 SelectActiveSG

16.3.2.1 SelectActiveSG parameter table

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

Parameter name
Request
SGCBReference

Parameter name
SettingGroupNumber
Response+
Response–
ServiceError

16.3.2.2 Request

16.3.2.2.1 SGCBReference

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

16.3.2.2.2 SettingGroupNumber

The parameter **SettingGroupNumber** shall define the number **ActSG** of the **SettingGroup** (between 1 and **NumOfSG**) that shall be used to determine the new values of data object of the respective logical nodes.

The values of all instances of the setting data objects 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 objects of the setting group referenced in the service request.

16.3.2.3 Response+

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

16.3.2.4 Response–

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

16.3.3 SelectEditSG

16.3.3.1 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 **SettingGroup** to the edit buffer.

Parameter name
Request
SGCBReference
SettingGroupNumber
Response+
Response–
ServiceError

The client that issues a SelectEditSG with a SettingGroupNumber (between 1 and **NumOfSG**) gets the SGCB reserved, in case the SGCB has not been reserved previously by another client.

The reservation of a SGCB remains granted until:

- the editing is completed and the client that has the SGCB reservation granted issues a ConfirmEditSGValues,
- the association with the client that has the SGCB reservation granted is lost,
- a timeout occured (no ConfirmEditSGValues service has been received within the specific timeout attribute ResvTms after SelectEditSG service has been received from the client that has the SGCB reservation granted),
- the edition is cancelled by the client that has the SGCB reservation granted with means of the service SelectEditSG where SettingGroupNumber is 0 (zero).

16.3.3.2 Request

16.3.3.2.1 SGCBReference

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

The **ObjectReference SGCBReference** shall be:

LDName/LLN0.SGCB

16.3.3.2.2 SettingGroupNumber

The parameter **SettingGroupNumber** shall define the number **EditSG** of the **SettingGroup** (between 1 and **NumOfSG**) that shall be used to set values (**SetEditSGValue**), confirm value (**ConfirmEditSGValues**), and retrieve value (**GetEditSGValue**) of the specified **SettingGroup**.

16.3.3.3 Response+

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

16.3.3.4 Response–

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

16.3.4 SetEditSGValue

16.3.4.1 SetEditSGValue parameter table

A client shall use the **SetEditSGValue** service to set the value of the data object of the **SettingGroup** 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 value shall become effective only after the client has confirmed the value by issuing the service **ConfirmEditSGValues**.

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

16.3.4.2 Request

16.3.4.2.1 Reference

The parameter **Reference** shall define the **functional constrained data** (**FCD**) or **functional constrained data attributes** (**FCDA**) of the data object whose **DataAttribute** value is to be written. The **Reference** shall be **FCD** or **FCDA**.

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

16.3.4.2.2 DataAttributeValue [1..n]

The parameter **DataAttributeValue** shall contain

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

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

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

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

16.3.4.3 Response+

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

16.3.4.4 Response–

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

16.3.5 ConfirmEditSGValues

16.3.5.1 ConfirmEditSGValues parameter table

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

Parameter name
Request
SGCBReference
Response+
Result
Response–
ServiceError

In case the edition were made on the “active” **SettingGroup**, then the service **ConfirmEditSGValues** re-activates automatically the active **SettingGroup** with the updated setting values.

16.3.5.2 Request

SGCBReference

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

16.3.5.3 Response+

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

16.3.5.4 Response–

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

16.3.6 GetEditSGValue

16.3.6.1 GetEditSGValue parameter table

A client shall use the **GetEditSGValue** service to retrieve the value of a data object of **Setting Groups** made visible and thus accessible to the requesting client by the referenced **LLN0**.

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

16.3.6.2 Request

Reference

The parameter **Reference** shall define the **functional constrained data** (**FCD**) or **functional constrained data attributes** (**FCDA**) of the data object whose **DataAttribute** value 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 value of the **SettingGroup** in the edit buffer; and

– **SG** to retrieve the value of the active **SettingGroup**.

16.3.6.3 Response+

DataAttributeValue [1..n]

The parameter **DataAttributeValue** shall contain

– the value of all **DataAttributes** of a data object 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 **DataAttributeValue** is defined in an SCSM.

16.3.6.4 Response–

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

16.3.7 GetSGCBValues

16.3.7.1 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**.

Parameter name
Request
SGCBReference
Response+
NumberOfSettingGroup
ActiveSettingGroup
EditSettingGroup
LastActivateTime
ResvTms [0..1]
Response–
ServiceError

16.3.7.2 Request

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

16.3.7.3 Response+

16.3.7.3.1 NumberOfSettingGroup – number of setting group controls

The parameter **NumberOfSettingGroup** shall define the total number of the **SettingGroup** of the attribute **NumOfSG** of the referenced **SGCB**.

16.3.7.3.2 ActiveSettingGroup – active setting group

The parameter **ActiveSettingGroup** shall define the number of the **SettingGroups** of the attribute **ActiveSG** from which the current active **SettingGroup** values shall be derived.

16.3.7.3.3 EditSettingGroup – edit setting group

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

16.3.7.3.4 LastActivateTime – last time of activation of a setting group

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

16.3.7.4 Response–

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

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

17.1 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 (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 a data object 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 objects at any time: GetDataValues and GetDataSetValues.

NOTE 1 Clause 18 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 values once (**GSE** transmits and retransmits data values with heartbeat). IEC 61850-7-1 provides a comparison of the models.

NOTE 2 Clause 19 specifies special services for communication of measured values of, for example, voltage transformer (VT) and current transformer (CT) under crucial time constraints.

The principal building blocks and services for reporting and logging are depicted in Figure 23.

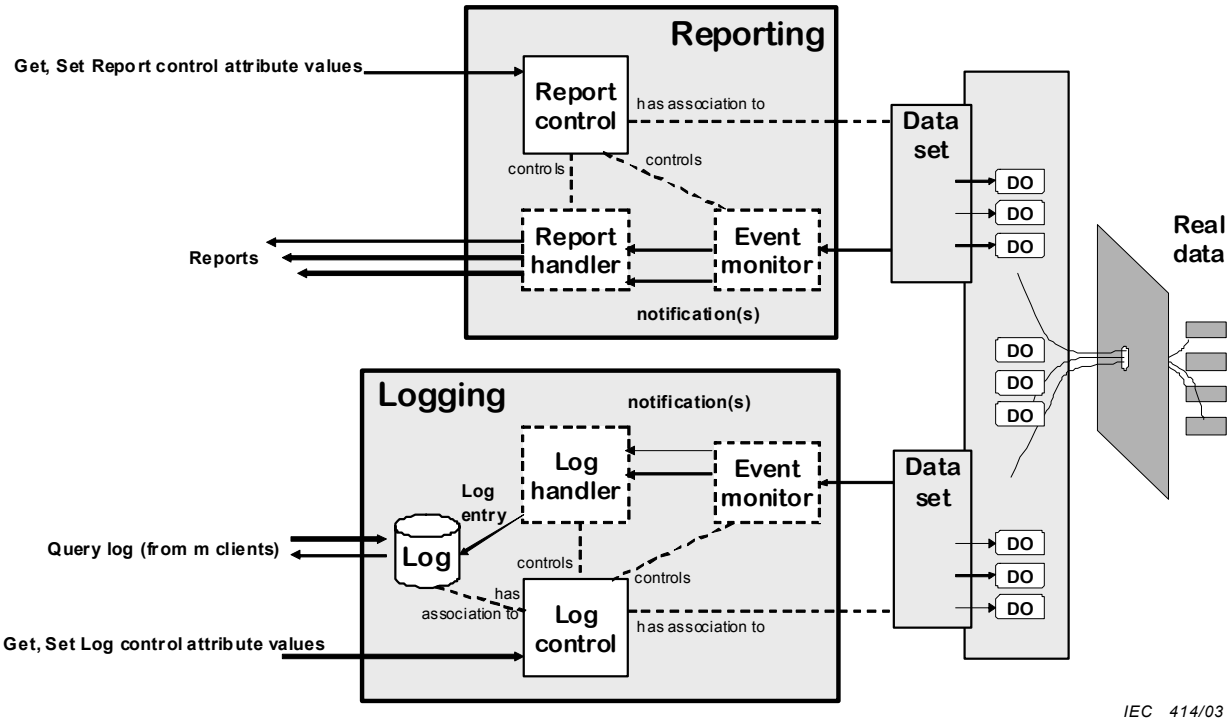


Figure 23 – Basic building blocks for reporting and logging

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 object) represent the values of data objects. The 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 generate a notification to the appropriate handlers (for example log or report handler). This notification includes the data object values and reasons for data inclusion.

NOTE 4 The contents of the event notification are determined by a combination of I/O scan and event monitoring. These are conceptually two asynchronous processes. Therefore, the number of values included within a single notification is a local issue.

The number of values, within a notification, is a local implementation issue. The report handler assigns EntryID(s) and TimeOfEntry(s) to the values contained within a set of notifications. The number of notifications combined into a single EntryID is determined by the RCB control parameters (for example BufTm). The value of the EntryID is a local issue but it shall be a unique arbitrary OCTET STRING whose value is unique within the scope of entries for a specific RCB. The value of the TimeOfEntry shall be the timestamp representing the time at which the report handler received the first notification that is used to form an EntryID. 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 client may initiate a general interrogation at any time to receive all values of an application specific set of data object. Using this mechanism, clients can synchronize their databases with the current status of one or more logical nodes.

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

17.2 REPORT-CONTROL-BLOCK class model

17.2.1 Basic concepts

The **report-control-block** shall control the procedures that are required for reporting values of data objects from one or more logical node to one client. Instances of report control blocks shall be configured in the server at configuration time.

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

There are two classes of report control blocks 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 object 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.

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

Report control block instances are named. These names must be unique within the scope of a logical node. The number of instances is a local implementation issue and must be appropriately reflected in the configuration provided (for example by SCL). Once a report control block is reserved, by a specific client, no other client shall have access rights to set the control block attributes.

Buffered report control blocks are usually configured to be used by a specific 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.

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

17.2.2.1 BRCB class Syntax

The **BRCB** class shall have the structure defined in Table 37.

Table 37 – BRCB class definition

BRCB class			
Attribute name	**Attribute type**	**r/w**	**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 STRING129	r/w	c1, c2
RptEna	BOOLEAN	r/w	
DatSet	ObjectReference	r/w	c1, c2
ConfRev	INT32U	r	
OptFlds	PACKED LIST	r/w	c2
sequence-number	BOOLEAN		
report-time-stamp	BOOLEAN		
reason-for-inclusion	BOOLEAN		
data-set-name	BOOLEAN		
data-reference	BOOLEAN		
buffer-overflow	BOOLEAN		
entryID	BOOLEAN		
conf-revision	BOOLEAN		
BufTm	INT32U	r/w	c1, c2
SqNum	INT16U	r	
TrgOps	TriggerConditions	r/w	c1, c2
IntgPd	INT32U	r/w	c1, 0.. MAX; 0 implies no integrity report
GI	BOOLEAN	r/w	
PurgeBuf	BOOLEAN	r/w	
EntryID	EntryID	r/w	c2
TimeOfEntry	EntryTime	r	
ResvTms	INT16	r/w	c3
Owner	OCTET STRING64	r	c4
Services Report GetBRCBValues SetBRCBValues			
Notes and conditions **c1:** These attributes may only be set when RptEna = FALSE. If a SetBRCBValues service is executed against these attributes and causes a change of value, the implementation shall execute a purge of the buffered events as if PurgeBuf had been set to TRUE. **c2:** These attributes may only be set when RptEna = FALSE. A SetBRCBValues of these parameters, when RptEna=TRUE, shall fail. **c3:** This attribute is optional. If the attribute is not present, then the reservation of the control block shall occur based upon pre-configuration or by the client that performs the first SetBRCBValues with RptEna=TRUE. **c4**: This attribute is optional. NOTE An attribute that is marked “r” indicates that the BRCB attribute may be obtained (for example read) through the use of the GetBRCBValues service. An attribute that is marked “w” indicates that the BRCB attribute may be set (for example written) through the use of the SetBRCBValues service.			

The use of a particular instance of a **BRCB** (for example which remote clients can actually use the SetBRCBValues service on a particular **BRCB**) is typically pre-configured locally. The means of performing this configuration is a local issue. However, it is assumed that one or more cooperating clients (for example primary and secondary SCADA systems) may be allowed to have access to a single **BRCB**. For each set of cooperating clients, for which the server is to provide buffered reporting service, different **BRCB** instances shall be provided.

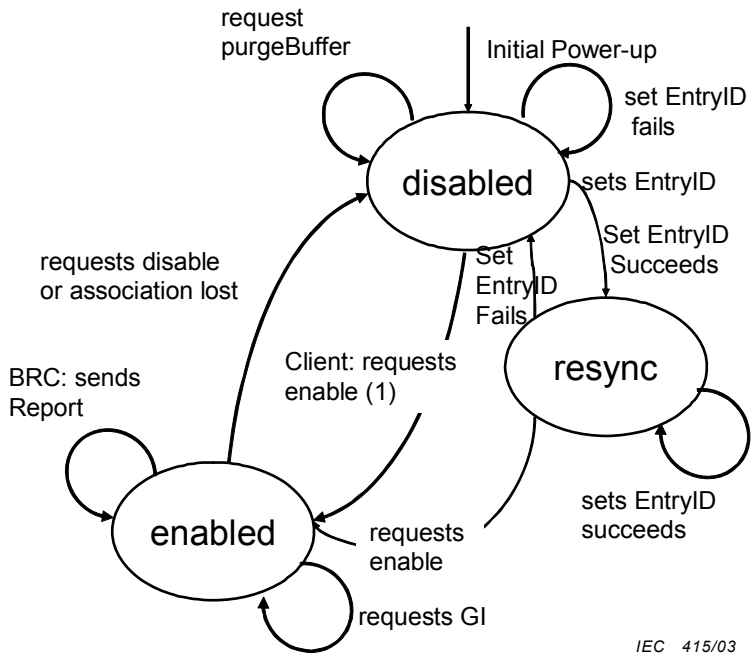


Figure 24 – BRCB state machine

The behavior of a BRCB instance, when used by clients, is shown in the BRCB state machine in Figure 24. The behaviour of the different conceptual states is as follows:

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

Upon power-up, buffering shall be started if the BRCB's DatSet attribute contains a reference to an existing DataSet. If the DatSet attribute contains a NULL value or a reference to an invalid DataSet, then no transition shall be allowed from the disabled state.

When a client uses SetBRCBValues to set the EntryID attribute value and the set EntryID does not exist within the queue of entries, a ServiceError of parameter-value-inappropriate shall be returned.

If the EntryID value, in the SetBRCBValues, is zero (0), this value is reserved to be used by the client to resynchronize to the first entry in the queue.

If the set value of the EntryID exists within the queue of entries, a SetBRCBValues response+ is returned and the BRCB state shall transition from **disabled** to **resynchronize**.

When a client uses SetBRCBValues to set RptEna=TRUE, the state shall transition to the **enabled** state.

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

When a client uses SetBRCBValues to set an EntryID attribute value and the EntryID value exists within the queue of entries, a SetBRCBValues response+ is returned and the BRCB state shall remain **resync**.

If the value of the set EntryID does not exist within the queue of entries, a ServiceError of parameter-value-inappropriate shall be returned and the BRCB state shall transition to **disabled**.

If the EntryID value, in the SetBRCBValues, is zero (0), this value is reserved to be used by the client to resynchronize to the first entry in the queue, the state shall transition from **resync** to **disabled**.

When an association is lost, the state shall transition to **disabled**.

When a client uses SetBRCBValues to set RptEna=TRUE the state shall transition to **enabled**.

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

When an association is lost, the state shall transition to **disabled**.

When a client uses SetBRCBValues to set RptEna=FALSE, the state shall transition to **disabled** and reporting shall cease.

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.

Logically, the report handler has a queue of entries that are used to sequence the formatting and “transmission” (for example queuing to the N-1 layer) reports. The report handler, logically, has a pointer to the next entry to be queued for formatting and transmission. See Figure 25.

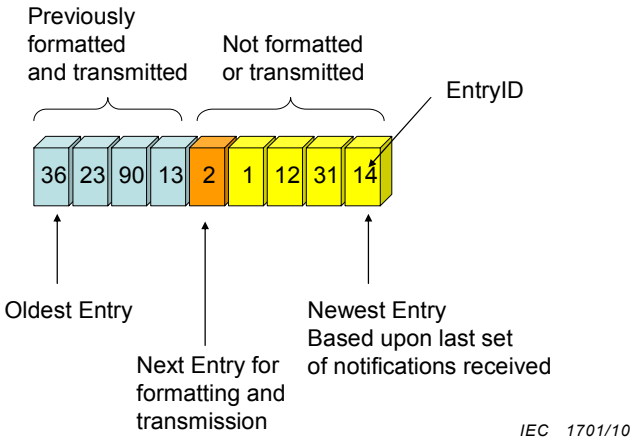


Figure 25 – General queue of entries for report handler

17.2.2.2 BRCBName – buffered report control name

The attribute **BRCBName** shall be the name of the **BRCB** that unambiguously identifies the **BRCB** within a **logical-node**.

17.2.2.3 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

17.2.2.4 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 reference 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.

17.2.2.5 RptEna – report enable

The attribute **RptEna** shall be used by a client to enable and disable reporting, and it indicates this state in the **BRCB**. The state machine for the attribute **RptEna** shall be as depicted in Figure 24.

The **BRCB** shall monitor the values of the **DataAttributes** 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 value of RptEna, in the disabled and resync states of Figure 24, shall be FALSE.

The client shall configure the **BRCB** and shall then set this attribute to **enabled** (see (1) in Figure 24).

While in the state **enabled**, no changes of attribute values of the **BRCB** shall be allowed except triggering a general-integration.

17.2.2.6 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** attribute value 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 successfull SetBRCBValues request of the attribute **DatSet** shall have the same effect as setting **purgeBuf** to TRUE. If the DatSet reference is a valid DataSet or NULL, the SetBRCBValues service shall indicate success. If the DataSet reference is invalid, then the SetBRCBValues service shall indicate failure of badValue and the value of the attribute shall remain unchanged.

It is up to local DataSet configuration and client DataSet creation to ensure that DataSet members have TrgOps defined. DataSet members, that have no TrgOps defined, shall only be reported due to Integrity and GI reports.

17.2.2.7 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**;

– the reordering of members of the **data-set**; and

– successful SetBRCBValues of the DatSet attribute where the DatSet attribute value changes.

The counter shall be incremented when the configuration changes. At configuration time, the configuration tool will be responsible for incrementing/maintaining the ConfRev value. When configuration changes occur due to SetBRCBValues, the IED shall be responsible for incrementing the value of ConfRev.

The initial value for **ConfRev** is outside the scope of this part of IEC 61850. The value of 0 shall be reserved. The value of ConfRev, upon a restart of the IED, is a local issue.

There are two possible options for the initializing of the value of ConfRev and the associated configuration: the value is restored to the original value of the base local configuration or the value is retained from the configuration prior to restart. The PICS shall clarify which is implemented.

17.2.2.8 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 17.2.3.2.2.1) that shall be included in the report:

– sequence-number (if TRUE **SqNum** 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);

– conf-revision (if TRUE **ConfRev** 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.

17.2.2.9 BufTm – buffer time

The attribute **BufTm** (see Figure 26) 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.

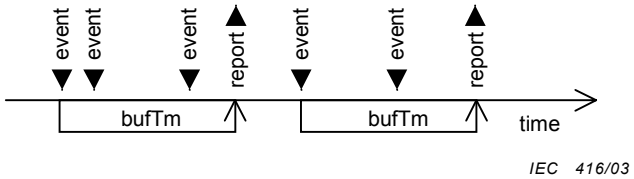


Figure 26 – Buffer time

Upon receipt of the first set of internal notification of events of the referenced **data-set**, the **BRCB** shall start a timer of the duration buffer time. The number of events within the internal notification is a local issue. When the timer expires, the **BRCB** shall combine all internal notifications that have been received during the time interval into a single report. The next internal notification 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**. In this case, each internal notification 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 a specific data object for changes of the value 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 (FC=ST) 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 (FC=MX) 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 (FC=MX) substitute the current value in the pending report with the new one.

NOTE 2 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 **SetBRCBValues** service.

A SetBRCBValues of the attribute **BufTm** shall have the same effect as setting **purgeBuf** to TRUE if the attribute value changes.

17.2.2.10 SqNum – sequence number

The attribute **SqNum** 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 it for transmission.

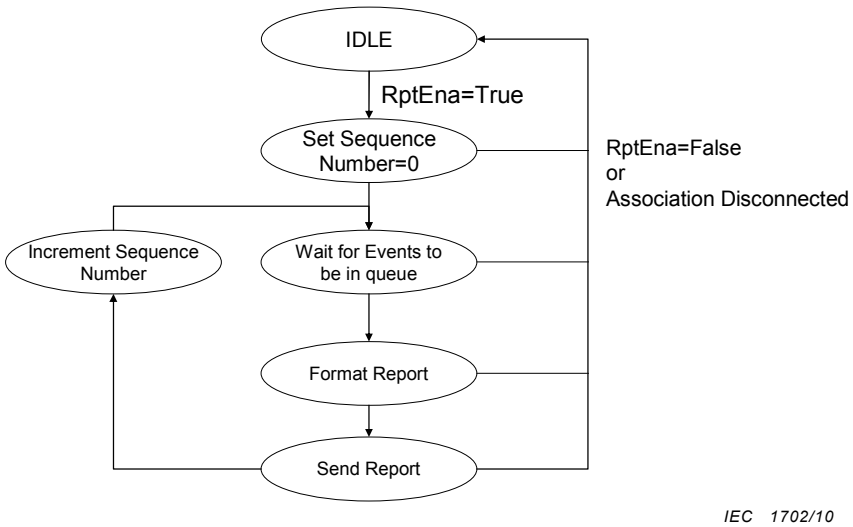


Figure 27 – State Machine for Sequence Number Generation

Figure 27 depicts the logical state machine used in the generation of SqNum. The transition from the IDLE state is caused by the transition of RptEna changing from a value of FALSE to a value of TRUE. Upon that transition, the value of SqNum shall be set to zero (0). As Reports are sent, the value of SqNum shall be incremented. When the maximum value of SqNum is obtained, the next value shall be a value of zero (0).

Subsequent writes of RptEna=TRUE, that do not cause the transition from FALSE to TRUE, shall not cause the value of SqNum to be set to zero (0).

17.2.2.11 TrgOps – trigger options

The attribute **TrgOps** 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 (**TrgOps**) of the **DataAttribute** of the common data classes as defined in this part and for example in IEC 61850-7-3. The trigger options **integrity** and **general-interrogation** shall be the trigger conditions defined by the attributes **IntgPd** and **GI** of the **BRCB** respectively.

Details related to the generation of a report based on the different trigger options shall be as specified in 17.2.3.2.3.

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

A SetBRCBValues request of the attribute **TrgOps** shall have the same effect as setting **purgeBuf** to TRUE, if the attribute value changes.

17.2.2.12 IntgPd – integrity period

If **TrgOps** includes a setting indicating **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 **SetBRCBValues** service.

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

A SetBRCBValues of the attribute **IntgPd** shall have the same effect as setting **purgeBuf** to TRUE if the attribute value changes.

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

17.2.2.13 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**.

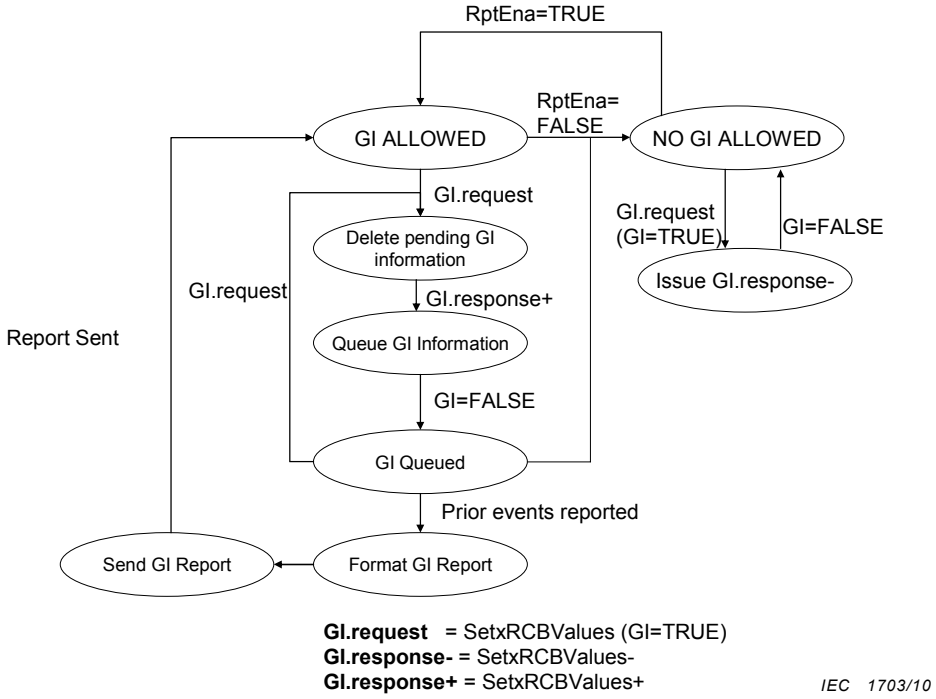


Figure 28 – Logical state machine for general interrogation

Figure 28 depicts the logical state machine for general interrogation.

The initial state is “NO GI ALLOWED”. Within this state, any GI.request (for example GI set TRUE) shall cause a GI.response– to be sent. Additionally, the implementation shall set the value of GI to FALSE.

A transition of RptEna from FALSE to TRUE causes a state transition from “NO GI ALLOWED” to “GI ALLOWED”. Within this state, a client may issue a GI.request (for example GI transitioning from FALSE to TRUE). Upon receipt of the GI.request, the implementation for a BRCB shall delete/remove any previously queued general-interrogation information, queue the requested general-interrogation information, set GI=FALSE, and issue a GI.response+. The order of execution of these steps is a local issue.

NOTE A URCB model does not model a buffer. Therefore, there is no need to delete pending GIs.

Normal reporting will eventually cause previously queued events to be reported and the queued general-interrogation information to be formatted and reported.

If a GI request (for example GI transitioning from FALSE to TRUE) is issued while in the GI Queued state, the general-interrogation process shall be restarted.

Once a GI is formatted for reporting, it is a local issue regarding the impact of a new GI.request. However, once the GI is being sent, the entire set of general-interrogation information shall be sent regardless if a new GI.request has been received. In this case, once the sending of the general-interrogation information is complete, the newly requested general-interrogation process shall start (for example an immediate transition through the GI

ALLOWED state). While GI=TRUE, subsequent GI requests shall not result in additional general-interrogation processing.

If a client attempts to set GI=FALSE, a positive response shall be issued. However, GI pending processing will not be impacted.

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 **SetBRCBValues** service containing not-supported.

If a BRCB contains TrgOps.GI = FALSE then an attempt to set the GI attribute to TRUE shall cause a SetBRCBValues positive response. No GI report shall be generated.

17.2.2.14 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 including those 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**.

17.2.2.15 EntryID – entry identifier

The reported entries are dependant upon the transitions of state of the BRCB.

– A transition from disabled to enabled shall start reporting with the first available entry (i.e. oldest) in the queue of entries. Reporting of the next sequential entries shall occur.
– A transition from resync to enabled shall start reporting with the next available entry (i.e. in time sequence), in the queue of entries, after the entry associated with the EntryID value set by the client. Reporting of the next sequential entries shall occur.

The value of EntryID, returned in a GetBRCBValues response shall be defined as follows.

– When the BRCB state is RptEna=FALSE: a GetBRCBValues shall return the EntryID value that represents the last (i.e., newest) entry that has been entered into the buffer.
– When the BRCB RptEna=TRUE: The value of EntryID, returned in a GetBRCBValues response, shall be the EntryID of the last EntryID formatted and queued for transmission.

An EntryID value of all zeros (0) is reserved to indicate an empty buffer, no reported EntryID shall have a value of zero (0).

NOTE 1 Clients needing to resynchronize to the first set of entries in the buffer should use SetBRCBValues with an EntryID whose value is all zeros (0).

NOTE 2 The resync state 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.

17.2.2.16 TimeOfEntry – time of entry

The attribute **TimeOfEntry** shall be the time at which the internal event notification was received by the report handler. This value is assigned to a specific EntryID which is also assigned at the time of internal notification receipt.

The value, returned in a GetBRCBValues response, shall provide the time stamp of the EntryID whose value is exposed in the control block. The value exposed for TimeOfEntry, when the value of EntryID is zero (0), is a local issue.

17.2.2.17 ResvTms – reservation time

The values of the attribute ResvTms are defined as follows.

– A value of –1 shall indicate that the **BRCB** is currently exclusively reserved for a set of specific clients based upon configuration.

– A value of zero (0), shall indicated that the **BRCB** is not reserved.

– A positive value shall indicate that the **BRCB** has been dynamically reserved. The value represents the number of seconds that the reservation will be maintained after association loss. Upon expiration of the reservation, the ResvTms value shall be locally set to a value of zero (0).

– A **SetBRCBValues** request, for setting **ResvTms**, shall:

- generate a negative response if the **BRCB's ResvTms** value is non-zero and if the SetBRCBValues request is being issued by another client for whom the BRCB is not reserved;
- generate a negative response if the **BRCB's ResvTms** value is –1;
- generate a negative response if the **ResvTms** value to be set is negative;
- generate a positive response if the **BRCB's ResvTms** value is zero (0) and the value being set is zero (0) or positive;
- generate a positive response if the **SetBRCBValues** request where:

  a) the **ResvTms** value, being set, is zero (0) or positive,

  b) and is from the client for which the **BRCB** has been reserved via a positive value.

17.2.2.18 Owner – the owner of the control block instance

The attribute **Owner** identifies the client which owns the control block instance. This will typically be an IP address or an IED name and is defined in the SCSMs. The value can be set at a reserved control block instance to indicate the configured owner, and shows the current owner on an activated control block. An **Owner** value of all zero (0) is used if the control block is free or the owner not identifiable.

17.2.3 BRCB class services

17.2.3.1 Overview

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

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

17.2.3.2 Report

17.2.3.2.1 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.

17.2.3.2.2 Request

17.2.3.2.2.1 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 38.

Table 38 – Report format specification

ReportFormat		
Parameter name	**Parameter type**	**Explanation**
RptID	VISIBLE STRING129 [a]	Report identification
OptFlds	[a]	Optional fields to be included in the report
IF sequence-number = TRUE in OptFlds		
SqNum	INT16U	Sequence number
SubSqNum	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	TRUE shall indicate that a buffer overflow has occurred.
If conf-revision = TRUE in OptFlds		
ConfRev	INT32U	
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	(*)	(*) type(s) depend on the definition of common data classes (CDC)
IF reason-for-inclusion = TRUE in OptFlds		
ReasonCode	ReasonForInclusion	If reason-for-inclusion (= TRUE) in OptFlds. For the definition see 6.1.2.12.

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

17.2.3.2.2.2 RptID – report ID

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

17.2.3.2.2.3 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**, **entryID, or conf-revision**) shall be included in the report.

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

17.2.3.2.2.4 SqNum – sequence number

The **BRCB** that has report enable set to TRUE shall maintain the parameter **SqNum**. 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 29 gives an example of report generation and sequence number.

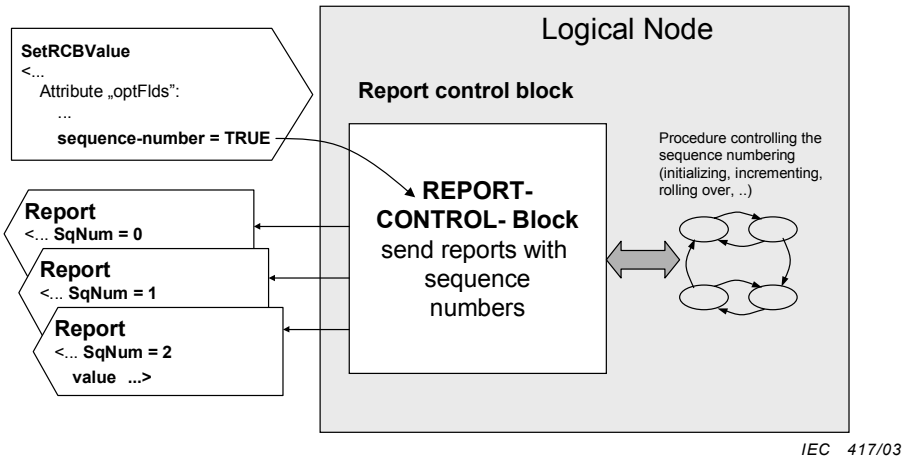


Figure 29 – Report example on the use of sequence number

17.2.3.2.2.5 SubSqNum – 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 **SubSqNum**.

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 subreport 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.

17.2.3.2.2.6 MoreSegmentsFollow – more report segments follow

The parameter **MoreSegmentsFollow** indicates that more report segments with the same sequence number follow. The **MoreSegmentsFollow** attribute shall be included in the report if the **OptFlds** attribute of the **BRCB** includes **sequence-number** (=TRUE); otherwise, it shall be omitted.

17.2.3.2.2.7 DatSet – data set reference

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

17.2.3.2.2.8 BufOvfl – possible information loss

The parameter **BufOvfl** shall indicate to the client that entries within the buffer may have been lost. The detection of possible loss of information occurs when a client requests a resynchronization to a non-existent entry or to the first entry in the queue.

The **Report Handler** shall set BufOvfl= TRUE in the first report that is sent after the transition from disabled to enabled. Subsequent reports shall have BufOvfl = FALSE.

Information can also be lost if there are resource constraints that are encountered during the **enabled** state (for example bandwidth, memory due to a high influx of notifications, etc.). Implementations shall discard the oldest entry(s) in the queue in order to accept new notifications. If one of the entries discarded causes the report handler to move the pointer to the next entry for transmission, the implementation shall indicated BufOvfl=TRUE in the next entry that is formatted and transmitted only.

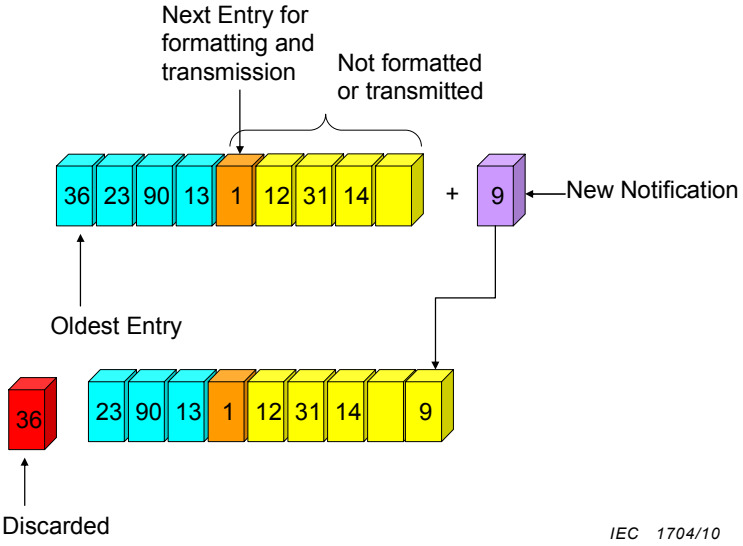


Figure 30 – Entry discard that does not cause indication of loss of information in enabled state

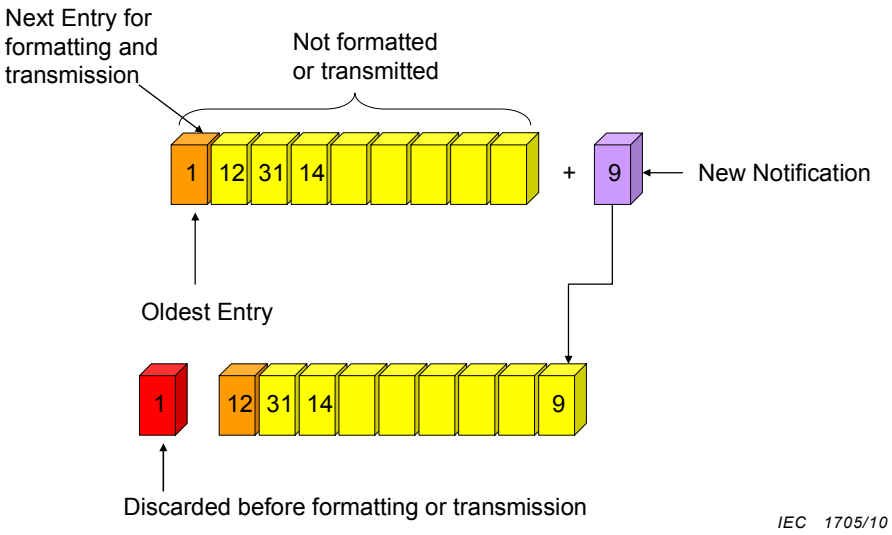


Figure 31 – Indication of loss of information due to resource constraints in enable state

Figure 30 shows an example where an entry is discarded that has already been formatted and queued for transmission and therefore **BufOvfl** shall not be set TRUE.

Figure 31 shows an example where an entry is discarded that has not been formatted and queued for transmission. Therefore, **BufOvfl** shall be set TRUE in the next report generated for transmission.

17.2.3.2.2.9 Entry

TimeOfEntry – report time stamp

The parameter **TimeOfEntry** shall specify the time when the EntryID was created. 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

For the definition, see 17.2.2.15.

EntryData [1..n]

The parameter **EntryData** shall contain the data reference **DataRef, 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**.

a) **DataRef**
The parameter DataRef shall contain the reference of the DataSet member that is being reported.

b) **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 notifications. In case of (BufTm = 0), only the value(s) of the member(s) of a DATA-SET shall be included that produced the EntryID.

EXAMPLE The data attribute stVal of the data object MyLD/**XCBR1.Pos** (Position) in Figure 32 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 object 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**, ...).

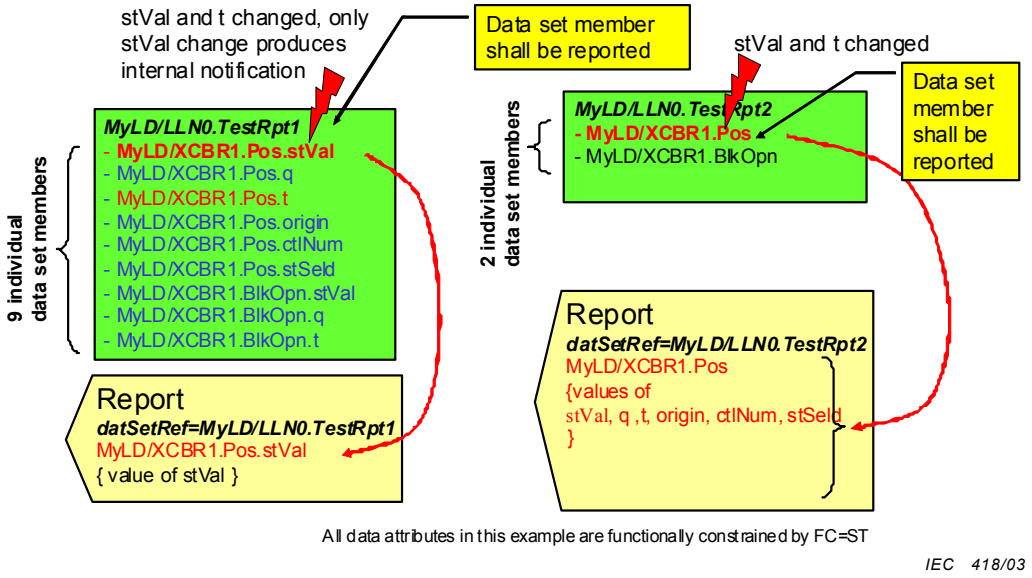


Figure 32 – Data set members and reporting

BufTm > 0

In the case of (**BufTm** > 0), the values of all members of a **data-set** containing attributes that have changed with respect to TrgOps during the bufferTime interval shall be included that produced internal notifications during the buffer time. Further constraints apply; see 17.2.2.9 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 **TrgOps** that caused the creation of the report. The value range for reasons for inclusion shall be as defined in 6.1.2.12.

17.2.3.2.3 Procedures for report generation

17.2.3.2.3.1 Overview

Figure 33 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 Figure 33 does not show all attributes and not all details..

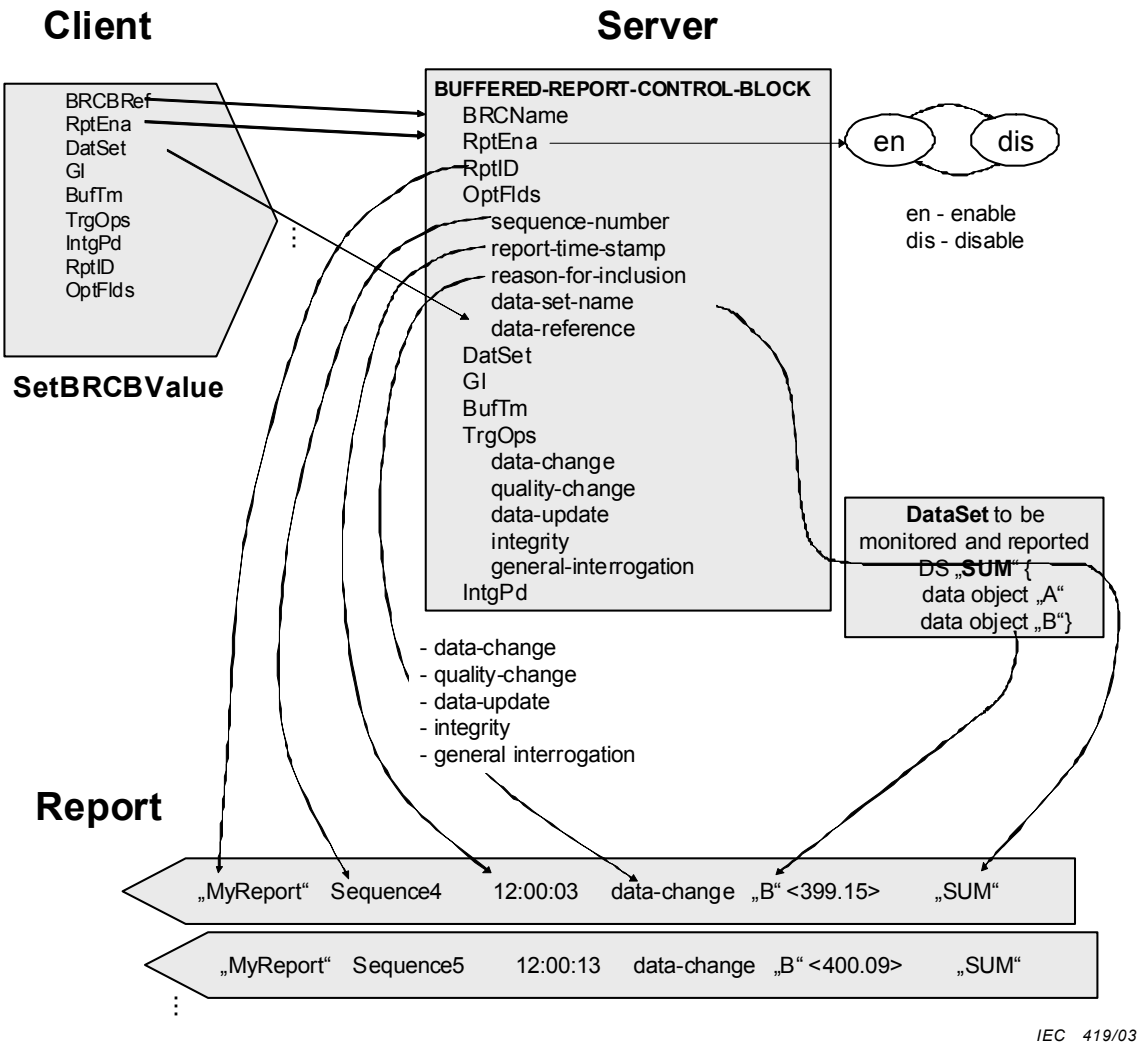


Figure 33 – Report example

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.

17.2.3.2.3.2 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** (**TrgOps**.**dchg** is TRUE) relates to a change in a value of a **DataAttribute** representing the process-related value of the data object. If the **TrgOps**.**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** (**TrgOps**.**qchg=TRUE**) relates to a change in the quality value of a **DataAttribute**. If the **TrgOps**.**qchg** is FALSE, then no report should be issued on a **quality-change** in the value of that **DataAttribute**.

data-update

The trigger option **data-update** (**TrgOps**.**dupd=TRUE**) relates to a freeze event in a value of a **DataAttribute** representing a freeze value of the data object (for example, frozen counters) or to an event triggered by updating the value of a **DataAttribute**. If the **TrgOps**.**dupd** is FALSE, then no report should be issued on a **data-update** 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 The trigger option applying to a specific **DataAttribute** is defined in the appropriate common data class (CDC).

When the **BRCB** is notified by an internal notification of a **data-change**, **quality-change**, or **data-update** event of a member(s) of the referenced **data-set** whose values are to be reported, the **BRCB** shall include the value of the member(s) of the referenced **data-set** that produced the internal notification in the report according to 17.2.3.2.2.9. The value to be reported shall be the value that produced the internal notification.

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

17.2.3.2.3.3 Integrity

The trigger option **integrity** supports integrity report generation. In addition, to activate this trigger option (set **TrgOps 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 **TrgOps** (= **integrity**) is FALSE or if IntgPd is zero, no integrity report should be issued.

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

A new internal notification 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). No other reports shall be transmitted until the entire integrity report has transmitted.

A new notification 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 notification 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.

17.2.3.2.3.4 GI (general interrogation)

The attribute **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 **TrgOps**.**general-interrogation** is FALSE, then no general-integrity report should be issued.

All buffered entries 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.

A new notification 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**.

17.2.3.2.3.5 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 entries have been created.

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 0) shall be sent until all values have been sent. If – while sending these reports or subreports respectively – values of data objects caused by **data-change**, **quality**-**change**, or **data-update** need to be sent, this shall be done with a new report sent after the transmission of the **integrity** or **general-interrogation** report.

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 object 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 object, the client may not use this information to update the process data image.

17.2.3.2.3.6 Buffering events

The **BRCB** shall buffer entries based on the trigger options **data-change**, **quality-change**, **data-update,** and **integrity** during all states: disabled, resync, and enabled.

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 entries that have been buffered. The **BRCB** shall use the sequence and subsequence numbers so that no gaps occur.

NOTE 1 Since the buffer entries based on the trigger option integrity are buffered by the BRCB, and the memory of the IED dedicated for the buffering is limited, it is recommended to use the trigger option integrity in the BRCB with great care, to avoid a BufOvfl, and keep a long historical of the entries.

NOTE 2 Server implementations determine the number of entries that can be buffered. During association loss, the number of entries may increase to a point that upon transition to the **enabled** state, reporting of all the entries may take a long time. Client and server implementations should attempt to address this issue through high-priority formatting and queuing for transmission of the entries or limiting the number of entries.

17.2.3.3 GetBRCBValues

17.2.3.3.1 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**.

Parameter name
Request
BRCBReference
Response+
ReportIdentifier
ReportEnable
DataSetReference
ConfigurationRevision
OptionalFields
BufferTime
SequenceNumber
TriggerOptionsEnabled
IntegrityPeriod
GeneralInterrogation
PurgeBuf
EntryIdentifier
TimeOfEntry
ReserveTimeSecond [0..1]
Response–
ServiceError

17.2.3.3.2 Request

17.2.3.3.2.1 BRCBReference

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

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

17.2.3.3.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. All SCSMs shall be able to return the Response+ parameter values of the following.

17.2.3.3.3.1 ReportIdentifier

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

17.2.3.3.3.2 ReportEnable

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

17.2.3.3.3.3 DataSetReference

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

17.2.3.3.3.4 ConfigurationRevision

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

17.2.3.3.3.5 OptionalFields

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

17.2.3.3.3.6 BufferTime

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

17.2.3.3.3.7 SequenceNumber

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

17.2.3.3.3.8 TriggerOptionsEnabled

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

17.2.3.3.3.9 IntegrityPeriod

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

17.2.3.3.3.10 GeneralInterrogation

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

17.2.3.3.3.11 PurgeBuf

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

17.2.3.3.3.12 EntryIdentifier

The parameter **EntryIdentifier** shall contain the value of the corresponding attribute **EntryID** of the referenced **BRCB** (see 17.2.2.15).

17.2.3.3.3.13 TimeOfEntry

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

17.2.3.3.3.14 ReserveTimeSecond [0..1]

The parameter **ReserveTimeSecond** shall contain the value of the corresponding attribute **ResvTms** of the referenced **BRCB**.

17.2.3.3.4 Response–

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

17.2.3.4 SetBRCBValues

17.2.3.4.1 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**.

Parameter name
Request
BRCBReference
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]
ReserveTimeSecond [0..1]
Response+
Response–
ServiceError

SetBRCBValues.request shall be processed atomically by the server. If the request contains any parameters, whose sets fails, a Response– shall be returned.

17.2.3.4.2 Request

17.2.3.4.2.1 BRCBReference

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

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

17.2.3.4.2.2 ReportIdentifier [0..1]

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

17.2.3.4.2.3 ReportEnable [0..1]

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

17.2.3.4.2.4 DataSetReference [0..1]

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

17.2.3.4.2.5 OptionalFields [0..1]

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

17.2.3.4.2.6 BufferTime [0..1]

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

17.2.3.4.2.7 TriggerOptionsEnabled [0..1]

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

17.2.3.4.2.8 IntegrityPeriod [0..1]

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

17.2.3.4.2.9 GeneralInterrogation [0..1]

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

17.2.3.4.2.10 PurgeBuffer [0..1]

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

17.2.3.4.2.11 EntryIdentifier[0..1]

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

17.2.3.4.2.12 ReserveTimeSecond[0..1]

The parameter **ReserveTimeSecond** shall contain the value of the corresponding attribute **ResvTms** of the referenced **BRCB**.

17.2.3.4.3 Response+

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

17.2.3.4.4 Response–

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

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

17.2.4.1 URCB class syntax

The **URCB** class shall have the structure defined in Table 39.

Table 39 – URCB class definition

URCB class			
Attribute name	**Attribute type**	**r/w**	**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 STRING129	r/w	c1
RptEna	BOOLEAN	r/w	
Resv	BOOLEAN	r/w	
DatSet	ObjectReference	r/w	c1
ConfRev	INT32U	r	
OptFlds	PACKED LIST	r/w	c1
sequence-number	BOOLEAN		
report-time-stamp	BOOLEAN		
reason-for-inclusion	BOOLEAN		
data-set-name	BOOLEAN		
data-reference	BOOLEAN		
buffer-overflow	BOOLEAN		
entryID	BOOLEAN		c2
conf-revision	BOOLEAN		
BufTm	INT32U	r/w	c1
SqNum	INT8U	r	
TrgOps	TriggerConditions	r/w	c1
IntgPd	INT32U	r/w	0.. MAX; 0 implies no integrity report.
GI	BOOLEAN	r/w	
Owner	OCTET STRING64	r	c3
Services Report GetURCBValues SetURCBValues			
Notes and conditions **c1:** These attributes may only be set when RptEna = FALSE. A SetURCBValues of these parameters, when RptEna=TRUE, shall fail. **c2:** It is a local issue if a SetURCBValues with OptFlds.EntryID = TRUE should result with a GetURCBValues of OptFlds.EntryID = TRUE. It is suggested that a GetURCBValues of OptFlds.EntryID return a value of FALSE for OptFlds.EntryID. c3: This attribute is optional. NOTE An attribute that is marked “r” indicates that the URCB attribute may be obtained (for example read) through the use of the GetURCBValues service. An attribute that is marked “w” indicates that the URCB attribute may be set (for example written) through the use of the SetURCBValues service.			

Except **URCBName**, **URCBRef**, **RptEna**, **SqNum**, and **Resv**, all other attributes shall be as defined for the **BRCB** in 17.2.2.

17.2.4.2 URCBName – unbuffered report control name

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

17.2.4.3 URCBRef – unbuffered report control ObjectReference

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

The **ObjectReference URCBRef** shall be:

LDName/LNName.URCBName

17.2.4.4 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.

17.2.4.5 Resv – reserve URCB

The attribute **Resv** (if set to TRUE) shall indicate that the **URCB** is currently exclusively 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** works as a semaphore for the configuration, reserving and un-reserving the **URCB**.

17.2.5 URCB class services

17.2.5.1 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**

17.2.5.2 Report

The report service shall be as defined for **BRCB** in 17.2.3.2, with exception of:

– the parameters of BufOvfl and EntryID of the report format shall not be available,
– regardless of the value of OptFlds.EntryID, no EntryID shall be included in the report,
– regardless of the value of OptFlds.BufOvfl, no BufOvfl shall be included in the report,
– the SqNum is formated to an INT8U.

17.2.5.3 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 17.2.3.3, except that the parameter **BRCBReference** shall be **URCBReference**, and the parameters **PurgeBuffer** and ResvTms shall not be available – instead Resv is returned.

17.2.5.4 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 17.2.3.4, except that the parameter **BRCBReference** shall be **URCBReference**, and the parameters **PurgeBuffer** and ResvTms shall not be available – instead Resv could be set.

17.3 LOG-CONTROL-BLOCK class model

17.3.1 General

17.3.1.1 Basic concepts

Many IEDs have requirements for the internal storage of historical values of data objects and retrieval over communications systems. These 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 objects. Several criteria are used to differentiate requirements for logging of values of data objects (historical values) from immediate report-oriented exchange of values.

– Data object 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 values of data objects 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.

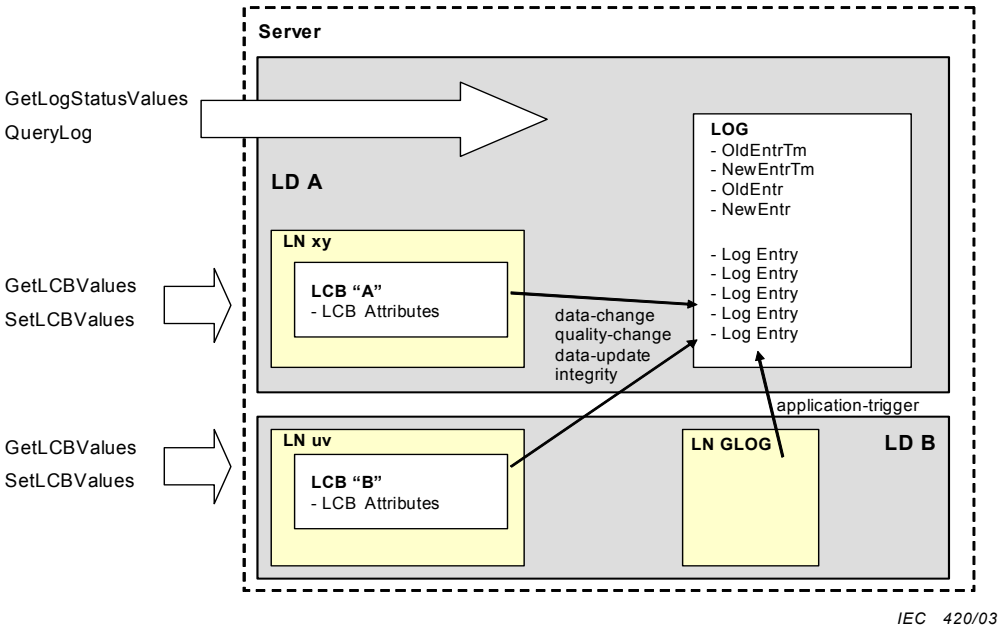


Figure 34 – Log model overview

Figure 34 gives an overview of the **LOG** and **LCB** classes. One **LOG** may be written by multiple **LCBs**.

17.3.1.2 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**.

17.3.2 LCB class definition

17.3.2.1 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 entries. 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 40.

Table 40 – LCB class definition

LCB class			
Attribute name	**Attribute type**	**r/w**	**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	r/w	
DatSet	ObjectReference	r/w	
OptFlds	PACKED LIST	r/w	
reason-for-inclusion	BOOLEAN		
BufTm	INT32U	r/w	
TrgOps	TriggerConditions	r/w	Valid values for TrgOps of type TriggerConditions shall be dchg, qchg, dupd, and integrity
IntgPd	INT32U	r/w	1..MAX; 0 implies no integrity logging.
Specific to building the log			
LogRef	ObjectReference	r/w	
Services GetLCBValues SetLCBValues			

17.3.2.2 LCB class attributes

17.3.2.2.1 LCBName – log control name

The attribute **LCBName** shall unambiguously identify a **LCB** within the scope of a **logical-node**.

17.3.2.2.2 LCBRef – log control ObjectReference

The attribute **LCBRef** shall be the unique path-name of a **LCB**.

The ObjectReference **LCBRef** shall be:

LDName/LNName.LCBName

17.3.2.2.3 LogEna – log enable

The attribute **LogEna** shall indicate that this **LCB** is recording into the **LOG** specified by **LogRef**.

NOTE 1 The Tracking Services for the Log may be used to log or report when **LogEna** transits from disabled to enabled or from enabled to disabled.

NOTE 2 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.

17.3.2.2.4 DatSet – data set reference

The attribute **DatSet** shall indicate the **data-set**, whose member values are to be logged.

17.3.2.3 OptFlds – optional fields to include in log

The attribute **OptFlds** shall be the client-specified optional fields to be included in the log issued by this **LCB**. This attribute defines a subset of the optional header fields of the log **EntryData** (see 17.3.3.1) that shall be included in the log:

– reason-for-inclusion (if TRUE **ReasonCode** shall be included in the log).

If a **LCB** does not support the above option, then an attempt to set the corresponding bit to TRUE shall cause a negative response of the **SetLCBValues** service.

17.3.2.3.1 BufTm – buffer time

This attribute shall be the same as defined in 17.2.2.9.

17.3.2.3.2 TrgOps – trigger options

The attribute **TrgOps** shall specify the trigger conditions that shall be monitored by this **LCB** to cause a Log entry to be created. 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 (CDC). The trigger option **integrity** shall be the trigger condition defined by the attributes **IntgPd** of the **LCB**.

The **TrgOps general-interrogation** shall not be supported for logging.

17.3.2.3.3 IntgPd – integrity period

If **TrgOps** is set to **integrity**, the attribute **IntgPd** indicates the period in milliseconds used for logging caused by integrity scans.

17.3.2.3.4 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.

17.3.2.4 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**

17.3.2.5 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**.

Parameter name
Request
LCBReference
Response+
LogEnable
DataSetReference
TriggerOptions
IntegrityPeriod
LogReference
OptionalFields [0..1]
BufferTime [0..1]
Response–
ServiceError

17.3.2.5.1 Request

17.3.2.5.1.1 LCBReference

The parameter **LCBReference** shall specify the **ObjectReference** of the **LCB**.

The service parameter **LCBReference** shall be **LCBRef**.

17.3.2.5.2 Response+

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

17.3.2.5.2.1 LogEnable

The parameter **LogEnable** shall contain the value for the corresponding attribute **LogEna** of the referenced **LCB**.

17.3.2.5.2.2 DataSetReference

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

17.3.2.5.2.3 TriggerOptions

The parameter **TriggerOptions** shall contain the value for the corresponding attribute **TrgOps** of the referenced **LCB**.

17.3.2.5.2.4 IntegrityPeriod

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

17.3.2.5.2.5 LogReference

The parameter **LogReference** shall contain the value for the corresponding attribute **LogRef** of the referenced **LCB**.

17.3.2.5.2.6 OptionalFields [0..1]

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

17.3.2.5.2.7 BufferTime [0..1]

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

17.3.2.5.3 Response–

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

17.3.2.6 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**.

Parameter name
Request
LCBReference
LogEnable [0..1]
DataSetReference [0..1]
OptionalFields [0..1]
IntegrityPeriod [0..1]
LogReference [0..1]
TriggerOptionsEnabled [0..1]
BufferTime [0..1]
Response+
Response–
ServiceError

SetLCBValues.request shall be processed atomically by the server. If the request contains any parameters, whose sets fails, a **Response–** shall be returned.

17.3.2.6.1 Request

17.3.2.6.1.1 LCBReference

The parameter **LCBReference** shall specify the ObjectReference of **LCB**.

The service parameter **LCBReference** shall be **LCBRef**.

17.3.2.6.1.2 LogEnable [0..1]

The parameter **LogEnable** shall contain the value of the corresponding attribute **LogEna** of the referenced **LCB**.

17.3.2.6.1.3 DataSetReference [0..1]

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

17.3.2.6.1.4 OptionalFields [0..1]

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

17.3.2.6.1.5 IntegrityPeriod [0..1]

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

17.3.2.6.1.6 LogReference [0..1]

The parameter **LogReference** shall contain the value of the corresponding attribute **LogRef** of the referenced **LCB**.

17.3.2.6.1.7 TriggerOptionsEnabled [0..1]

The parameter **TriggerOptionsEnabled** shall contain the value of the corresponding attribute **TrgOps** of the referenced **LCB**.

17.3.2.6.1.8 BufferTime [0..1]

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

17.3.2.6.2 Response+

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

17.3.2.6.3 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 LCB is enabled and the service has been issued for any other attribute of the **LCB** as LogEna.

17.3.3 LOG class definition

17.3.3.1 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 values of data objects 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. Log entries shall be persistently stored, even in case of loss of power supply or IED restart.

The **LOG** shall have the structure defined in Table 41.

Table 41 – LOG class definition

LOG class			
Attribute name	**Attribute type**	**r/w**	**Value/value range/explanation**
LogName	ObjectName		Instance name of an instance of LOG
LogRef	ObjectReference		Path-name of an instance of LOG
OldEntrTm	TimeStamp	r	
NewEntrTm	TimeStamp	r	
OldEntr	INT32U	r	
NewEntr	INT32U	r	
Entry [1..n]			
TimeOfEntry	EntryTime		
EntryID	EntryID		
EntryData [1..n]			
DataRef	ObjectReference		
Value	(*)		(*) type(s) depend on the definition of the concerned common data classes (CDC)
ReasonCode	ReasonForInclusion		If reason-for-inclusion (=TRUE) in optFlds. ReasonCode general-interrogation shall never occur as TRUE. For the definition, see 6.1.2.12.
Services			
QueryLogByTime QueryLogAfter GetLogStatusValues			

17.3.3.2 LOG class attributes

17.3.3.2.1 LogName – log name

The attribute **LogName** shall unambiguously identify a **LOG** within the scope of **a Logical Node**.

17.3.3.2.2 LogRef – log reference

The attribute **LogRef** shall be the unique path-name of a **LOG**.

The **ObjectReference LogRef** shall be:

LDName/LNName.LogName

17.3.3.2.3 OldEntrTm – oldest log entry time of LOG

The attribute **OldEntrTm** shall indicate the time when the oldest log entry available in the **LOG** 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.

17.3.3.2.4 NewEntrTm – newest log entry time of LOG

The attribute **NewEntrTm** shall indicate the time when the newest log entry available in the **LOG** has been stored.

17.3.3.2.5 OldEntr – oldest log entry sequence number

The attribute **OldEntr** shall indicate the **EntryID** of the oldest entry available in the **LOG**.

17.3.3.2.6 NewEntr – newest log entry sequence number

The attribute **NewEntr** shall indicate the **EntryID** of the newest entry available in the **LOG**.

17.3.3.2.7 Entry [1..n]

17.3.3.2.7.1 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 value of a data object, which shall be the time when the event occurred that caused the log entry to be created.

17.3.3.2.7.2 EntryID – entry identifier

The attribute **EntryID** shall be a unique reference to all log entries having the same value of TimeOfEntry.

17.3.3.2.7.3 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**. The values stored in the log shall be non-volatile.

DataRef

The parameter **DataRef** shall contain the DataSet member reference of the value of the EntryData.

Value

The parameter value shall contain the DataSet member values to be included in the EntryData. There shall be one and only one EntryData per DataRef.

The number of members of the **data-set** whose values shall be included in the log shall depend on the **TrgOps** of the **LCB** selected and the following values of **TrgOps** of the respective **DataAttributes**:

In case of **TrgOps** (**dchg**, **qchg**, or **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 a value greater than zero (0) and **TrgOps integrity** (=TRUE), **all** values of **all** members of a **data-set** shall be included in the log entry that was produced by the internal event.

In case of **ReasonCode application-trigger**, it is up to the application to provide the entry data.

NOTE The application-trigger is independent of a log control block. There is no data set involved.

ReasonCode – reason for inclusion

The reason for inclusion shall be set according the **TrgOps** that caused the creation of the **EntryData**. The value for reason for inclusion shall be set according the **TrgOps** that caused the creation of the log entry. The value range for reasons for inclusion shall be as listed:

- **data-change** (caused by TrgOps = **dchg** in an instance of a data object class),
- **quality-change** (caused by TrgOps **= qchg** in an instance of a data object class),
- **data-update** (caused by TrgOps = **dupd** in an instance of of a data object class),
- **integrity** (caused by the attribute **IntgPd** in the **LCB**),
- **application-trigger** (caused by a trigger defined by an application).

17.3.4 Reason code for log entries

17.3.4.1 Overview

Basically, the conditions and constraints for log generation shall be the same as for report generation (see 17.2.3.2.3; 17.3.4 specifies the differences only).

17.3.4.2 Reason code data-change, quality-change, or data-update

When the **LCB** is notified by an internal event of a data-change, a quality-change, or 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.

17.3.4.3 Reason code integrity

When a **LCB** is notified as a result of the trigger option integrity, the **LCB** shall create a single LOG entry comprising values of all members the referenced **data-set**.

17.3.4.4 Reason code application-trigger

When an application-trigger is issued by the application, the application shall create a single LOG entry comprising consistent values.

NOTE The application is responsible for triggering the log entry to be logged and for providing consistent values.

17.3.5 LOG services

17.3.5.1 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**

17.3.5.2 QueryLogByTime

17.3.5.2.1 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 [0..1]
RangeStopTime [0..1]
Response+
ListOfLogEntries
Response–
ServiceError

17.3.5.2.2 Request

17.3.5.2.2.1 LogReference

The parameter **LogReference** shall contain the **ObjectReference LogRef** of a **LOG**. The **ObjectReference LogReference** shall be:

LDName/LNName.LogName

17.3.5.2.2.2 RangeStartTime [0..1]

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 **TimeOfEntry** 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.

17.3.5.2.2.3 RangeStopTime [0..1]

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 **TimeOfEntry** 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.

17.3.5.2.3 Response+

ListOfLogEntries

The parameter **ListOfLogEntries** shall contain the list of log entries whose **TimeOfEntry** is in the range as specified with the parameters **RangeStartTime** and **RangeStopTime** of the service request.

17.3.5.2.4 Response–

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

17.3.5.3 QueryLogAfter

17.3.5.3.1 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

17.3.5.3.2 Request

17.3.5.3.2.1 LogReference

The parameter **LogReference** shall specify the **ObjectReference LogRef** of the **LOG**. The **ObjectReference LogReference** shall be:

LDName/LNName.LogName

17.3.5.3.2.2 RangeStartTime

The parameter **RangeStartTime** shall contain the time of the log entry selected.

NOTE There might be several entries for the same time.

17.3.5.3.2.3 Entry

The parameter **Entry** shall reference the **LOG** entry of the selected **RangeStartTime** after which the log entries shall be selected.

17.3.5.3.3 Response+

17.3.5.3.3.1 ListOfLogEntries

The parameter **ListOfLogEntries** shall contain the list of log entries that follow after the log entry as specified with the parameters **RangeStartTime** and **Entry** of the service request.

17.3.5.3.4 Response–

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

17.3.5.4 GetLogStatusValues

A client shall use the **GetLogStatusValues** service to retrieve the attribute values of a **LOG** made visible and thus accessible to the requesting client by the referenced **Logical-Node**.

Parameter name
Request
LogReference
Response+
OldestEntryTime
NewestEntryTime
OldestEntry
NewestEntry
Response–
ServiceError

17.3.5.4.1 Request

17.3.5.4.1.1 LogReference

The parameter **LogReference** shall specify the **ObjectReference** of the **LOG**.

The service parameter **LogReference** shall be:

LDName/LNName.LogName

17.3.5.4.2 Response+

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

17.3.5.4.2.1 OldestEntryTime

The parameter **OldestEntryTime** shall contain the value for the corresponding attribute **OldEntrTm** of the referenced **LOG**.

17.3.5.4.2.2 NewestEntryTime

The parameter **NewestEntryTime** shall contain the value for the corresponding attribute **NewEntrTm** of the referenced **LOG**.

17.3.5.4.2.3 OldestEntry

The parameter **OldestEntry** shall contain the value for the corresponding attribute **OldEntr** of the referenced **LOG**.

17.3.5.4.2.4 NewestEntry

The parameter **NewestEntry** shall contain the value for the corresponding attribute **NewEntr** of the referenced **LOG**.

17.3.5.4.3 Response–

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

18 Generic substation event class model (GSE)

18.1 Overview

The generic substation event model provides the possibility for a fast and reliable system-wide distribution of input and output 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 publishing 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.

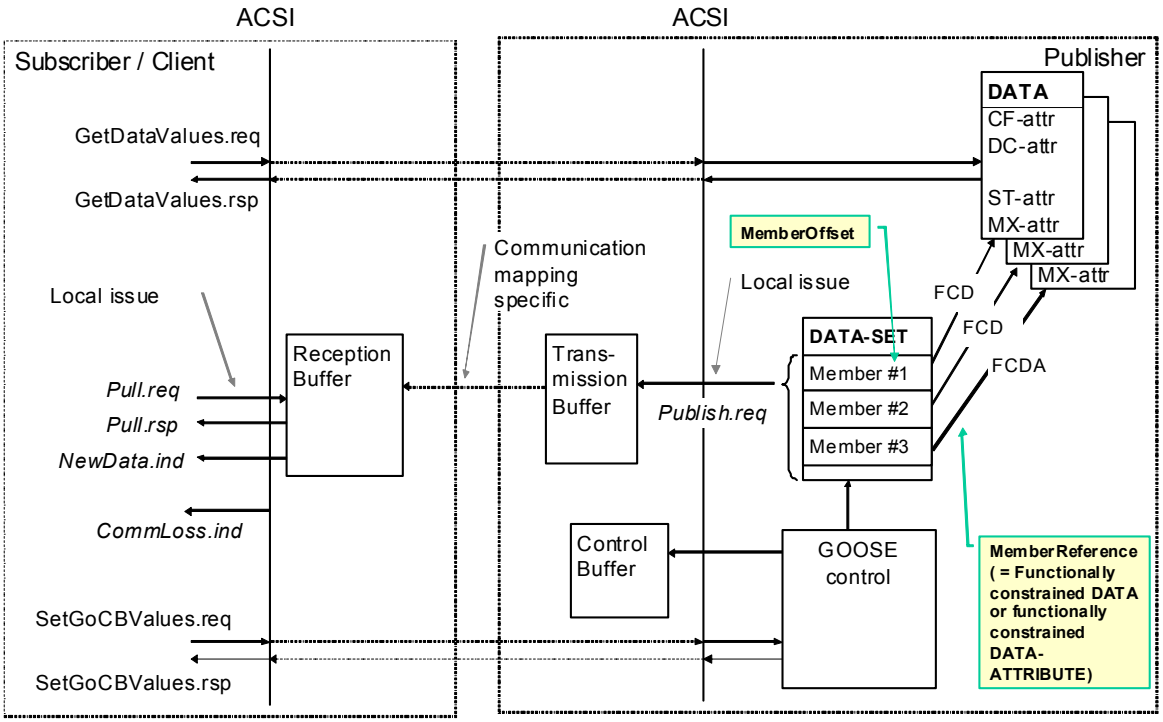
The generic substation event model applies to the exchange of values of a collection of **DataAttribute** respective **FCD / FCDA**. Two control classes and the structure of two messages are defined in this clause:

– 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. Its usage is deprecated in this version of IEC 61850 and therefore moved to Annex B.

The information exchange is based on a publisher/subscriber mechanism. The publisher writes the values into a local buffer at the sending side; the subscriber 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.



IEC 421/03

Figure 35 – GoCB model

Figure 35 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** changes, the transmission buffer of the publisher is updated with the local service "publish", and all 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 the current value of a data object (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.

18.2 GOOSE-CONTROL-BLOCK (GoCB) class

18.2.1 GoCB definition

The **GoCB** shall be as defined in Table 42.

Table 42 – GOOSE control block class definition

GoCB class			
Attribute name	**Attribute type**	**r/w**	**Value/value range/explanation**
GoCBName	ObjectName		Instance name of an instance of GoCB
GoCBRef	ObjectReference		Path-name of an instance of GoCB
GoEna	BOOLEAN	r/w	Enabled (TRUE) | disabled (FALSE)
GoID	VISIBLE STRING129	r/w	Attribute that allows a user to assign an identification for the GOOSE message
DatSet	ObjectReference	r/w	
ConfRev	INT32U	r	
NdsCom	BOOLEAN	r	
DstAddress	PHYCOMADDR	r	
Services SendGOOSEMessage GetGoReference GetGOOSEElementNumber GetGoCBValues SetGoCBValues			

18.2.1.1 GoCBName – GOOSE control block name

The attribute **GoCBName** shall unambiguously identify a **GoCB** within the scope of a **LLN0**.

18.2.1.2 GoCBRef – GOOSE control block reference

The attribute **GoCBRef** shall be the unique path-name of a **GoCB** within the **LLN0**.

The **ObjectReference GoCBRef** shall be:

LDName/LLN0.GoCBName

18.2.1.3 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.

If there are inconsistent attribute values in the GoCB (for example the value of DatSet is Null) or if the value of ConfRev equals 0, a SetGoCBValues with the parameter GoEna equals TRUE shall fail and a negative response shall be issued.

While being TRUE (**GoCB** enabled), no change of attribute values of the **GoCB** other than disabling shall be allowed.

The value of GoEna at IED startup depends on the IED configuration.

18.2.1.4 GoID – GOOSE identifier

The attribute **GoID** shall be a user definable identification of the GOOSE message.

18.2.1.5 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 functional constraint **data** (**FCD**) or **DataAttribute** (**FCDA**)).

NOTE The service **GetGoReference** 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.

18.2.1.6 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**;

– Any adding of a member to 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. At configuration time, the configuration tool will be responsible for incrementing/maintaining the ConfRev value. When configuration changes occur due to SetGoCBValues, the IED shall be responsible for incrementing the value of ConfRev.

If the value of **DatSet** is set through a SetGoCBValues service to the same value, the ConfRev value shall still be incremented.

The initial value for **ConfRev** is outside the scope of this part of IEC 61850. The value of 0 shall be reserved. The value of ConfRev, upon a restart of the IED, is a local issue.

18.2.1.7 NdsCom – needs commissioning

The attribute **NdsCom** shall have a value of TRUE if the **GoCB** requires further configuration.

Examples, where further configuration is required are:

– the attribute DataSet has a value of NULL,

– the number or size of values being conveyed by the elements in the **DatSet** referenced **data-set** exceeds constraint determined by the SCSM or the implementation.

18.2.1.8 DstAddress

The attribute DstAddress shall be the SCSM specific addressing information like media access address, priority, and other information.

18.2.2 GOOSE service definitions

18.2.2.1 Overview

For the **GoCB**, the following services are defined:

Service	Description
SendGOOSEMessage	Send GOOSE message
GetGoReference	Retrieve the **FCD/FCDA** and DatSetReference 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**

NOTE The services GetGoReference and GetGOSEElementNumber are used to validate the actual configuration of the publisher versus what the subscriber is expecting to receive. These services provide an alternate solution than reading GoCB and DATA-SET definitions and can be used by SCSMs for alternate mappings.

18.2.2.2 SendGOOSEMessage

18.2.2.2.1 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

18.2.2.2.2 Request

The parameter **GOOSE message** shall specify the **GOOSE** message as defined in 18.2.3 of the given **GoCB**.

18.2.2.3 GetGoReference

18.2.2.3.1 GetGoReference parameter table

A client shall use the **GetGoReference** 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
DatSet
MemberReference [1..n]
Response–
ServiceError

18.2.2.3.2 Request

18.2.2.3.2.1 GoCBReference

The parameter **GoCBReference** shall identify the attribute **GoCBRef** of the **GoCB** for which **MemberReferences** are being requested.

18.2.2.3.2.2 MemberOffset [1..n]

The parameter **MemberOffset** shall contain a number identifying a member of the **data-set** referenced by the attribute **DatSet**.

18.2.2.3.3 Response+

18.2.2.3.3.1 GoCBReference

The parameter **GoCBReference** shall contain the parameter that identifies the attribute **GoCBRef** of the **GoCB** for which **MemberReferences** are returned.

18.2.2.3.3.2 ConfigurationRevision

The parameter **ConfigurationRevision** shall contain the attribute **ConfRev** of the **GoCB**.

18.2.2.3.3.3 DatSet

The parameter DatSet shall contain the attribute DatSet of the GoCB.

18.2.2.3.3.4 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**.

18.2.2.3.4 Response–

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

18.2.2.4 GetGOOSEElementNumber

18.2.2.4.1 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
DatSet
MemberOffset [1..n]
Response–
ServiceError

18.2.2.4.2 Request

18.2.2.4.2.1 GoCBReference

The parameter **GoCBReference** shall identify the attribute **GoCBRef** of the **GoCB** for which **MemberOffset** are being requested.

18.2.2.4.2.2 MemberReference [1..n]

The parameter **MemberReference** shall contain the **MemberReference** for which the **MemberOffset** of a member of the **data-set** is requested.

18.2.2.4.3 Response+

18.2.2.4.3.1 GoCBReference

The parameter **GoCBReference** shall contain the parameter that identifies the attribute **GoCBRef** of the **GoCB** for which **MemberOffsets** are returned.

18.2.2.4.3.2 ConfigurationRevision

The parameter **ConfigurationRevision** shall contain the attribute **ConfRev** of the **GoCB**.

18.2.2.4.3.3 DatSet

The parameter DatSet shall contain the attribute DatSet of the GoCB.

18.2.2.4.3.4 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**.

18.2.2.4.4 Response–

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

18.2.2.5 GetGoCBValues

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

Parameter name
Request
GoCBReference
Response+
GoEnable
GOOSEID
DataSetReference
ConfigurationRevision
NeedsCommissioning
DestinationAddress [0..1]
Response–
ServiceError

18.2.2.5.1 Request

18.2.2.5.1.1 GoCBReference

The parameter **GoCBReference** shall specify the ObjectReference of the **GoCB** and shall be **LDName/LLN0.GoCBName**.

18.2.2.5.2 Response+

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

18.2.2.5.2.1 GoEnable

The parameter **GoEnable** shall contain the value of the corresponding attribute **GoEna** of the referenced **GoCB**.

18.2.2.5.2.2 GOOSEID

The parameter **GOOSEID** shall contain the value of the corresponding attribute **GoID** of the referenced **GoCB**.

18.2.2.5.2.3 DataSetReference

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

18.2.2.5.2.4 ConfigurationRevision

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

18.2.2.5.2.5 NeedsCommissioning

The parameter **NeedsCommissioning** shall contain the value of the corresponding attribute **NdsCom** of the **GoCB**.

18.2.2.5.2.6 DestinationAddress [0..1]

The parameter **DestinationAddress** shall contain the value of the corresponding attribute **DstAddress** of the **GoCB**.

18.2.2.5.3 Response–

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

18.2.2.6 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**.

Parameter name
Request
GoCBReference
GoEnable [0..1]
GOOSEID [0..1]
DataSetReference [0..1]
Response+
Response–
ServiceError

18.2.2.6.1 Request

18.2.2.6.1.1 GoCBReference

The parameter **GoCBReference** shall specify the **ObjectReference** of the **GoCB** and shall be **LDName/LLN0.GoCBName**.

18.2.2.6.1.2 GoEnable [0..1]

The parameter **GoEnable** shall contain the value for the corresponding attribute **GoEna** of the referenced **GoCB**.

18.2.2.6.1.3 GOOSEID [0..1]

The parameter **GOOSEID** shall contain the value for the corresponding attribute **GoID** of the referenced **GoCB**.

18.2.2.6.1.4 DataSetReference [0..1]

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

18.2.2.6.2 Response+

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

18.2.2.6.3 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.

18.2.3 Generic object oriented substation event (GOOSE) message

18.2.3.1 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 43.

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 43 – GOOSE message definition

GOOSE message		
Parameter name	**Parameter type**	**Value/value range/explanation**
DatSet	ObjectReference	Value from the instance of GoCB
GoID	VISIBLE STRING129	Value from the instance of GoCB
GoCBRef	ObjectReference	Value from the instance of GoCB
T	TimeStamp	
StNum	INT32U	
SqNum	INT32U	
Simulation	BOOLEAN	(TRUE) simulation | (FALSE) real values
ConfRev	INT32U	Value from the instance of GoCB
NdsCom	BOOLEAN	Value from the instance of GoCB
GOOSEData [1..n]		
Value	(*)	(*) type depends on the appropriate common data classes (CDC).

18.2.3.2 DatSet – data set

The parameter **DatSet** shall contain the value of the attribute **DatSet** of the **GOOSE** control block.

18.2.3.3 GoID – application identifier

The parameter **GoID** shall contain the value of the attribute **GoID** of the GoCB.

18.2.3.4 GoCBRef – GOOSE control block reference

The parameter **GoCBRef** shall contain the reference of the **GOOSE** control block.

18.2.3.5 T – time stamp

The parameter **T** shall contain the time at which the attribute **StNum** was incremented.

18.2.3.6 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** upon a transition of GoEna to TRUE shall be 1. The value of 0 shall be reserved.

NOTE A transition of GoEna to TRUE happens directly at power-up or restart for properly and consistently configured GoCB.

18.2.3.7 SqNum – sequence number

The parameter **SqNum** shall contain the counter that shall increment each time a **GOOSE** message has been sent.

Following a **StNum** change, the counter **SqNum** shall be set to a value of 0. The initial value for **SqNum** upon a transition of GoEna to TRUE is recommended to be 1.

18.2.3.8 Simulation – Simulation

The parameter **Simulation** shall indicate with the value TRUE that the message and therefore its value have been issued by a simulation unit. The GOOSE subscriber will report the value of the simulated message to its application instead of the “real” message depending on the setting of the receiving IED. The allowance for an IED to switch from acceptance of real messages to simulated messages is specified by a data object defined in IEC 61850-7-4.

18.2.3.9 ConfRev – configuration revision

The parameter **ConfRev** shall contain the value of the **ConfRev** attribute of the **GOOSE** control block.

18.2.3.10 NdsCom – needs commissioning

The parameter **NdsCom** shall contain the value of the attribute **NdsCom** of the **GoCB**.

18.2.3.11 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, in the order defined by the **data-set**.

The parameter **Value** shall contain the value of a member of the **data-set** referenced in the **GoCB**.

19 Transmission of sampled value class model

19.1 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 DataSetMembers typically belong to the Common Data Class SAV (sampled analogue value), but can belong to any other Common Data Classes as defined in IEC 61850-7-3, if sampled with the specific rate. 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 block (**SVCB**) in the publisher shall be used to control the communication procedure.

Figure 36 gives an overview on the classes and services of the model.

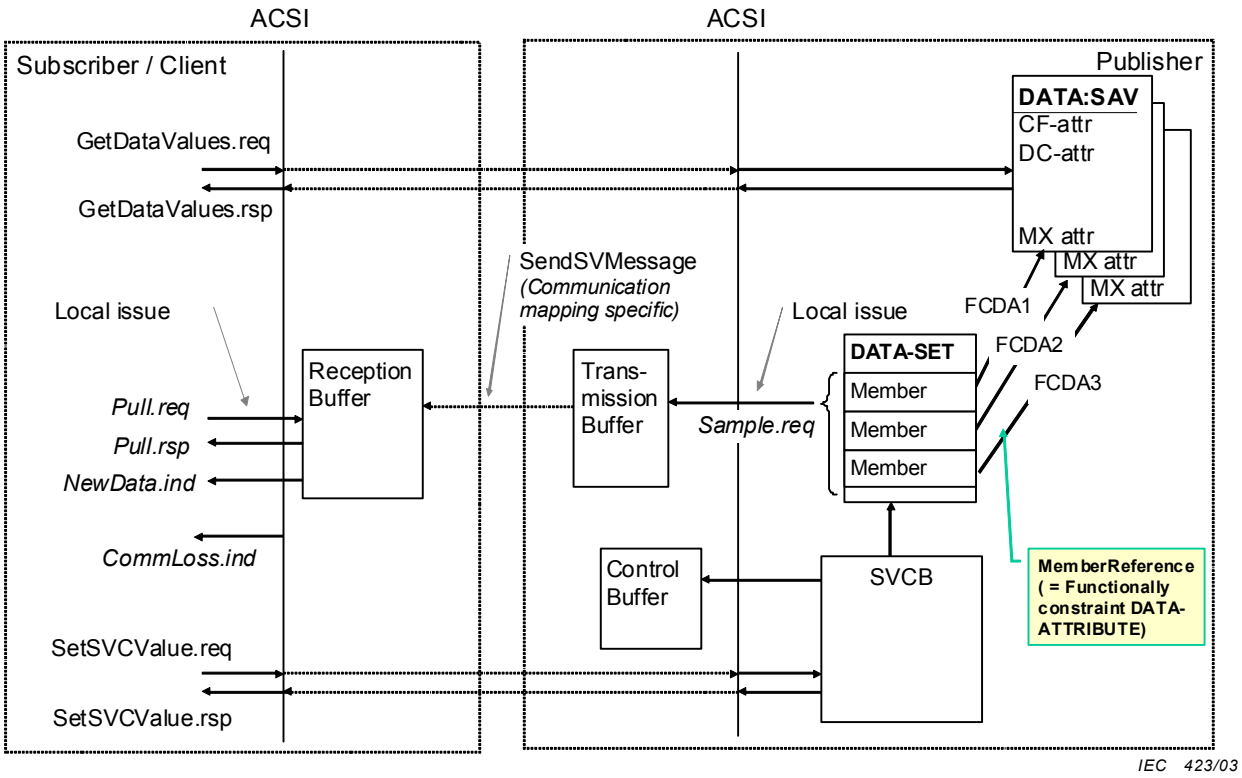


Figure 36 – Model for transmission of sampled values

There shall be two methods to exchange sampled values between a publisher and one or more subscribers. 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 Figure 36 describes the model for transmission of sampled values.

When the **multicast-application-association** applies, the following service mapping occurs:

- **SetSVCValues** represents the **SetMSVCBValues**,
- **SendSVMessage** represents **SendMSVMessage**,
- the type of the **SVCB** is **MSVCB**.

When the **two-party-application-association** applies, the following service mapping occurs:

- **SetSVCValues** represents the **SetUSVCBValues**,
- **SendSVMessage** represents **SendUSVMessage**,
- the type of the **SVCB** is **USVCB**.

The producer shall sample the inputs with the specified sample rate. The synchronization of this sampling may be done internally or over the network. The samples shall be posted in the transmission buffer.

The network embedded scheduler shall send the contents 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 by which 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.

19.2 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.

19.2.1 MSVCB class definition

The **MSVCB** shall be as defined in Table 44.

Table 44 – MSVCB class definition

MSVCB class			
Attribute name	**Attribute type**	**r/w**	**Value/value range/explanation**
MsvCBName	ObjectName	-	Instance name of an instance of **MSVCB**
MsvCBRef	ObjectReference	-	Path-name of an instance of **MSVCB**
SvEna	BOOLEAN	r/w	Enabled (TRUE) | disabled (FALSE), DEFAULT FALSE
MsvID	VISIBLE STRING129	r/w	
DatSet	ObjectReference	r/w	
ConfRev	INT32U	r	
SmpMod	ENUMERATED	r/w	samples per nominal period (DEFAULT) | samples per second | seconds per sample
SmpRate	INT16U	r/w	(0..MAX)
OptFlds	PACKED LIST	r/w	
refresh-time	BOOLEAN		
reserved	BOOLEAN		
sample-rate	BOOLEAN		
data-set-name	BOOLEAN		
DstAddress	PHYCOMADDR	r	
Services SendMSVMessage GetMSVCBValues SetMSVCBValues			

19.2.1.1 MsvCBName – multicast sampled value control name

The attribute **MsvCBName** shall unambiguously identify a **MSVCB** within the scope of an **LLN0**.

19.2.1.2 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

19.2.1.3 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 change of attribute values of the **MSVCB** other than disabling shall be allowed.

The value of SvEna at IED startup is a local issue, depending on the IED configuration.

19.2.1.4 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.

NOTE Depending upon the message definition, for example by the control block option field, it may not be possible to uniquely identify the SV control through the control block 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.

19.2.1.5 DatSet

The attribute **DatSet** shall specify the reference of the **data-set** whose values of members are to be transmitted in the **MSVCB** message.

19.2.1.6 ConfRev – configuration revision

The attribute **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 an attribute (MsvID, DatSet, SmpMod, SmpRate, OptFlds) of **MSVCB** (multicast sampled value control block).

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.

19.2.1.7 SmpMod

The attribute **SmpMod** shall specify if the sample rate is defined in units of samples per nominal period, samples per second or seconds per sample. If it is missing, the default value is samples per period.

19.2.1.8 SmpRate

The attribute **SmpRate** shall specify the sample rate. The value shall be interpreted depending on the value of the **SmpMod**.

19.2.1.9 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 **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);

– **SmpRate** (sample rate and sample mode from the instance of MSVCB);

– **DatSet** (data set name).

19.2.1.10 DstAddress

The attribute **DstAddress** shall be the SCSM specific addressing information like media access address, priority, and other information.

19.2.2 Multicast sampled value class services

19.2.2.1 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**

19.2.2.2 SendMSVMessage

19.2.2.2.1 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

19.2.2.2.2 Request

19.2.2.2.2.1 MSV message

The parameter **MSV message** shall specify the values of the members of the referenced **data-set** of the **MSVCB** as specified in the abstract sampled value format definition (see 19.4). The concrete format of the **MSV message** shall be defined in the SCSM.

19.2.2.3 GetMSVCBValues

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

Parameter name
Request
MsvCBReference
Response+
SvEnable
MulticastSampleValueID
DataSetReference
ConfigurationRevision
SampleMode [0..1]
SampleRate
OptionalFields
DestinationAddress [0..1]
Response–
ServiceError

19.2.2.3.1 Request

19.2.2.3.1.1 MsvCBReference

The parameter **MsvCBReference** shall specify the **ObjectReference** of the **MSVCB**.

The service parameter **MsvCBReference** shall be **LDName/LLN0.MsvCBName**.

19.2.2.3.2 Response+

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

19.2.2.3.2.1 SvEnable

The parameter **SvEnable** shall contain the value of the corresponding attribute **SvEna** of the referenced **MSVCB**.

19.2.2.3.2.2 MulticastSampleValueID

The parameter **MulticastSampleValueID** shall contain the value of the corresponding attribute **MsvID** of the referenced **MSVCB**.

19.2.2.3.2.3 DataSetReference

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

19.2.2.3.2.4 ConfigurationRevision

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

19.2.2.3.2.5 SampleMode [0..1]

The parameter **SampleMode** shall contain the value of the corresponding attribute **SmpMod** of the **MSVCB**. In case it is missing, the default value sample per period applies.

19.2.2.3.2.6 SampleRate

The parameter **SampleRate** shall contain the value of the corresponding attribute **SmpRate** of the **MSVCB**.

19.2.2.3.2.7 OptionalFields

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

19.2.2.3.2.8 DestinationAddress [0..1]

The parameter **DestinationAddress** shall contain the value of the corresponding attribute **DstAddress** of the **MSVCB**.

19.2.2.3.3 Response–

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

19.2.2.4 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**.

Parameter name
Request
MsvCBReference
SvEnable [0..1]
MulticastSampleValueID [0..1]
DataSetReference [0..1]
SampleMode [0..1]
SampleRate [0..1]
OptionalFieds [0..1]
Response+
Response–
ServiceError

19.2.2.4.1 Request

19.2.2.4.1.1 MsvCBReference

The parameter **MsvCBReference** shall specify the **ObjectReference** of the **MSVCB**.

The service parameter **MsvCBReference** shall be **LDName/LLN0.MsvCBName**.

19.2.2.4.1.2 SvEnable [0..1]

The parameter **SvEnable** shall contain the value for the corresponding attribute **SvEna** of the referenced **MSVCB**.

19.2.2.4.1.3 MulticastSampleValueID [0..1]

The parameter **MulticastSampleValueID** shall contain the value for the corresponding attribute **MsvID** of the referenced **MSVCB**.

19.2.2.4.1.4 DataSetReference [0..1]

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

19.2.2.4.1.5 SampleMode [0..1]

The parameter **SampleMode** shall contain the value of the corresponding attribute **SmpMod** of the **MSVCB**.

19.2.2.4.1.6 SampleRate [0..1]

The parameter **SampleRate** shall contain the value for the corresponding attribute **SmpRate** of the **MSVCB**.

19.2.2.4.1.7 OptionalFields [0..1]

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

19.2.2.4.2 Response+

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

19.2.2.4.3 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.

19.3 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 instance of the control block 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 block class shall be released.

The samples shall be sent using the two-party application association.

19.3.1 USVCB class definition

The **USVCB** shall be as defined in Table 45.

Table 45 – USVCB class definition

USVCB class			
Attribute name	**Attribute type**	**r/w**	**Value/value range/explanation**
UsvCBName	ObjectName	-	Instance name of an instance of UNICAST-SVC
UsvCBRef	ObjectReference	-	Path-name of an instance of UNIICAST-SVC
SvEna	BOOLEAN	r/w	Enabled (TRUE) | disabled (FALSE), DEFAULT FALSE
Resv	BOOLEAN	r/w	
UsvID	VISIBLE STRING129	r/w	
DatSet	ObjectReference	r/w	
ConfRev	INT32U	r	
SmpMod	ENUMERATED	r/w	samples per nominal period (DEFAULT) | samples per second | seconds per sample
SmpRate	INT16U	r/w	(0..MAX)
OptFlds	PACKED LIST	r/w	
refresh-time	BOOLEAN		
reserved	BOOLEAN		
sample-rate	BOOLEAN		
data-set-name	BOOLEAN		
DstAddress	PHYCOMADDR	r	
Services SendUSVMessage GetUSVCBValues SetUSVCBValues			

19.3.1.1 UsvCBName – unicast sampled value control name

The attribute **UsvCBName** shall unambiguously identify a **USVCB** within the scope of a **LLN0**.

19.3.1.2 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

19.3.1.3 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 change 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.

19.3.1.4 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**.

19.3.1.5 UsvID

The attribute **UsvID** shall be a unique identification of the sampled source.

19.3.1.6 DatSet

The attribute **DatSet** shall specify the reference of the **data-set** whose values of members are to be transmitted in the **USVCB** message.

19.3.1.7 ConfRev – configuration revision

The attribute **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 an attribute (UsvID, DatSet, SmpMod, SmpRate, OptFlds) 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.

19.3.1.8 SmpMod

The attribute **SmpMod** shall specify if the sample rate is defined in units of samples per nominal period, samples per second or seconds per sample. If it is missing, the default value is samples per period.

19.3.1.9 SmpRate

The attribute **SmpRate** shall specify the sample rate. The value shall be interpreted depending on the value of the **SmpMod**.

19.3.1.10 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),

– **SmpRate** (sample rate and sample mode from the instance of **USVCB**),

– **DatSet** (data set name).

19.3.1.11 DstAddress

The attribute **DstAddress** shall be the SCSM specific addressing information like address, priority, and other information.

19.3.2 Unicast sampled value services

19.3.2.1 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**

19.3.2.2 SendUSVMessage

19.3.2.2.1 SendUSVMessage parameter table

The **SendUSVCMessage** service shall be used by a **USVCB** to send sampled values from the server to the client over a **two-party-application-association**.

Parameter name
Request
USV message

19.3.2.2.2 Request

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 19.4). The concrete format of the **USV message** shall be defined in the SCSM.

19.3.2.3 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**.

Parameter name
Request
UsvCBReference
Response+
SvEnable
CBReserved
UnicastSampleValueID
DataSetReference
ConfigurationRevision
SampleMode [0..1]
SampleRate
OptionalFields [0..1]
DestinationAddress[0..1]
Response–
ServiceError

19.3.2.3.1 Request

The parameter **UsvCBReference** shall specify the **ObjectReference** of the **USVCB**.

The service parameter **UsvCBReference** shall be **LDName/LLN0.UsvCBName**.

19.3.2.3.2 Response+

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

19.3.2.3.2.1 SvEnable

The parameter **SvEnable** shall contain the value of the corresponding attribute **SvEna** of the referenced **USVCB**.

19.3.2.3.2.2 CBReserved

The parameter **CBReserved** shall contain the value of the corresponding attribute **Resv** of the referenced **USVCB**.

19.3.2.3.2.3 UnicastSampleValueID

The parameter **UnicastSampleValueID** shall contain the value of the corresponding attribute **UsvID** of the referenced **USVCB**.

19.3.2.3.2.4 DataSetReference

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

19.3.2.3.2.5 ConfigurationRevision

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

19.3.2.3.2.6 SampleMode [0..1]

The parameter **SampleMode** shall contain the value of the corresponding attribute **SmpMod** of the **USVCB**. If missing, the default value sample per period applies.

19.3.2.3.2.7 SampleRate

The parameter **SampleRate** shall contain the value of the corresponding attribute **SmpRate** of the **USVCB**.

19.3.2.3.2.8 OptionalFieds [0..1]

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

19.3.2.3.2.9 DestinationAddress [0..1]

The parameter **DestinationAddress** shall contain the value of the corresponding attribute **DstAddress** of the **USVCB**.

19.3.2.3.3 Response–

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

19.3.2.4 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**.

Parameter name
Request
UsvCBReference
SvEnable [0..1]
CBReserved [0..1]
UnicastSampleValueID [0..1]
DataSetReference [0..1]
SampleMode [0..1]
SampleRate [0..1]
OptionalFields [0..1]
Response+
Response–
ServiceError

19.3.2.4.1 Request

19.3.2.4.1.1 UsvCBReference

The parameter **UsvCBReference** shall specify the **ObjectReference** of the **USVCB**.

The service parameter **UsvCBReference** shall be **LDName/LLN0.UsvCBName**.

19.3.2.4.1.2 SvEnable [0..1]

The parameter **SvEnable** shall contain the value for the corresponding attribute **SvEna** of the referenced **USVCB**.

19.3.2.4.1.3 CBReserved [0..1]

The parameter **CBReserved** shall contain the value for the corresponding attribute **Resv** of the referenced **USVCB**.

19.3.2.4.1.4 UnicastSampleValueID [0..1]

The parameter **UnicastSampleValueID** shall contain the value for the corresponding attribute **UsvID** of the referenced **USVCB**.

19.3.2.4.1.5 DataSetReference [0..1]

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

19.3.2.4.1.6 SampleMode [0..1]

The parameter **SampleMode** shall contain the value of the corresponding attribute **SmpMod** of the **USVCB**.

19.3.2.4.1.7 SampleRate [0..1]

The parameter **SampleRate** shall contain the value for the corresponding attribute **SmpRate** of the **USVCB**.

19.3.2.4.1.8 OptionalFieds [0..1]

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

19.3.2.4.2 Response+

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

19.3.2.4.3 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.

19.4 Sampled value format

The abstract sampled value format used for the sampled value message shall be as defined in Table 46.

Table 46 – Sampled value (SV) format definition

Sampled value format		
Parameter name	**Parameter type**	**Value/value range/explanation**
MsvID or **UsvID**	VISIBLE STRING129	Value from the **MSVCB** or **USVCB**
OptFlds	[a]	Optional fields to be included in the SV message
DatSet	ObjectReference	OPTIONAL; value from the **MSVCB** or **USVCB**
Sample [1..n]		
Value	(*)	(*) The value of the member of the instance of the **DATA-SET** referenced by the **MSVCB or USVCB**. The type of the value typically belongs to the common data class **SAV** (sampled analogue value), but can be any other CDC's process values as defined in IEC 61850-7-3 if sampled with the specified rate.
SmpCnt	INT16U	Sample counter
RefrTm	TimeStamp	OPTIONAL; time of refresh activity
ConfRev	INT32U	Configuration revision number from the instance of **MSVCB** or **USVCB**
SmpSynch	INT8U	Samples are synchronized by clock signals
SmpRate	INT16U	OPTIONAL; sample rate from the instance of **MSVCB** or **USVCB**
SmpMod	ENUMERATED	OPTIONAL; sample mode from the instance of **MSVCB** or **USVCB;** if only SmpRate is sent, this implies SmpMod is in samples per period
Simulation	BOOLEAN	TRUE (simulation or test values) | FALSE (operational values)

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

19.4.1 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.

19.4.2 OptFlds

The parameter **OptFlds** shall specify which of the optional fields (**RefrTm**, **SmpRate and SmpMod**, and **DatSet**) are included in the sampled value message. If for example the attribute refresh-time of the attribute OptFlds of the sampled value control block is TRUE, then the field **RefrTm** shall be contained in the sampled value message.

The parameter **OptFlds** shall be derived from the attribute **OptFlds** of the respective **USVCB** or **MSVCB**.

19.4.3 DatSet

The parameter **DatSet** (taken from the **MSVCB** or **USVCB**) shall contain the **ObjectReference** of the **data-set** whose values of the members are transmitted in the message.

19.4.4 Sample [1..n]

The parameter **Sample** shall contain the value of a member of **data-set** sampled at a given time.

19.4.5 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 IEC 61850; details can be found in a SCSM.

19.4.6 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 value of the data object.

19.4.7 ConfRev

The parameter **ConfRev** shall contain the value of the attribute **ConfRev** of the **MSVCB** or **USVCB**.

19.4.8 SmpSynch

The parameter **SmpSynch** shall indicate whether the sampled values sent by the **MSVCB** or **USVCB** are synchronized by clock signals. The following values shall be used:

2 = indicates that the SV are synchronized by a global area clock signal;
1 = indicates that the SV are synchronized by a local area clock signal;
0 = indicates that the SV are not synchronized by an external clock signal.

19.4.9 SmpRate

The parameter **SmpRate** shall contain the value of the attribute **SmpRate** of the **MSVCB** or **USVCB**.

19.4.10 SmpMod

The parameter **SmpMod** shall contain the value of the attribute **SmpMod** of the **MSVCB** or **USVCB**.

19.4.11 Simulation

The parameter **Simulation** shall indicate with the value TRUE that the message and therefore its value have been issued by a simulation unit. The SMV subscriber reports the values of the simulated message to its application instead of the “real” message depending on the setting of the receiving IED. The setting of the simulation property of an IED is specified by a data object defined in IEC 61850-7-4.

20 CONTROL class model

20.1 Introduction

The control model provides a specific way to change the state of internal and external processes by a client. The control model can only be applied to data object instances of a Controllable common data class (CDC) and whose ctlModel **DataAttribute** is not set to “status-only”. Such data objects will be referred to as "**control objects**".

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) / TimeActivatedOperateTermination (TimOperTermination)**;

– **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 37.

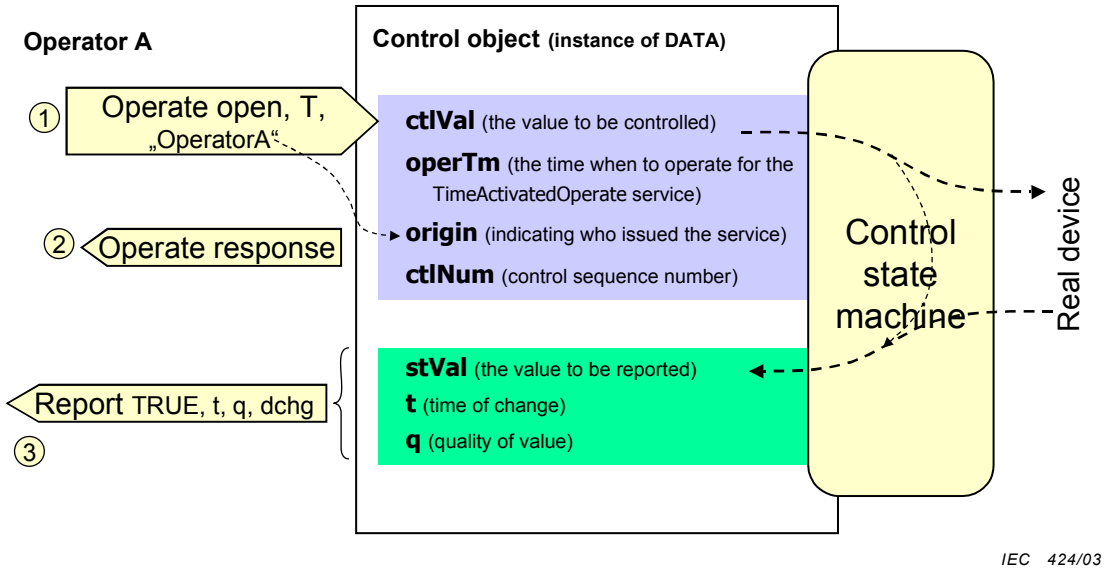


Figure 37 – Principle of the control model

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**, **TimeActivatedOperateTermination**, 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 by configuration **dataAttributes** such as ctlModel, sboTimeout, sboClass, or operTimeout (see IEC 61850-7-3 for the semantic definition of the configuration **dataAttributes**). 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 for a dynamic test, and interlock for an operative test). Besides the check condition specified by the service parameter, the control object shall perform additional checks depending on the controlled object, such as checking Logical Node Behavior, the command blocked condition, remote/local condition, etc. The proper control response shall be generated according to the used model (for example, direct control with normal security). The checks mentioned above, which are dependent on the operative condition of the object and its process environment, are called ‘operative test’, and are performed e.g. in the state PerformTest. The synchrocheck in contrast is a ‘dynamic test’, i.e. a test performed after acceptance of the Operate command e.g. in the state WaitForExecution, which means to check the moment of allowance for the command to the object, which might depend on some dynamic condition like the correct phase angle before closing a circuit breaker.

The behaviour for the different control models is described in the following subclauses with state diagrams. These state diagrams however show only those conditions which lead to new states. Table 47 contains the handling of control model related service requests, which do not change the state and are negatively responded. Table 47 only takes care of defining the behavior of the control model when application errors occur.

Table 47 – Generic behavior and negative responses

State	Service request	Action (AddCause see Table 54)	Comments
Unselected	Oper_req	Oper_resp- (object-not-selected)	
Unselected	Cancel_req	Cancel_resp+	Nothing selected – cancel goal already reached
Ready	SelVal_req or Sel_req	SelVal_resp- (already-selected) or Sel_resp- (already-selected)	For direct control the AddCause is class-not-supported
Ready	Cancel_req	Cancel_resp-(inconsistent-parameters)	For SBO control with normal and enhanced security control model: discrepancy between Select service parameters and Cancel service parameters.
Ready	Cancel_req	Cancel_resp- (not-supported)	For direct control
Ready	Oper_req or Cancel_req	Oper-resp-(locked-by-other-client) or Cancel_resp-(locked-by-other-client)	The second service is not performed by the client which performed the positively acknowledged Sel_req resp. SelVal_req
PerformTest	SelVal_req	SelVal_resp- (command-already-in-execution)	
PerformTest	Oper_req	Oper_resp- (command-already-in-execution)	
PerformTest	Cancel_req	Cancel_resp-(inconsistent-parameters)	Discrepancy between Select resp. Oper service parameters and Cancel

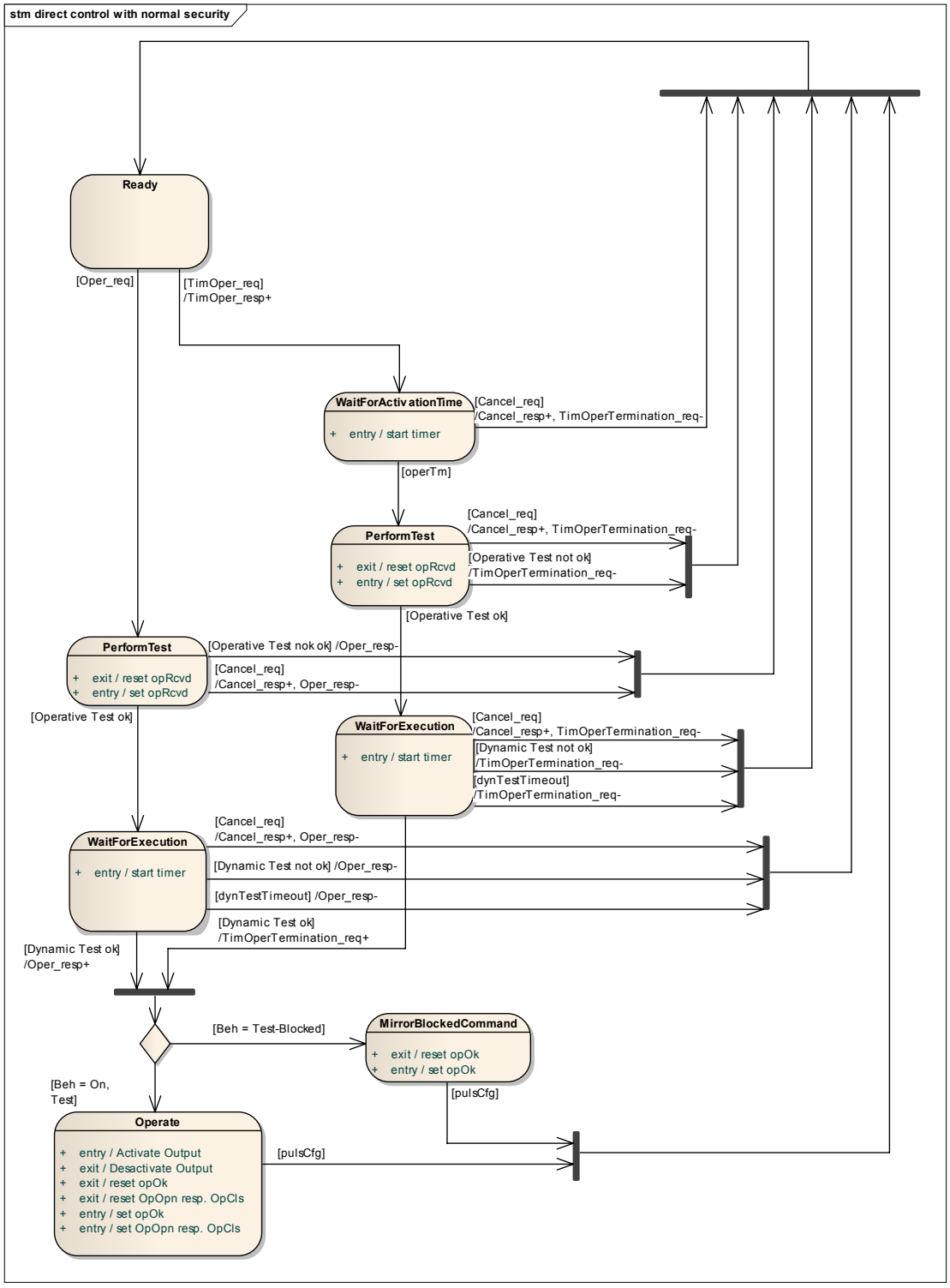
State	Service request	Action (AddCause see Table 54)	Comments
			service parameters
PerformTest	Oper_req or Cancel_req	Oper-resp-(locked-by-other-client) or Cancel_resp-(locked-by-other-client)	The second service is not performed by the client which performed the positively acknowledged Sel_req resp. SelVal_req
WaitForExecution	SelVal_req or Sel_req	SelVal_resp- (command-already-in-execution) or Sel_resp- (command-already-in-execution)	
WaitForExecution	Oper_req	Oper_resp- (command-already-in-execution)	
WaitForExecution	Cancel_req	Cancel_resp-(inconsistent-parameters)	Discrepancy between Oper service parameters and Cancel service parameters
WaitForExecution	Cancel_req	Cancel_resp-(locked-by-other-client)	The Cancel_req is not performed by the client which performed the positively acknowledged Sel_req resp. SelVal_req
WaitForChange	SelVal_req	SelVal_resp- (command-already -in-execution)	
WaitForChange	Oper_req	Oper_resp- (command-already -in-execution)	
WaitForChange	Cancel_req	Cancel_resp- (command-already -in-execution)	
WaitForActivationTime	SelVal_req	SelVal_resp- (object-already_selected)	
WaitForActivationTime	TimOper_req	TimOper_resp+	Use new time
WaitForActivationTime	Cancel_req	Cancel_resp-(inconsistent-parameters)	Discrepancy between Oper service parameters and Cancel service parameters
WaitForActivationTime	Cancel_req	Cancel_resp-(locked-by-other-client)	The Cancel_req service is not performed by the client which performed the positively acknowledged Sel_req resp. SelVal_req
Operate	Oper_req	Oper_resp-(command-already-in-execution)	
Operate	Sel_req	Sel_resp- (command-already-in-execution)	For direct control, the AddCause is class-not-supported
Operate	Cancel_req	Cancel_resp- (command-already-in-execution)	
All states	Control related request, not belonging to control model	xxx-resp- (unknown) NOTE The SCSM mapping can define here service errors instead.	For example a Sel_req for ctlModel = direct or ctlModel = SBO-with-enhanced-security

20.2 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, for example if the status value did not change to the demanded value, the client will not get information about the failure from the control object.

20.2.1 Direct control with normal security

This model shall use the services **Operate, TimeActivatedOperate** and **Cancel.** The state machine in Figure 38 and the sequence diagram in Figure 39 define the operation.



IEC 425/03

Figure 38 – State machine of direct control with normal security

Direct control with normal security should be used for operations that act either on a local data object (for example, a LED test) or on a data object that influences external devices where returned information is not supervised and independent from any other object (for example, switch on a heating).

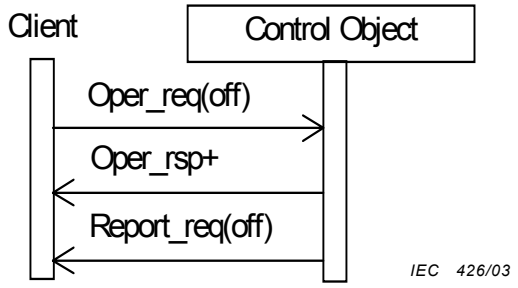


Figure 39 – Direct control with normal security

The procedure is as follows:

On receipt of an **Operate** request, the control object shall determine if the client has appropriate access authority and shall check validation of the control execution. The check validation is performed in the states **PerformTest** where Operative Tests are applied and **WaitForExecution** where Dynamic Tests are applied.

- 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 according to the **DataObject Beh** associated to the parent **logical node**, for example by changing a state value, activating a binary output, sending an equivalent signal on a process bus, or mirroring the received command without executing it.

The new status may optionally be reported by the **Report** service (see reporting model).

20.2.2 SBO control with normal security

This model shall use the services **Select**, **Cancel**, **Operate**, and **TimeActivatedOperate**. Once not in the unselected state, only the client that selected the control object shall be able to force state transitions for the control object. All control requests for the selected control object from other clients shall generate a negative response. The state machine in Figure 40 defines the operation.

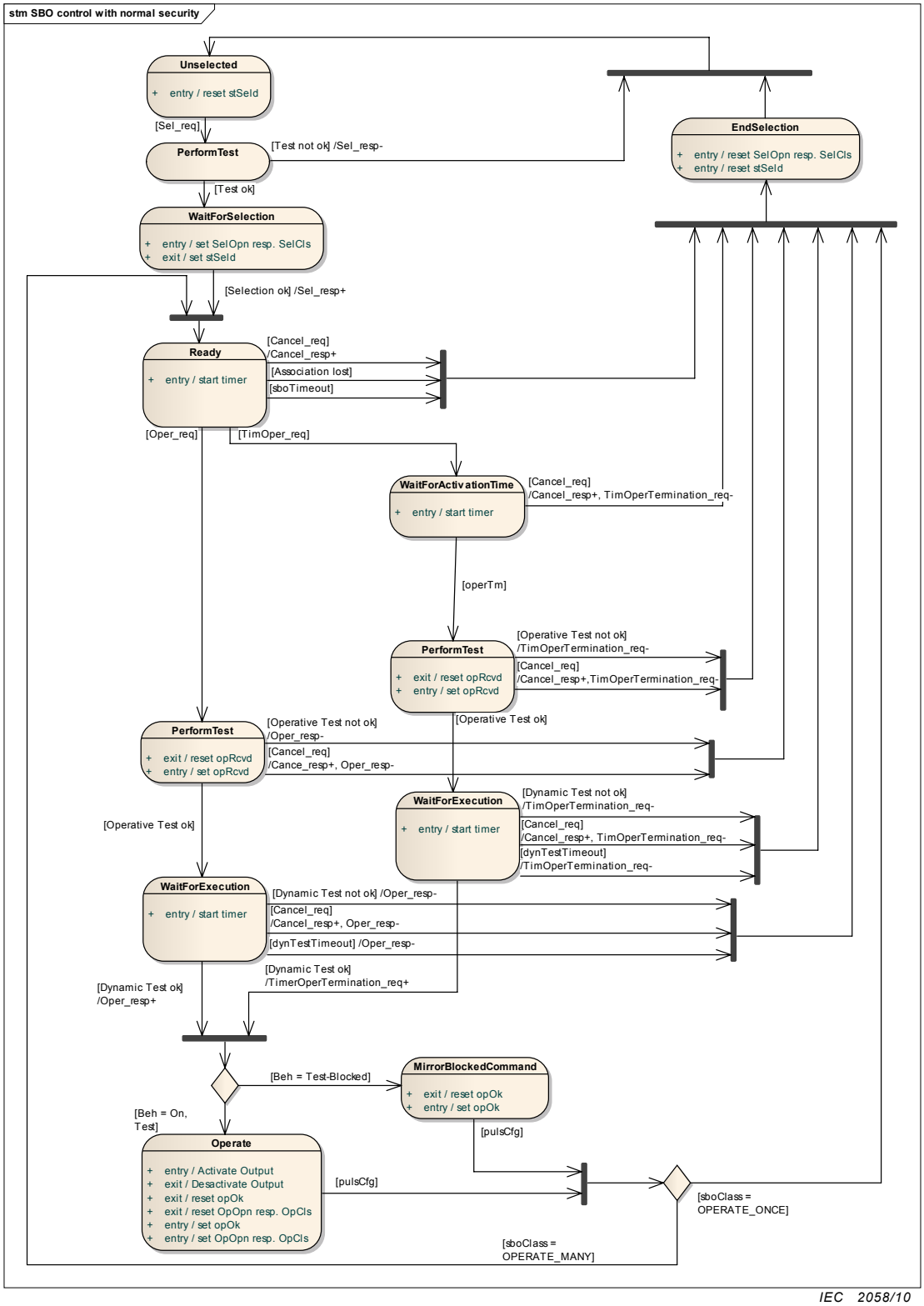


Figure 40 – State machine of SBO control with normal security

The procedure is as follows.

a) On receipt of a **Select** request, the control object shall determine if the client has appropriate access authority, shall check 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 sboTimeout 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 unselected 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. The check validation is performed in the states **PerformTest** where Operative Tests are aplied and **WaitForExecution** where Dynamic Tests are applied.
   - 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 according to the **DataObject Beh** associated to the parent **logical node**, for example by changing a state value, activating a binary output, sending an equivalent signal on a process bus or mirroring the received command without executing it.

20.3 Control with enhanced security

20.3.1 Introduction

In 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.

20.3.2 Direct control with enhanced security

This model shall use the services **Operate**, **Cancel**, **TimeActivatedOperate**, and **CommandTermination**. The state machine in Figure 41 defines the operation.

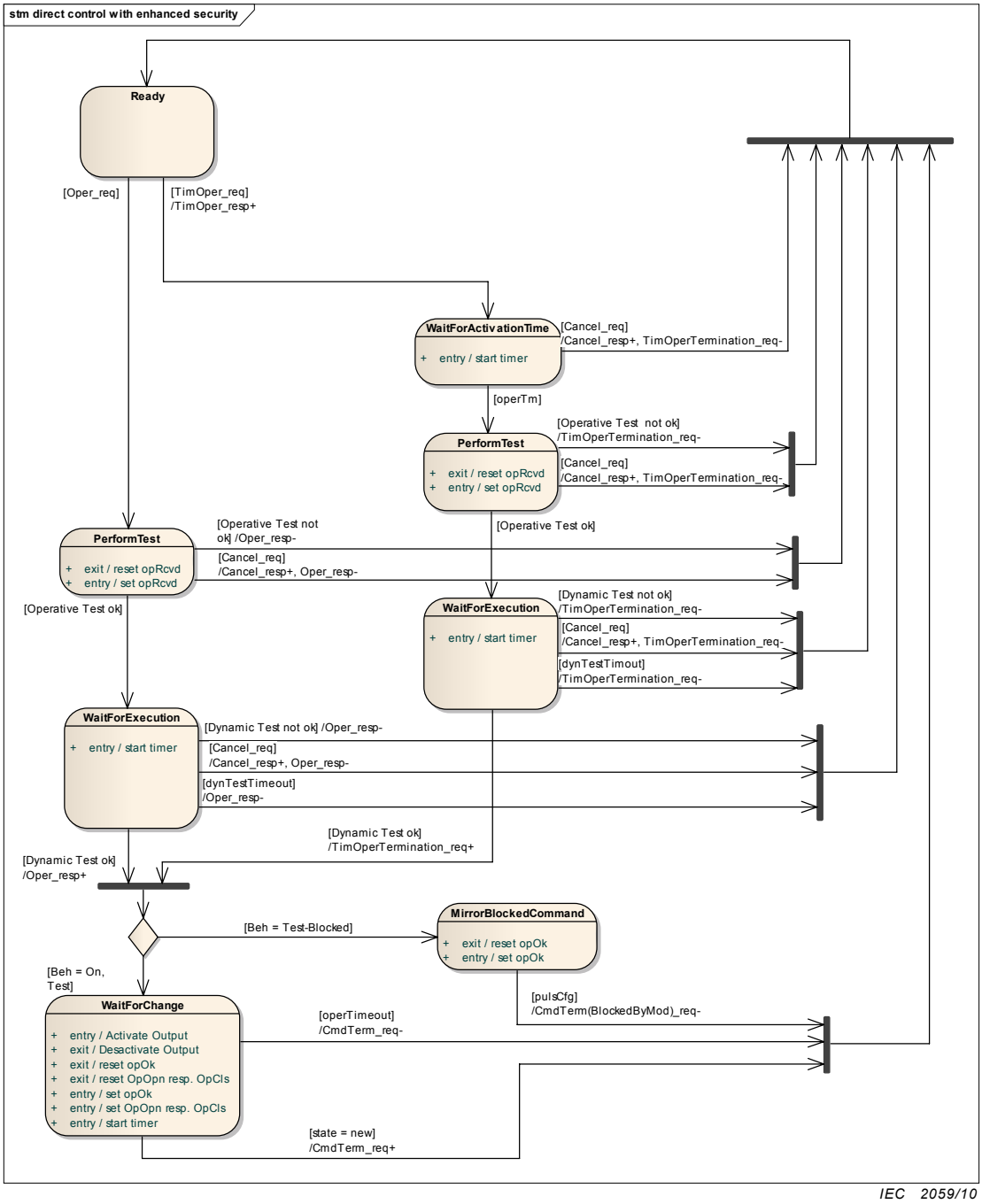


Figure 41 – State machine of direct control with enhanced security

NOTE As a result of the state change, the status dataAttributes of the control object may change and this change may be for example reported. In the following sequence diagrams, such a report is shown.

20.3.3 SBO control with enhanced security

This model uses the services **SelectWithValue**, **Cancel**, **Operate**, **TimeActivatedOperate**, and **CommandTermination**. Once not in the unselected state, only the client that selected the control object shall be able to force state transitions for the control object. All control requests, for the selected control object from other clients shall generate a negative response. The state machine in Figure 42 and the sequence diagrams in Figure 43 and Figure 44 define the operation.

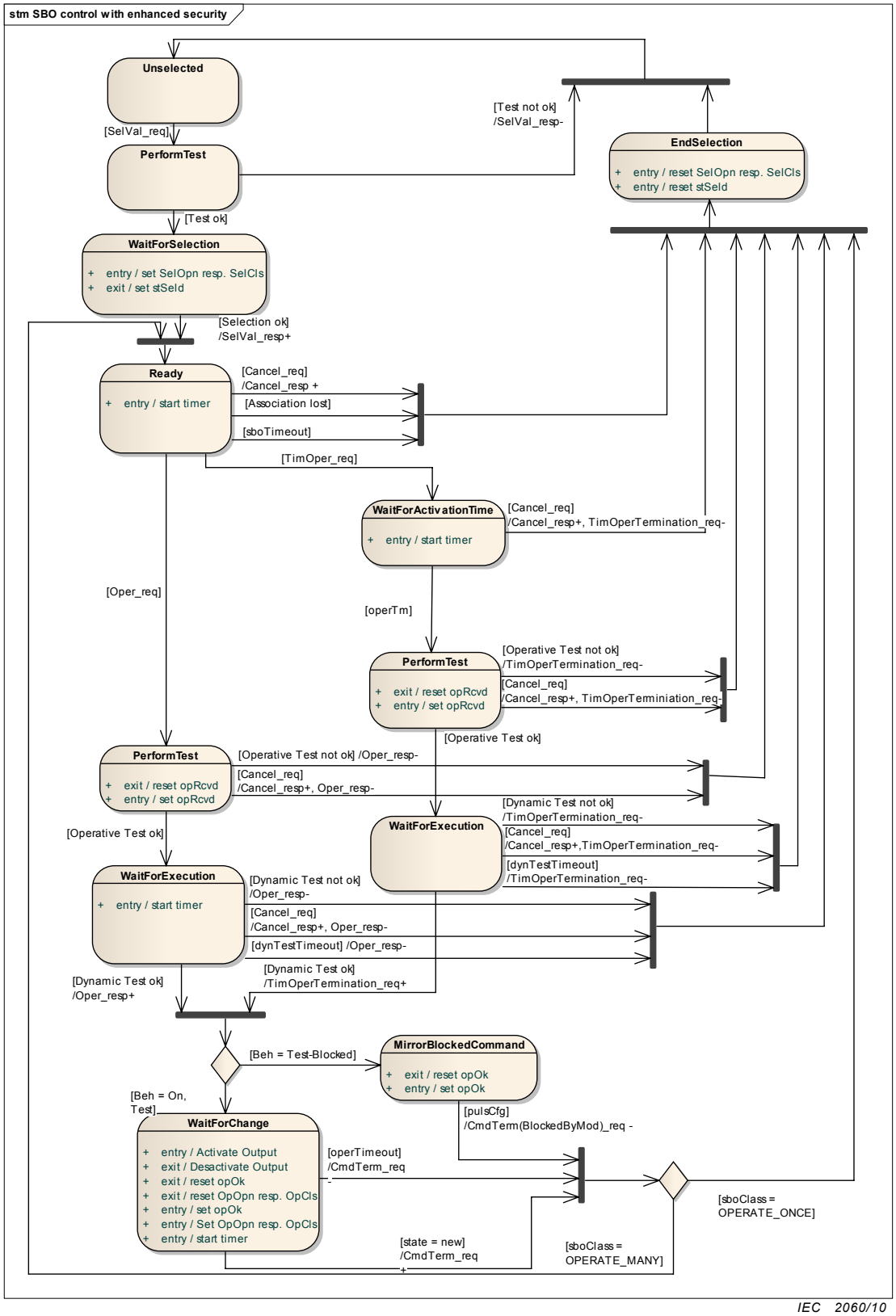


Figure 42 – State machine SBO control with enhanced security

NOTE As a result of the state change, the status attribute of the control object may change and this change may be for example reported. In the following sequence diagrams, such a report is shown.

Control with enhanced security should be used for control procedures that cause an important action outside the device containing the accessed control object.

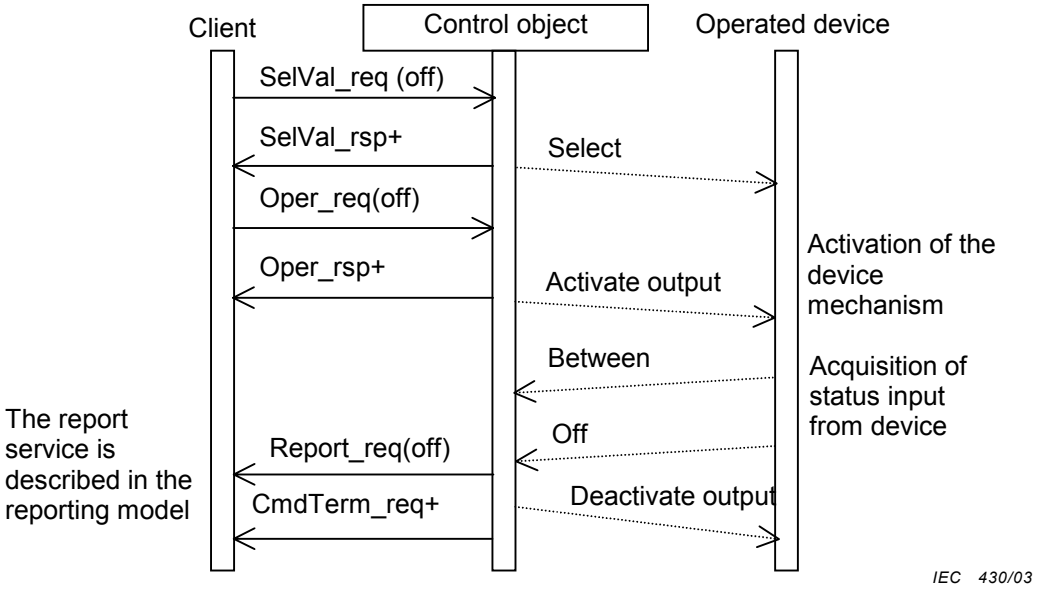


Figure 43 – Select before operate with enhanced security – positive case

NOTE The dashed lines in Figure 43 and Figure 44 indicate that these "services" are local and not visible at the communication level.

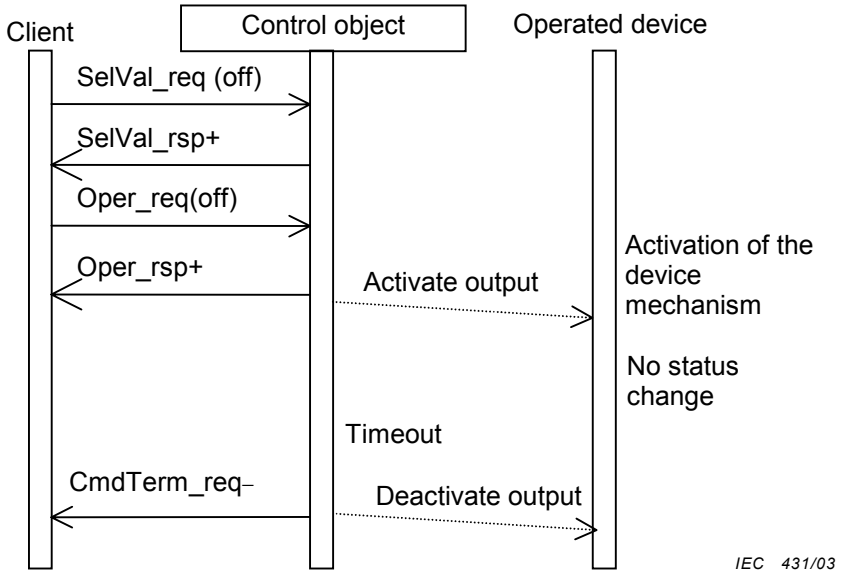


Figure 44 – Select before operate with enhanced security – negative case (no status change)

The procedure is as follows:

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 **sboTimeout** 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. The check validation is performed in the states **PerformTest** where Operative Tests are aplied and **WaitForExecution** where Dynamic Tests are applied.

   – 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 according to the **DataObject Beh** associated to the parent **logical node** e.g by activating a binary output, sending an equivalent signal on a process bus, or mirroring the received command without executing it. 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 defined by the attribute operTimeout, 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 issue 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.

20.4 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**. The sequence diagram in Figure 45 defines the operation.

NOTE The example below is shown with the control class direct-operate with enhanced security. 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.

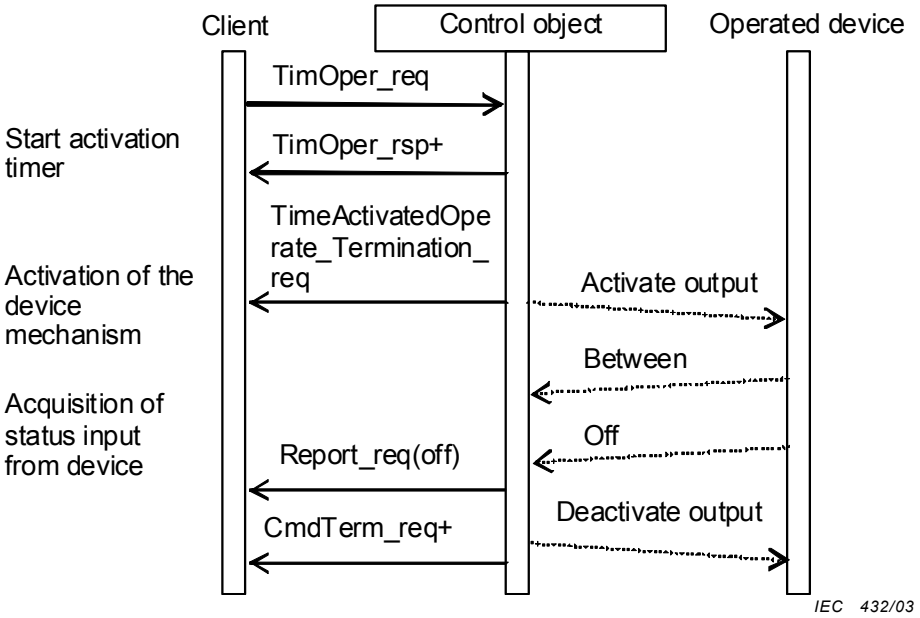


Figure 45 – Time-activated operate

The procedure is as follows:

a) On receipt of a **TimeActivatedOperate** request, the control object shall determine if the client has appropriate access authority, and 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 **TimeActivateOperateTermination** request shall be sent to the client.

c) All further information exchange shall be as described in the model for control with enhanced security.

20.5 CONTROL class service definitions

20.5.1 Overview

For **control**, the following services listed in Table 48 are defined.

Table 48 – Control services

ACSI control service
Select (Sel)
SelectWithValue (SelVal)
Cancel (Cancel)
Operate (Oper)
CommandTermination (CmdTerm)
TimeActivatedOperate (TimOper)
TimeActivatedOperateTermination

20.5.2 Service parameter definition

The following service parameters shall be applied in the service definitions.

20.5.2.1 ControlObjectReference

The parameter **ControlObjectReference** shall contain the ObjectReference of the controllable data object (for example defined in IEC 61850-7-4) to be accessed, for example **Pos**, which represents the data object "**Position**".

20.5.2.2 ctlVal

The service parameter **ctlVal** determines the control activity. Its TypeDefinition depends on the common data class used by the control object. The controllable common data classes (CDCs) define the TypeDefinition of the service parameter ctlVal.

All service primitives belonging to one control sequence shall carry the same control value.

20.5.2.3 origin

The service parameter **origin** determines the originator of the control service. **origin** is a ConstructedAttribute, whose TypeDefinition shall be identical to the functional constrained ST respectively MX DataAttribute **origin** defined in the controllable common data class of IEC 61850-7-3 used by the control object.

All service primitives belonging to one control sequence shall be identified by the same origin.

20.5.2.4 ctlNum

The service parameter **ctlNum** determines the control sequence number of the control service. Its TypeDefinition shall be identical to the functional constrained ST respectively MX DataAttribute **ctlNum** defined in the controllable common data class of IEC 61850-7-3 used by the control object.

All service primitives belonging to one control sequence shall be identified by the same control sequence number. The use of ctlNum is an issue of the client.

20.5.2.5 T – control time-stamp

The service parameter **T** (see Table 49) shall be the control time-stamp. It shall contain the time when the client issues the service control request.

NOTE Control requests can be Select, Operate, Cancel, TimeActivatedOperate.

Table 49 – T definition

T type		
Service parameter name	**Parameter type**	**Value/value range/explanation**
T	TimeStamp	

Observe that the command service tracking allows logging of this timestamp, for example to compare the time a client issues a command with the time the command has been received or executed.

20.5.2.6 Test – test status

The service parameter **Test** (see Table 50) shall be the test status of the control service. It shall determine if the client sends a control service for test purpose or not.

All service primitives belonging to one control sequence shall carry the same test status.

Table 50 – Test definition

Test status type		
Service parameter name	**Parameter type**	**Value/value range/explanation**
Test	BOOLEAN	no-test (FALSE) | test (TRUE)

NOTE The fact that the control service is executed depends on the value of the data object Beh (defined in IEC 61850-7-4) of the **LOGICAL-NODE** that contains the control object.

20.5.2.7 Check – Check condition

The service parameter **Check** (see Table 51) 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). In case that the addressed function respective data object does not support these checks, the appropriate check bits shall be ignored and the command be handled.

Table 51 – Check condition definition

Check condition type		
Service parameter name	**Parameter type**	**Value/value range/explanation**
Check	PACKET LIST	
synchrocheck	BOOLEAN	TRUE means perform synchrocheck
Interlock-check	BOOLEAN	TRUE means check for interlocking condition

20.5.2.8 operTm – Operate time

The service parameter **operTm** (see Table 52) shall specify, when using a **TimeActivatedOperate** service, the absolute time when the command shall be executed. The service parameter shall be present only if the **TimeActivatedOperate** service is supported by the control object.

Table 52 – operTm definition

operTm type		
Service parameter name	**Parameter type**	**Value/value range/explanation**
operTm	TimeStamp	

20.5.2.9 AddCause – additional cause diagnosis

The parameter **AddCause** (see Table 53) shall identify the reason of failure in a negative control service specific response. “None" indicates that no failure occurs.

Table 53 – Additional cause diagnosis definition

Additional cause diagnosis type		
Attribute name	**Attribute type**	**value/value range/explanation**
AddCause	ENUMERATED	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 Object-not-selected Object-already-selected No-access-authority Ended-with-overshoot Abortion-due-to-deviation Abortion-by-communication-loss Unknown Blocked-by-command None Inconsistent-parameters Locked-by-other-client

The description of the values shall be as defined in Table 54.

NOTE Most values of the AddCause are determined by data objects of specific logical nodes defined in IEC 61850-7-4 and other documents.

Table 54 – AddCause semantic

Value	Explanation
Blocked-by-switching-hierarchy	Not successful since one of the downstream **Loc** switches like in **CSWI** has the value TRUE
Select-failed	Canceled 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 doesn't allow any switching.
Blocked-by-process	Control action is blocked due to some external event 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 exceed of **time limit** and missing synchronism condition.
Command-already-in-execution	Control, select or cancel service 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** of that domain, 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** with **ACT.general="TRUE**").
Object-not-selected	Control action is rejected, because control object was not selected
Object-already-selected	Select action is not executed, because the addressed object is already selected.
No-access-authority	Control action is blocked due to lack of access authority
Ended-with-overshoot	Control action executed but the end position has overshoot
Abortion-due-to-deviation	Control action is aborted due to deviation between the command value and the measured value.
Abortion-by-communication-loss	Control action is aborted due to the loss of connection with the client that issued the control.
Unknown	Command not successful due to Unknown causes
Blocked-by-command	Control action is blocked due to the data attribute CmdBlk.stVal is TRUE.
None	Control action successfully executed
Inconsistent-parameters	The parameters between successive control services are not consistent, for example the ctlNum of Select and Operate service are different.
Locked-by-other-client	Another client has already reserved the object.

20.5.3 Service specification

20.5.3.1 General

The control services use the service parameters defined in 20.5.2 to communicate the specific values of the service. All service parameters of a specific service need to be included in one service request.

NOTE 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.

20.5.3.2 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 20.5.2.

20.5.3.3 SelectWithValue (SelVal)

The **SelectWithValue** service shall define the following service parameters.

Parameter name
Request
ControlObjectReference
ctlVal
operTm[0..1]
orgin
ctlNum
T
Test
Check
Response+
ControlObjectReference
ctlVal
operTm[0..1]
orgin
ctlNum
T
Test
Check
Response–
ControlObjectReference
ctlVal
operTm[0..1]
orgin
ctlNum
T
Test
Check
AddCause

The service parameter operTm shall be present if the control object supports **TimeActivatedOperate** services.

NOTE The service parameters are defined in 20.5.2.

20.5.3.4 Cancel (Cancel)

The **Cancel** service shall be used to abort a control operation. It is indicated in the state machines, in which states it is possible to cancel the control operation.

Parameter name
Request
ControlObjectReference
ctlVal
operTm[0..1]
orgin
ctlNum
T
Test
Response+
ControlObjectReference
ctlVal
operTm[0..1]
orgin
ctlNum
T
Test
Response–
ControlObjectReference
ctlVal
operTm[0..1]
orgin
ctlNum
T
Test
AddCause

The service parameter operTm shall be present if the control object supports **TimeActivatedOperate** services.

NOTE The service parameters are defined in 20.5.2.

20.5.3.5 Operate (Oper)

The **Operate** service shall define the following service parameters.

Parameter name
Request
ControlObjectReference
ctlVal
orgin
ctlNum
T
Test
Check
Response+
ControlObjectReference
ctlVal
orgin
ctlNum
T
Test
Check
Response–
ControlObjectReference
ctlVal
orgin
ctlNum
T
Test
Check
AddCause

NOTE The service parameters are defined in 20.5.2.

20.5.3.6 CommandTermination (CmdTerm)

The **CommandTermination** service shall define the following service parameters.

Parameter name
Request+
ControlObjectReference
ctlVal
operTm[0..1]
orgin
ctlNum
T
Test
Check
Request–
ControlObjectReference
ctlVal
operTm[0..1]
orgin
ctlNum
T
Test
Check
AddCause

The service parameter operTm shall be present if the control object supports **TimeActivatedOperate** services. If it is then used for an Operate service, the value of operTm shall be NULL.

NOTE The service parameters are defined in 20.5.2.

20.5.3.7 TimeActivatedOperate (TimOper)

The **TimeActivatedOperate** service shall define the following service parameters.

Parameter name
Request
ControlObjectReference
ctlVal
operTm
orgin
ctlNum
T
Test
Check
Response+
ControlObjectReference
ctlVal
operTm
orgin
ctlNum
T
Test
Check
Response–
ControlObjectReference
ctlVal
operTm
orgin
ctlNum
T
Test
Check
AddCause

NOTE The service parameters are defined in 20.5.2.

The value of operTm shall not be NULL when using the **TimeActivatedOperate** service.

20.5.3.8 TimeActivatedOperateTermination (TimOperTermination)

The **TimeActivatedOperateTermination** service shall define the following service parameters.

Parameter name
Request+
ControlObjectReference
ctlVal
operTm
orgin
ctlNum
T
Test
Check
Request–
ControlObjectReference
ctlVal
operTm
orgin
ctlNum
T
Test
Check
AddCause

NOTE The service parameters are defined in 20.5.2.

20.6 Tracking of control services

20.6.1 General

Similarly to the service tracking for control block accesses defined in 15.3, the control services can be also tracked. The base concept consists in defining within the data model where to store the values of the service parameters used by a service for any control objects. For all controllable data objects of a given **Server** there shall be a single data object instance (tracking data object) available in the object directory, that will mirror the value of the service parameters of the control services. Therewith, the control service can be logged or reported to any client, as soon as the tracking data object is a **data-set** member of the **data-set** associated to a **LCB** or to a **BRCB / URCB**.

NOTE This approach does not require any change in the service models nor in the configuration language.

20.6.2 Control service tracking (CTS)

The tracking of control services shall be as defined in Table 55. It offers a logistic for tracking any control service applied to a data object. Since the type definition of the service parameter ctlVal is dependant on the common data class of the control object, there shall be one instance of the **CTS** available in a given **logical-device** for all control objects belonging to a given common data class (see IEC 61850-7-4).

Table 55 – Control service tracking (CTS) definition

CTS Class						
Attribute name	**Attribute type**	**FC**	**TrgOp**	**r/w**	**Value/value range**	**M/O/C**
Shall inherit all the data attributes of the CST CDC – see Table 29						
Specific to the CTS						
ctlVal	Depending on the CDC that is being controlled. See CDC service parameter definition	SR		r		M
operTm	TimeStamp	SR		r	NULL value, if the tracked service is not TimeActivedOperate	C1
origin	Originator	SR		r		M
ctlNum	INT8U	SR		r		M
T	TimeStamp	SR		r		M
Test	BOOLEAN	SR		r		M
Check	PACKED LIST	SR		r		M
synchrocheck	BOOLEAN					
interlock-check	BOOLEAN					
respAddCause	AddCause	SR		r		M

C1: The attribute shall be present if at least one controlled object on the IED supports time activation service.

NOTE 1 The value of the attribute Check when the serviceType is Cancel can reflect the value of the latest Check used during the service Select request respectively Operate request associated with the controllable object whose control sequence is being cancelled.

NOTE 2 The attribute T, Test and Check keep an upper case first character within the CTS CDC.

21 Time and time-synchronization model

21.1 General

The time and time-synchronization model shall provide the UTC synchronized time to applications located in server and client utility IEDs. The components of the time and time-synchronization model are depicted in Figure 46.

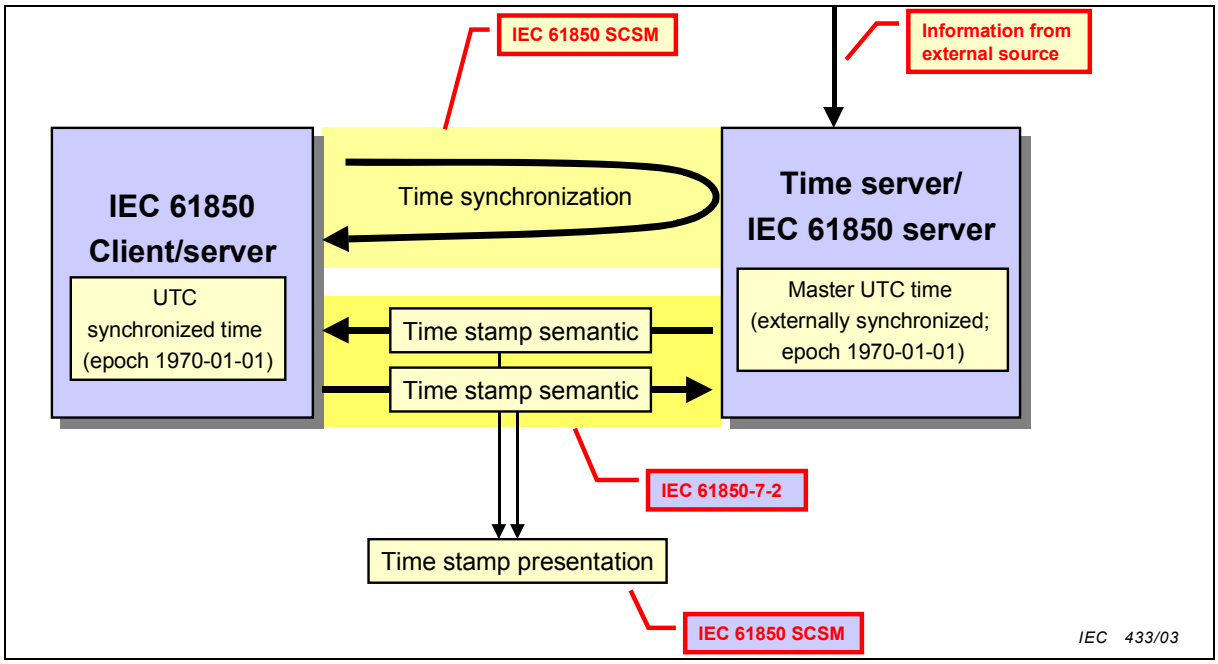


Figure 46 – Time model and time synchronization (principle)

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 21.2);

– **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 6.1.2.9);

– the **presentation** of the time stamps according to the chosen SCSM;

– the **server** and **clients** that need substation-wide synchronized time.

21.2 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).

22 Naming conventions

22.1 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 47 shows an overview of the specializations.

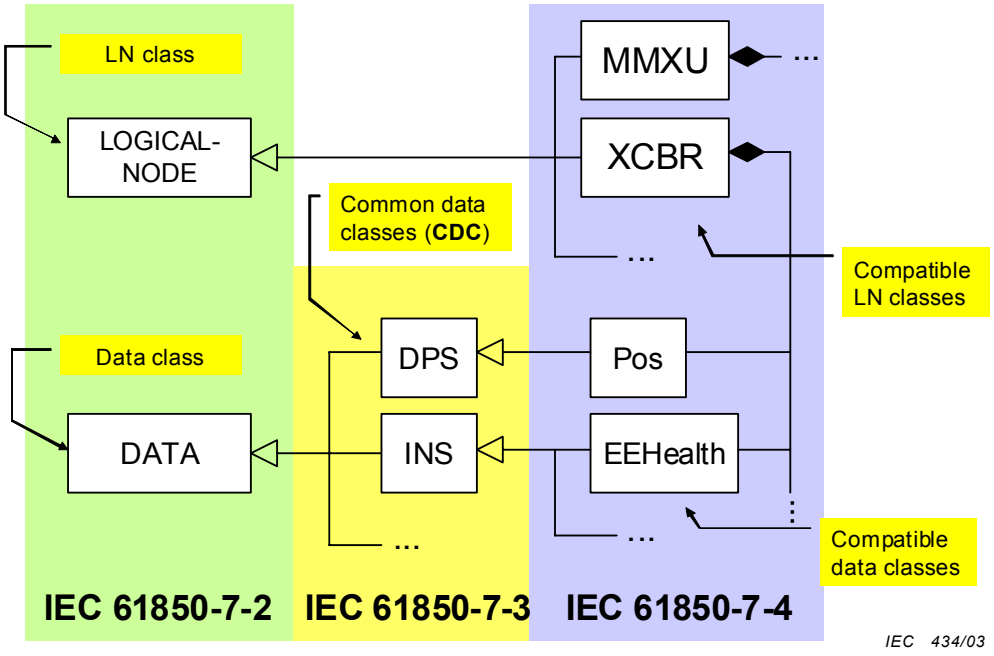


Figure 47 – Specializations

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.

22.2 Referencing an instance of a class

The following naming conventions (structure, lengths and character set) for object names and object references shall apply. See IEC 61850-7-1 on how to use the Data-Instance-ID for modeling:

LDName	≤	64 characters, application specific. It shall start with an alpha character
LNName	=	[LN-Prefix] LN class name [LN-Instance-ID]
LN-Prefix	=	m characters (application specific); it shall start with an alpha character
LN class name	=	4 alpha characters, upper case (for example, compatible logical node name as defined in IEC 61850-7-4)
LN-Instance-ID	=	n numeric characters (application specific) m+n ≤ 12 characters
DataObjectClassName	≤	12 characters (as, for example, used in IEC 61850-7-4); start with upper case alpha character; one numeric last character indicates multiple instantiation capability
DataObjectName	≤	12 characters: DataObjectClassName without last numeric character, followed by [Data-Instance-ID], if the last character was numeric
Data-Instance-ID	=	n numeric characters, optional; n shall be equal for all instances of the same data class in the same logical node
FCD	≤	61 characters including all separators "." (without the value of the **FC** and without LDName)
FCDA	≤	61 characters including all separators "." (without the value of the **FC** and without LDName)
DataSetName	≤	32 characters, start with an alpha character
CBName	≤	32 characters, start with an alpha character; recommended syntax: [CB-Prefix] CB class name [CB-Instance-ID]
CB-Prefix	=	m characters (application specific)
CB class name	=	4 characters (as defined in this part of IEC 61850)
CB-Instance-ID	=	n numeric characters (application specific)
m+n		≤ 28 characters
LogName	≤	32 characters, start with an alpha character

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 48 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 E1QA5/**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 1 The LD name E1QA5 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.

NOTE 2 The type of the **ObjectReference** is VISIBLESTRING129; this is different to the VisibleString (as defined above). The visibleString does not provide the separators ".", "/" and "()".

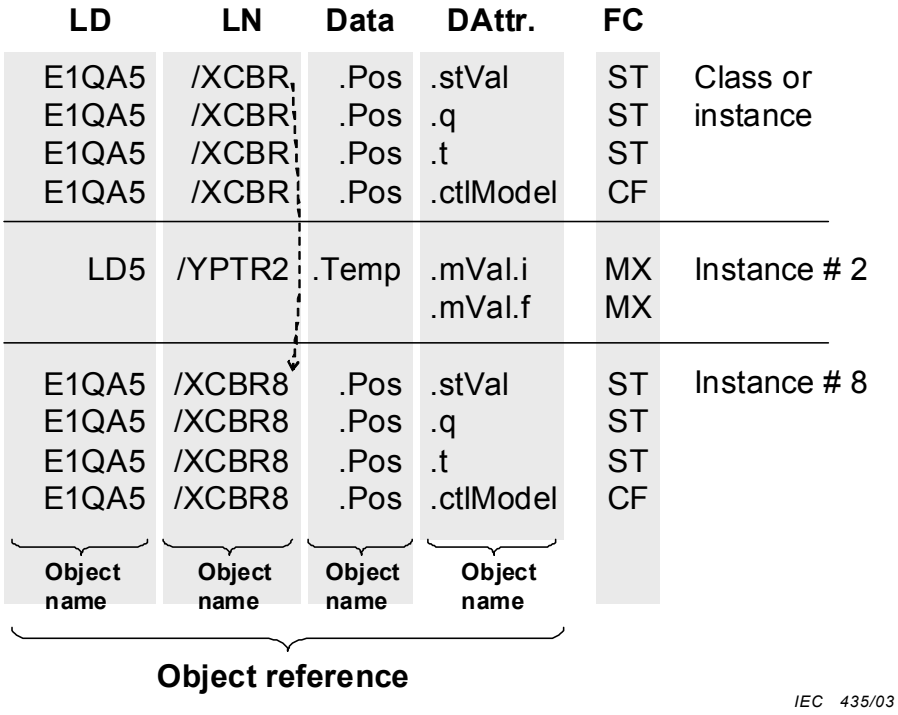


Figure 48 – Object names and object reference

22.3 Scope

Server specific scope (instances are defined outside of all **LDs** but in the server) shall be defined using the "/" and up to 64 characters to the right.

EXAMPLE /ABC.xyz

Logical device specific scope (instances are defined inside a specific **LD**) shall be defined as up to 64 characters, then "/" followed by up to 64 characters to the right.

EXAMPLE Atlanta_110/XCBR.Pos

TPAA specific scope (instances are defined inside a specific **TPAA**) shall be defined using "@"followed by up to 64 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.

23 File transfer model

23.1 File class

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 functional restricted devices. The ACSI file store addresses a single file format – sequential unstructured binary – which may contain programs, values of data objects, 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 56.

Table 56 – FILE class definition

FILE class		
Attribute name	**Attribute type**	**Value/value range/explanation**
FileName	VISIBLE STRING255	
FileSize	INT32U	
LastModified	TimeStamp	
Services GetFile SetFile DeleteFile GetFileAttributeValues		

23.1.1 FileName

The attribute **FileName** shall be the name of the file in the ACSI file store. A **FileName** shall have the following syntax:

FileName = [Path sep] File [“.” Ext]
with sep = “/” or “\”; always the same on one server
Path = Subdir [sep Subdir […]]
Ext: an alphanumeric string with maximum length of 3 characters
File, Subdir: Visible string of maximum length of 64 characters with exception of the following characters: “/”, “\”.

Observe that a SCSM may introduce more restrictions.

23.1.2 FileSize

The attribute **FileSize** (in octets) shall be the length of a file in the file store.

In case the FileSize cannot be determined (for example, in the case of an on-the-fly created COMTRADE file), the value shall be set to zero (0), or may be missing.

23.1.3 LastModified

The attribute **LastModified** shall be the time when the file was last modified.

23.2 File services

23.2.1 GetFile

23.2.1.1 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

23.2.1.2 Request

The parameter **FileName** shall specify the name of the file being transferred.

23.2.1.3 Response+

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

The parameter **File-Data** shall contain the values of the data objects transferred; the type of file-data is OCTET STRING.

23.2.1.4 Response–

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

23.2.2 SetFile

23.2.2.1 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

23.2.2.2 Request

23.2.2.2.1 FileName

The parameter **FileName** shall specify the name of the file being transferred.

23.2.2.2.2 File-Data

The parameter **File-Data** shall contain the values of the data object transferred; the type of file-data is OCTET STRING.

23.2.2.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. The type for this parameter is SCSM specific.

23.2.2.4 Response–

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

23.2.3 DeleteFile

23.2.3.1 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

23.2.3.2 Request

23.2.3.2.1 FileName

The parameter **FileName** shall specify the name of the file being deleted.

23.2.3.3 Response+

The parameter **Response+** shall indicate that the service request succeeded. The type for this parameter is SCSM specific.

23.2.3.4 Response–

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

23.2.4 GetFileAttributeValues

23.2.4.1 GetFileAttributeValues parameter

The **GetFileAttributeValues** service shall be used by a client to obtain the attributes of the file with the name given in the parameter FileName of the service request in the server's file store.

Parameter name
Request
FileName
Response+
FileName
FileSize [0..1]
LastModified
Response–
ServiceError

23.2.4.2 Request

The parameter **FileName** shall, when present, specify the name of the file whose attributes are requested to be returned to the client.

23.2.4.3 Response+

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

23.2.4.3.1 FileName

The parameter **FileName** shall provide the name of the file whose attributes are returned.

23.2.4.3.2 FileSize

The parameter **FileSize** shall contain attribute information **FileSize** describing the selected file. This information consists of the size of the file. If this can not be determined, its value shall be zero (0), or it may be missing.

23.2.4.3.3 LastModified

The parameter **FileAttribute** shall contain attribute information **LastModified** describing the selected file. This information consists of the time of the last modification.

23.2.4.4 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; and

– 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 object 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 TrgOps (trigger option) qchg of the **BRCB** or **URCB**.

A.2 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 61850-8-1 used			
B22	**SCSM**: IEC 61850-9-1 used			deprecated
B23	**SCSM**: IEC 61850-9-2 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	**Subscriber** side	O	–	
c1 – shall be 'M' if support for **logical-device** model has been declared. O – Optional M – Mandatory				

A.3 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 (B11) 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			
M7-11	conf-revision			
M8	**Unbuffered report control**	O	O	
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			
M8-8	GI			
M8-9	conf-revision			
	Logging	O	O	
M9	**Log control**	O	O	
M9-1	IntgPd			
M10	**Log**	O	O	
M11	**Control**	M	M	
If **GSE** (B31/B32) is supported				
M12	**GOOSE**	O	O	
M13	**GSSE**	O	O	deprecated

		Client/ subscriber	Server/ publisher	Value/ comments
If **SVC** (B41/B42) is supported				
M14	Multicast SVC	O	O	
M15	Unicast SVC	O	O	
For all IEDs				
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 SV models has been declared.

M – Mandatory
O – Optional

A.4 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/ subscriber	Server/ publisher	Comments
Server (Clause 7)					
S1	ServerDirectory	TP		M	
Application association (Clause 8)					
S2	Associate	TP	M	M	
S3	Abort	TP	M	M	
S4	Release	TP	M	M	
Logical device (Clause 9)					
S5	LogicalDeviceDirectory	TP	M	M	
Logical node (Clause 10)					
S6	GetLogicalNodeDirectory	TP	M	M	
S7	GetAllDataValues	TP	O	M	
Data (Clause 11)					
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/ subscriber	Server/ publisher	Comments
Data set (Clause 12)					
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	
Setting group control (Clause 16)					
S18	SelectActiveSG	TP	O	O	
S19	SelectEditSG	TP	O	O	
S20	SetEditSGValue	TP	O	O	
S21	ConfirmEditSGValues	TP	O	O	
S22	GetEditSGValue	TP	O	O	
S23	GetSGCBValues	TP	O	O	
Reporting (Clause 17)					
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 17)					
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/ subscriber	Server/ publisher	Comments
Generic substation event model (GSE)					
GOOSE (Clause 18)					
S35	SendGOOSEMessage	MC	c8	c8	
S36	GetGoReference	TP	O	c9	
S37	GetGOOSEElementNumber	TP	O	c9	
S38	GetGoCBValues	TP	O	O	
S39	SetGoCBValues	TP	O	O	
GSSE (Annex C)					
S40	SendGSSEMessage	MC	c8	c8	
S41	GetGsReference	TP	O	c9	
S42	GetGSSEDataOffset	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 19)					
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 (Clause 20)					
S51	Select		M	O	
S52	SelectWithValue	TP	M	O	
S53	Cancel	TP	O	O	
S54	Operate	TP	M	M	
S55	Command-Termination	TP	M	O	
S56	TimeActivated-Operate	TP	O	O	

File transfer (Clause 23)					
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/ subscriber	Server/ publisher	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 value of 2**-n in seconds according to 6.1.2.9.3.2

Annex B
(normative)

Formal definition of IEC 61850-7-2 Common Data Classes

B.1 Introduction

The next clause contains the formal definition of the Common Data Classes and related enumerations as described in this part of IEC 61850. The formal definition is according to the rules described for data model name space definitions in Clause 6 of this standard.

Observe that this is just a part of the overall formal data model name space definitions. More common data classes (CDC) are defined in IEC 61850-7-3, while the logical node classes are defined in IEC 61850-7-4.

B.2 Formal CDC definition

```
<?xml version="1.0" encoding="UTF-8"?>
<IEC61850 xmlns="http://www.iec.ch/61850/2003/SCL" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <NS name="IEC 61850-7-4:2007" version="2007" revision="A" application="Substation Automation">
        <DOType id="CST" cdc="CST" desc="Common Service Tracking">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the object that is used in the tracking: either a control block that is being accessed or a control object that is being controlled." dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service" cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of the CST is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry." cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
        <Doc>The common service tracking common data class (CST) offers a logistic for tracking any service, and is the base from which all other service tracking classes are derived.</Doc>
        </DOType>
        <DOType id="BTS" cdc="BTS" desc="Tracking of buffered report control block services">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the buffered control block that is being accessed." dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service" cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="C"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of the BTS is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry." cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="rptID" bType="VisString129" fc="SR" desc="" cond="M"/>
            <DA name="rptEna" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="datSet" bType="ObjRef" fc="SR" desc="" cond="M"/>
            <DA name="confRev" bType="INT32U" fc="SR" desc="" cond="M"/>
            <DA name="optFlds" bType="OptFlds" fc="SR" desc="" cond="M"/>
            <DA name="bufTm" bType="INT32U" fc="SR" desc="" cond="M"/>
            <DA name="sqNum" bType="INT16U" fc="SR" desc="" cond="M"/>
            <DA name="trgOps" bType="TrgOps" fc="SR" desc="" cond="M"/>
            <DA name="intgPd" bType="INT32U" fc="SR" desc="" cond="M"/>
            <DA name="gi" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="purgeBuf" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="entryID" bType="EntryID" fc="SR" desc="" cond="M"/>
```

```
            <DA name="timeOfEntry" bType="EntryTime" fc="SR" desc="" cond="M"/>
            <DA name="resvTms" bType="INT16" fc="SR" desc="" cond="O"/>
        <Doc>The tracking of service for buffered reporting (buffered report tracking service - BTS) offers the logistic for tracking any service that is dedicated to a buffered report control block access. The BTS CDC shall be used by data objects for service tracking of the access of all BRCB instances.</Doc>
        </DOType>
        <DOType id="STS" cdc="STS" desc="Tracking of setting group related services">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the setting group control block accessed" dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service" cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of the CDC is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry." cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="numOfSG" bType="INT8U" fc="SR" desc="" cond="M"/>
            <DA name="actSG" bType="INT8U" fc="SR" desc="" cond="M"/>
            <DA name="editSG" bType="INT8U" fc="SR" desc="" cond="M"/>
            <DA name="cnfEdit" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="lActTm" bType="Timestamp" fc="SR" desc="" cond="M"/>
            <DA name="resvTms" bType="INT16U" fc="SR" desc="" cond="O"/>
        </DOType>
        <DOType id="UTS" cdc="UTS" desc="Tracking of Unbuffered control block services">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the Unbuffered control block that is being accessed" dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service" cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of the CDC is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry." cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="rptID" bType="VisString129" fc="SR" desc="" cond="M"/>
            <DA name="rptEna" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="resv" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="datSet" bType="ObjRef" fc="SR" desc="" cond="M"/>
            <DA name="confRev" bType="INT32U" fc="SR" desc="" cond="M"/>
            <DA name="optFlds" bType="OptFlds" fc="SR" desc="" cond="M"/>
            <DA name="bufTm" bType="INT32U" fc="SR" desc="" cond="M"/>
            <DA name="sqNum" bType="INT8U" fc="SR" desc="" cond="M"/>
            <DA name="trgOps" bType="TrgOps" fc="SR" desc="" cond="M"/>
            <DA name="intgPd" bType="INT32U" fc="SR" desc="" cond="M"/>
            <DA name="gi" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
        </DOType>
        <DOType id="LTS" cdc="LTS" desc="Tracking of log control block services">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the log control block that is accessed" dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service" cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of the CDC is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry." cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="logEna" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="datSet" bType="ObjRef" fc="SR" desc="" cond="M"/>
            <DA name="optFlds" bType="OptFlds" fc="SR" desc="" cond="M"/>
            <DA name="bufTm" bType="INT32U" fc="SR" desc="" cond="M"/>
```

```
            <DA name="trgOps" bType="TrgOps" fc="SR" desc="" cond="M"/>
            <DA name="intgPd" bType="INT32U" fc="SR" desc="" cond="M"/>
            <DA name="logRef" bType="ObjRef" fc="SR" desc="" cond="M"/>
        </DOType>
        <DOType id="OTS" cdc="OTS" desc="Tracking of log query services">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the log that is accessed" dupd="true"
cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service"
cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that
is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of
the OTS is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry."
cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="oldEntrTm" bType="Timestamp" fc="SR" desc="" cond="M"/>
            <DA name="newEntrTm" bType="Timestamp" fc="SR" desc="" cond="M"/>
            <DA name="oldEntr" bType="INT32U" fc="SR" desc="" cond="M"/>
            <DA name="newEntr" bType="INT32U" fc="SR" desc="" cond="M"/>
            <DA name="rangeStrTm" bType="Timestamp" fc="SR" desc="" cond="M"/>
            <DA name="rangeStpTm" bType="Timestamp" fc="SR" desc="" cond="M"/>
            <DA name="entry" bType="EntryID" fc="SR" desc="" cond="O"/>
        </DOType>
        <DOType id="GTS" cdc="GTS" desc="Tracking of GOOSE control block services">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the GOOSE control block that is accessed"
dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service"
cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that
is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of
the CST is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry."
cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="goEna" bType="BOOLEAN" fc="SR" desc="Enabled (TRUE) | disabled (FALSE)" cond="M"/>
            <DA name="goID" bType="VisString129" fc="SR" desc="Attribute that allows a user to assign a identification for the
GOOSE message " cond="M"/>
            <DA name="datSet" bType="ObjRef" fc="SR" desc="" cond="M"/>
            <DA name="confRev" bType="INT32U" fc="SR" desc="" cond="M"/>
            <DA name="ndsCom" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="dstAddress" bType="PhyComAddr" fc="SR" desc="" cond="M"/>
        </DOType>
        <DOType id="MTS" cdc="MTS" desc="Tracking of Multicast SV control block services">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the Multicast sampled value control block that is
accessed" dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service"
cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that
is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of
the CDC is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry."
cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="svEna" bType="BOOLEAN" fc="SR" desc="Enabled (TRUE) | disabled (FALSE), DEFAULT FALSE"
cond="M"/>
            <DA name="msvID" bType="VisString129" fc="SR" desc="" cond="M"/>
            <DA name="datSet" bType="ObjRef" fc="SR" desc="" cond="M"/>
            <DA name="confRev" bType="INT32U" fc="SR" desc="" cond="M"/>
            <DA name="smpMod" bType="Enum" type="smpMod" fc="SR" desc="samples per nominal period (DEFAULT) |
samples per second | seconds per sample" cond="M"/>
            <DA name="smpRate" bType="INT16U" fc="SR" desc="" cond="M"/>
            <DA name="optFlds" bType="OptFlds" fc="SR" desc="" cond="M"/>
```

```
            <DA name="dstAddress" bType="PhyComAddr" fc="SR" desc="" cond="M"/>
        </DOType>
        <DOType id="NTS" cdc="NTS" desc="Tracking of Unicast SV control block services">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the Unicast sampled value control block that is
accessed" dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service"
cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that
is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of
the CDC is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry."
cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="svEna" bType="BOOLEAN" fc="SR" desc="Enabled (TRUE) | disabled (FALSE), DEFAULT FALSE"
cond="M"/>
            <DA name="res" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="usvID" bType="VisString129" fc="SR" desc="" cond="M"/>
            <DA name="datSet" bType="ObjRef" fc="SR" desc="" cond="M"/>
            <DA name="confRev" bType="INT32U" fc="SR" desc="" cond="M"/>
            <DA name="smpMod" bType="Enum" type="smpMod" fc="SR" desc="samples per nominal period (DEFAULT) |
samples per second | seconds per sample" cond="M"/>
            <DA name="smpRate" bType="INT16U" fc="SR" desc="" cond="M"/>
            <DA name="optFlds" bType="OptFlds" fc="SR" desc="" cond="M"/>
            <DA name="dstAddress" bType="PhyComAddr" fc="SR" desc="" cond="M"/>
        </DOType>
        <DOType id="BoolCTS" cdc="CTS" desc="Tracking of Boolean control services">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the control object that is being controlled."
dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service"
cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that
is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of
the CDC is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry."
cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="ctlVal" bType="BOOLEAN" fc="SR" desc="The type is depending of the CDC that is being controlled.
See CDC service parameter definition" cond="M"/>
            <DA name="operTm" bType="Timestamp" fc="SR" desc="" cond="O"/>
            <DA name="origin" bType="Struct" type="Originator" fc="SR" desc="" cond="M"/>
            <DA name="ctlNum" bType="INT8U" fc="SR" desc="" cond="M"/>
            <DA name="T" bType="Timestamp" fc="SR" desc="" cond="M"/>
            <DA name="Test" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="Check" bType="Check" fc="SR" desc="" cond="M"/>
            <DA name="respAddCause" bType="Enum" type="AddCause" fc="SR" desc="AddCause" cond="M"/>
        <Doc>Command tracking for BOOLEAN commands. Observe that for each allowed command data type there must be
defined an appropriate CDC variant (DOType) of common data class CTS. Some examples follow below.
        </Doc>
        </DOType>
        <DOType id="Int8CTS" cdc="CTS" desc="Tracking of INT8 based control services">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the control object that is being controlled."
dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service"
cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that
is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of
the CDC is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry."
cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="ctlVal" bType="INT8" fc="SR" desc="INT8 control value" cond="M"/>
            <DA name="operTm" bType="Timestamp" fc="SR" desc="" cond="O"/>
```

```
            <DA name="origin" bType="Struct" type="Originator" fc="SR" desc="" cond="M"/>
            <DA name="ctlNum" bType="INT8U" fc="SR" desc="" cond="M"/>
            <DA name="T" bType="Timestamp" fc="SR" desc="" cond="M"/>
            <DA name="Test" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="Check" bType="Check" fc="SR" desc="" cond="M"/>
            <DA name="respAddCause" bType="Enum" type="AddCause" fc="SR" desc="AddCause" cond="M"/>
        </DOType>
        <DOType id="Int32CTS" cdc="CTS" desc="Tracking of INT32 control services">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the control object that is being controlled."
dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service"
cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that
is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of
the CDC is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry."
cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="ctlVal" bType="INT32" fc="SR" desc="TINT32 control value" cond="M"/>
            <DA name="operTm" bType="Timestamp" fc="SR" desc="" cond="O"/>
            <DA name="origin" bType="Struct" type="Originator" fc="SR" desc="" cond="M"/>
            <DA name="ctlNum" bType="INT8U" fc="SR" desc="" cond="M"/>
            <DA name="T" bType="Timestamp" fc="SR" desc="" cond="M"/>
            <DA name="Test" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="Check" bType="Check" fc="SR" desc="" cond="M"/>
            <DA name="respAddCause" bType="Enum" type="AddCause" fc="SR" desc="AddCause" cond="M"/>
        </DOType>
        <DOType id="AnalogCTS" cdc="CTS" desc="tracking of analog control services">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the control object that is being controlled."
dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service"
cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that
is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of
the CTS is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry."
cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="ctlVal" bType="Struct" type="AnalogueValue" fc="SR" desc="Analog control value - can be INT32 or
FLOAT32" cond="M"/>
            <DA name="operTm" bType="Timestamp" fc="SR" desc="" cond="O"/>
            <DA name="origin" bType="Struct" type="Originator" fc="SR" desc="" cond="M"/>
            <DA name="ctlNum" bType="INT8U" fc="SR" desc="" cond="M"/>
            <DA name="T" bType="Timestamp" fc="SR" desc="" cond="M"/>
            <DA name="Test" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="Check" bType="Check" fc="SR" desc="" cond="M"/>
            <DA name="respAddCause" bType="Enum" type="AddCause" fc="SR" desc="AddCause" cond="M"/>
        </DOType>
        <DOType id="ModCTS" cdc="CTS" desc="Tracking of Enumeration based control services">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the control object that is being controlled."
dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service"
cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that
is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of
the CDC is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry."
cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="ctlVal" bType="Enum" type="Beh" fc="SR" desc="Mod value to be set; use correct Enumeration type
for other Enumerations as defined in IEC 61850-7-4" cond="M"/>
            <DA name="operTm" bType="Timestamp" fc="SR" desc="" cond="O"/>
```

```
            <DA name="origin" bType="Struct" type="Originator" fc="SR" desc="" cond="M"/>
            <DA name="ctlNum" bType="INT8U" fc="SR" desc="" cond="M"/>
            <DA name="T" bType="Timestamp" fc="SR" desc="" cond="M"/>
            <DA name="Test" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="Check" bType="Check" fc="SR" desc="" cond="M"/>
            <DA name="respAddCause" bType="Enum" type="AddCause" fc="SR" desc="AddCause" cond="M"/>
        </DOType>
        <DOType id="BSC_CTS" cdc="CTS" desc="Tracking of Binary controlled integer control service">
            <DA name="objRef" bType="ObjRef" fc="SR" desc="Reference of the control object that is being controlled."
dupd="true" cond="M"/>
            <DA name="serviceType" bType="Enum" type="ServiceType" fc="SR" desc="Type of the tracked service"
cond="M"/>
            <DA name="errorCode" bType="Enum" type="ServiceError" fc="SR" desc="See error associated to the service that
is specified by serviceType; value no-error for successful service" cond="M"/>
            <DA name="originatorID" bType="Octet64" fc="SR" desc="Originator of the service" cond="O"/>
            <DA name="t" bType="Timestamp" fc="SR" desc="TimeStamp of the completion of the service. If the instance of
the CDC is logged or reported using the buffered reporting model, t shall be identical to the EntryTime of the associated Entry."
cond="M"/>
            <DA name="d" bType="VisString255" fc="DC" desc="" cond="O"/>
            <DA name="dU" bType="Unicode255" fc="DC" desc="" cond="O"/>
            <DA name="cdcNs" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="cdcName" bType="VisString255" fc="EX" desc="" cond="AC_DLNDA_M"/>
            <DA name="dataNs" bType="VisString255" fc="EX" desc="" cond="AC_DLN_M"/>
            <DA name="ctlVal" bType="Tcmd" fc="SR" desc="Tap changer Raise / Lower" cond="M"/>
            <DA name="operTm" bType="Timestamp" fc="SR" desc="" cond="O"/>
            <DA name="origin" bType="Struct" type="Originator" fc="SR" desc="" cond="M"/>
            <DA name="ctlNum" bType="INT8U" fc="SR" desc="" cond="M"/>
            <DA name="T" bType="Timestamp" fc="SR" desc="" cond="M"/>
            <DA name="Test" bType="BOOLEAN" fc="SR" desc="" cond="M"/>
            <DA name="Check" bType="Check" fc="SR" desc="" cond="M"/>
            <DA name="respAddCause" bType="Enum" type="AddCause" fc="SR" desc="AddCause" cond="M"/>
        </DOType>
        <DAType id="AnalogueValue">
            <BDA name="i" bType="INT32" cond="GC_1" desc="The value of i shall be an integer representation of the
measured value."/>
            <BDA name="f" bType="FLOAT32" cond="GC_1" desc="The value of f shall be the FLOAT representation of the
measured value. f shall represent the technological value in SI units."/>
        </DAType>
        <DAType id="Originator">
            <BDA name="orCat" bType="Enum" type="orCategory" cond="M" desc="Originator category indicates who/what
caused the change of a controllable value. See OriginatorCategory."/>
            <BDA name="orIdent" bType="Octet64" cond="M" desc="Originator identification shall show the address of the
originator who caused the change of the value. If NULL, originator of a particular action is not known or is not reported."/>
        </DAType>
        <EnumType id="ServiceType">
            <EnumVal ord="0">Unknown</EnumVal>
            <EnumVal ord="1">Associate</EnumVal>
            <EnumVal ord="2">Abort</EnumVal>
            <EnumVal ord="3">Release</EnumVal>
            <EnumVal ord="4">GetServerDirectory</EnumVal>
            <EnumVal ord="5">GetLogicalDeviceDirectory</EnumVal>
            <EnumVal ord="6">GetAllDataValues</EnumVal>
            <EnumVal ord="7">GetDataValues</EnumVal>
            <EnumVal ord="8">SetDataValues</EnumVal>
            <EnumVal ord="9">GetDataDirectory</EnumVal>
            <EnumVal ord="10">GetDataDefinition</EnumVal>
            <EnumVal ord="11">GetDataSetValues</EnumVal>
            <EnumVal ord="12">SetDataSetValues</EnumVal>
            <EnumVal ord="13">CreateDataSet</EnumVal>
            <EnumVal ord="14">DeleteDataSet</EnumVal>
            <EnumVal ord="15">GetDataSetDirectory</EnumVal>
            <EnumVal ord="16">SelectActiveSG</EnumVal>
            <EnumVal ord="17">SelectEditSG</EnumVal>
            <EnumVal ord="18">SetEditSGValue</EnumVal>
            <EnumVal ord="19">ConfirmEditSGValues</EnumVal>
            <EnumVal ord="20">GetEditSGValue</EnumVal>
            <EnumVal ord="21">GetSGCBValues</EnumVal>
            <EnumVal ord="22">Report</EnumVal>
            <EnumVal ord="23">GetBRCBValues</EnumVal>
            <EnumVal ord="24">SetBRCBValues</EnumVal>
            <EnumVal ord="25">GetURCBValues</EnumVal>
            <EnumVal ord="26">SetURCBValues</EnumVal>
            <EnumVal ord="27">GetLCBValues</EnumVal>
            <EnumVal ord="28">SetLCBValues</EnumVal>
            <EnumVal ord="29">QueryLogByTime</EnumVal>
            <EnumVal ord="30">QueryLogAfter</EnumVal>
            <EnumVal ord="31">GetLogStatus</EnumVal>
```

```
        <EnumVal ord="32">SendGOOSEMessage</EnumVal>
        <EnumVal ord="33">GetGoCBValues</EnumVal>
        <EnumVal ord="34">SetGoCBValues</EnumVal>
        <EnumVal ord="35">GetGoReference</EnumVal>
        <EnumVal ord="36">GetGOOSEElementNumber</EnumVal>
        <EnumVal ord="37">SendMSVMessage</EnumVal>
        <EnumVal ord="38">GetMSVCBValues</EnumVal>
        <EnumVal ord="39">SetMSVCBValues</EnumVal>
        <EnumVal ord="40">SendUSVMessage</EnumVal>
        <EnumVal ord="41">GetUSVCBValues</EnumVal>
        <EnumVal ord="42">SetUSVCBValues</EnumVal>
        <EnumVal ord="43">Select</EnumVal>
        <EnumVal ord="44">SelectWithValue</EnumVal>
        <EnumVal ord="45">Cancel</EnumVal>
        <EnumVal ord="46">Operate</EnumVal>
        <EnumVal ord="47">CommandTermination</EnumVal>
        <EnumVal ord="48">TimeActivatedOperate</EnumVal>
        <EnumVal ord="49">GetFile</EnumVal>
        <EnumVal ord="50">SetFile</EnumVal>
        <EnumVal ord="51">DeleteFile</EnumVal>
        <EnumVal ord="52">GetFileAttributValues</EnumVal>
        <EnumVal ord="53">TimeSynchronisation</EnumVal>
        <EnumVal ord="54">InternalChange</EnumVal>
    </EnumType>
    <EnumType id="ServiceError">
        <EnumVal ord="0">no-error</EnumVal>
        <EnumVal ord="1">instance-not-available</EnumVal>
        <EnumVal ord="2">instance-in-use</EnumVal>
        <EnumVal ord="3">access-violation</EnumVal>
        <EnumVal ord="4">access-not-allowed-in-current-state</EnumVal>
        <EnumVal ord="5">parameter-value-inappropriate</EnumVal>
        <EnumVal ord="6">parameter-value-inconsistent</EnumVal>
        <EnumVal ord="7">class-not-supported</EnumVal>
        <EnumVal ord="8">instance-locked-by-other-client</EnumVal>
        <EnumVal ord="9">control-must-be-selected</EnumVal>
        <EnumVal ord="10">type-conflict</EnumVal>
        <EnumVal ord="11">failed-due-to-communications-constraint</EnumVal>
        <EnumVal ord="12">failed-due-to-server-constraint</EnumVal>
    </EnumType>
    <EnumType id="AddCause">
        <EnumVal ord="0">Unknown</EnumVal>
        <EnumVal ord="1">Not-supported</EnumVal>
        <EnumVal ord="2">Blocked-by-switching-hierarchy</EnumVal>
        <EnumVal ord="3">Select-failed</EnumVal>
        <EnumVal ord="4">Invalid-position</EnumVal>
        <EnumVal ord="5">Position-reached</EnumVal>
        <EnumVal ord="6">Parameter-change-in-execution</EnumVal>
        <EnumVal ord="7">Step-limit</EnumVal>
        <EnumVal ord="8">Blocked-by-Mode</EnumVal>
        <EnumVal ord="9">Blocked-by-process</EnumVal>
        <EnumVal ord="10">Blocked-by-interlocking</EnumVal>
        <EnumVal ord="11">Blocked-by-synchrocheck</EnumVal>
        <EnumVal ord="12">Command-already-in-execution</EnumVal>
        <EnumVal ord="13">Blocked-by-health</EnumVal>
        <EnumVal ord="14">1-of-n-control</EnumVal>
        <EnumVal ord="15">Abortion-by-cancel</EnumVal>
        <EnumVal ord="16">Time-limit-over</EnumVal>
        <EnumVal ord="17">Abortion-by-trip</EnumVal>
        <EnumVal ord="18">Object-not-selected</EnumVal>
        <EnumVal ord="19">  Object-already-selected </EnumVal>
        <EnumVal ord="20">No-access-authority</EnumVal>
        <EnumVal ord="21">Ended-with-overshoot</EnumVal>
        <EnumVal ord="22">Abortion-due-to-deviation</EnumVal>
        <EnumVal ord="23">Abortion-by-communication-loss </EnumVal>
        <EnumVal ord="24">Blocked-by-command</EnumVal>
        <EnumVal ord="25">None</EnumVal>
        <EnumVal ord="26">Locked-by-other-client</EnumVal>
        <EnumVal ord="27">Inconsistent-parameters</EnumVal>
    </EnumType>
    <EnumType id="smpMod">
        <EnumVal ord="0">SamplesPerPeriod</EnumVal>
        <EnumVal ord="1">SamplesPerSecond</EnumVal>
        <EnumVal ord="2">SecondsPerSample</EnumVal>
    </EnumType>
    <EnumType id="orCategory">
        <EnumVal ord="0">not-supported</EnumVal>
        <EnumVal ord="1">bay-control</EnumVal>
```

```
            <EnumVal ord="2">station-control</EnumVal>
            <EnumVal ord="3">remote-control</EnumVal>
            <EnumVal ord="4">automatic-bay</EnumVal>
            <EnumVal ord="5">automatic-station</EnumVal>
            <EnumVal ord="6">automatic-remote</EnumVal>
            <EnumVal ord="7">maintenance</EnumVal>
            <EnumVal ord="8">process</EnumVal>
        </EnumType>
        <EnumType id="Beh">
            <EnumVal ord="1">on</EnumVal>
            <EnumVal ord="2">blocked</EnumVal>
            <EnumVal ord="3">test</EnumVal>
            <EnumVal ord="4">test/blocked</EnumVal>
            <EnumVal ord="5">off</EnumVal>
        </EnumType>
    </NS>
</IEC61850>
```

Annex C
(informative)

Generic substation state event (GSSE) control block (GsCB)

C.1 General

The use of the GSSE service shall be avoided in new system designs. This annex keeps its definition if needed for backwards compatibility to IEDs supporting only previous versions of this standard, and only support this service.

C.2 GsCB class definition

The specifics for the **GsCB** model (compared to the **GoCB** model) are depicted in the shadowed area in Figure C.1.

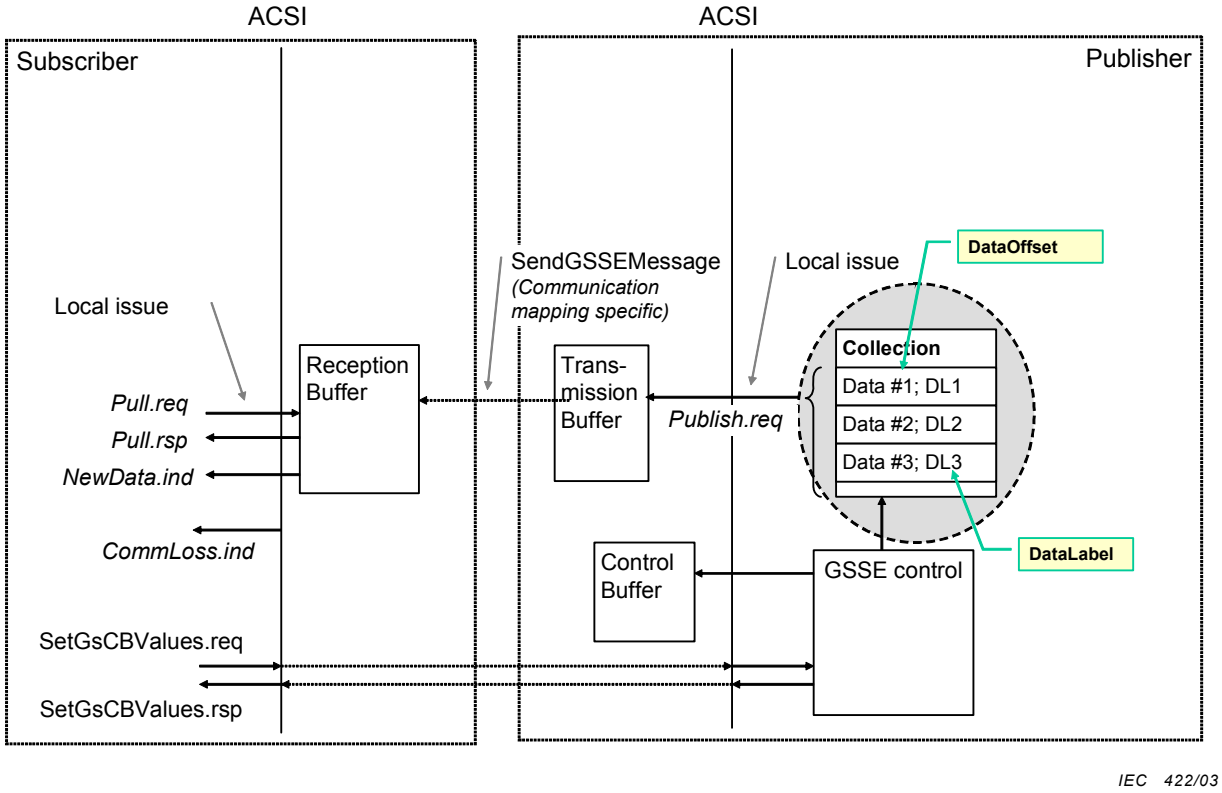


Figure C.1 – GsCB model

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 C.1.

Table C.1 – GSSE control block class definition

GsCB class			
Attribute name	**Attribute type**	**r/w**	**Value/value range/explanation**
GsCBName	ObjectName		Instance name of an instance of GsCB
GsCBRef	ObjectReference		Path-name of an instance of GsCB
GsEna	BOOLEAN		Enabled (TRUE) | disabled (FALSE)
GsID	VISIBLE STRING65		
DataLabel [1..n]	VISIBLE STRING65		
LSentData [1..n]	GSSEData		Derived from GSSE message
DstAddress	PHYCOMADDR		
Services SendGSSEMessage GetGsReference GetGSSEDataOffset GetGsCBValues SetGsCBValues			

C.3 Generic substation state event (GSSE) control block class attributes

C.3.1 GsCBName – GSSE control name

The attribute **GsCBName** shall unambiguously identify a **GsCB** within the scope of a **LLN0**.

C.3.2 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

C.3.3 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.

C.3.4 GsID – GSSE identification

The attribute **GsID** shall be a user definable unique identification of the context of the GSSE message.

NOTE 1 The context of the GSSE message is defined by the values configured in the GsCB.

NOTE 2 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.

C.3.5 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 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**.

C.3.6 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 maximum for the number of data values shall be at least 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.

C.3.7 DstAddress

The attribute DstAddress shall be the SCSM specific addressing information like media access address, priority, and other information.

C.4 GSSE service definitions

C.4.1 Overview

For the **GsCB**, the following services are defined:

Service	Description
SendGSSEMessage	Send GSSE message
GetGsReference	Retrieve the **DataLabel** of a specific value associated with the **GSSE** message
GetGSSEDataOffset	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**

C.4.2 SendGSSEMessage

C.4.2.1 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

C.4.2.2 Request

C.4.2.2.1 GSSE message

The parameter **GSSE message** shall specify the **GSSE** message as defined in C.4.2.5 of the given **GsCB**.

C.4.2.3 GetGsReference

C.4.2.3.1 GetGsReference parameter table

A client shall use the **GetGsReference** 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

C.4.2.3.1.1 Request

GsCBReference

The parameter **GsCBReference** shall identify the attribute **GsCBRef** of the **GsCB** for which **DataLabels** are being requested.

DataOffset [1..n]

The parameter **DataOffset** shall contain a number identifying a member of the **Collection**.

C.4.2.3.1.2 Response+

GsCBReference

The parameter **GsCBReference** shall contain the parameter that identifies the attribute **GoCBRef** of the **GsCB** for which **DataLabels** are returned.

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**.

C.4.2.3.1.3 Response–

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

C.4.2.4 GetGSSEDataOffset

C.4.2.4.1 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**.

Parameter name
Request
GsCBReference
DataLabel [1..n]
Response+
GsCBReference
DataOffset [1..n]
Response–
ServiceError

C.4.2.4.1.1 Request

GsCBReference

The parameter **GsCBReference** shall identify the attribute **GsCBRef** of the **GsCB** for which **MemberOffset** are being requested.

DataLabel [1..n]

The parameter **DataLabel** shall contain the **DataLabel** for which the **DataOffset** of the **Collection** is requested.

C.4.2.4.1.2 Response+

GsCBReference

The parameter **GsCBReference** shall contain the parameter that identifies the attribute **GoCBRef** of the **GsCB** for which **DataLabels** are returned.

DataOffset [1..n]

The parameter **DataOffset** shall contain a number identifying a member of the **Collection**. A value of 0 (Zero) shall indicate that no **DataOffset** is defined for the member being requested with the respective **DataLabel**.

C.4.2.4.1.3 Response–

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

C.4.2.4.2 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**.

Parameter name
Request
GsCBReference
Response+
GsEnable
GSSEID
DataLabel [1..n]
LastSentData [1..n]
Response–
ServiceError

C.4.2.4.2.1 Request

GsCBReference

The parameter **GsCBReference** shall specify the **ObjectReference** of the **GsCB**.

The service parameter **GsCBReference** shall be **LDName/LLN0.GsCBName**.

C.4.2.4.2.2 Response+

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

GsEnable

The parameter **GsEnable** shall contain the value of the corresponding attribute **GsEna** of the referenced **GsCB**.

GSSEID

The parameter **GSSEID** shall contain the value of the corresponding attribute **GsID** of the referenced **GsCB**.

DataLabel [1..n]

The parameter **DataLabel** shall contain the **DataLabel** of the **Collection**.

LastSentData [1..n]

The parameter **LastSentData** shall contain the value of the attribute **LSentData** of the **GsCB**.

C.4.2.4.2.3 Response–

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

C.4.2.4.3 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**.

Parameter name
Request
GsCBReference
GsEnable [0..1]
GSSEID [0..1]
Response+
Response–
ServiceError

C.4.2.4.3.1 Request

GsCBReference

The parameter **GsCBReference** shall specify the ObjectReference of the **GsCB**.

The service parameter **GsCBReference** shall be **LDName/LLN0.GsCBName**.

GsEnable [0..1]

The parameter **GsEnable** shall contain the value for the corresponding attribute **GsEna** of the referenced **GsCB**.

GSSEID [0..1]

The parameter **GSSEID** shall contain the value for the corresponding attribute **GsID** of the referenced **GsCB**.

C.4.2.4.3.2 Response+

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

C.4.2.4.3.3 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.

C.4.2.5 Generic substation state event (GSSE) message

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 C.2.

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 C.2 – GSSE message definition

GSSE message		
Parameter name	**Parameter type**	**Value/value range/explanation**
GsID	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)

GsID – application identifier

The parameter **GsID** shall contain the value of the attribute GsID of the GsCB.

T – time stamp

The parameter **T** shall contain the time at which the **StNum** attribute was incremented.

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.

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.

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.

PhsID – phase identification

The parameter PhsID shall indicate faulted phases.

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**.

Bibliography

IEC 61400-25-2, *Wind turbines – Part 25-2: Communications for monitoring and control of wind power plants – Information models*

IEC 61850-8-x (all parts), *Communication networks and systems for power utility automation – Part 8: Specific Communication Service Mapping (SCSM)*

IEC 61850-9-x (all parts), *Communication networks and systems for power utility automation – Part 9: Specific Communication Service Mapping (SCSM)*

IEC 61850-9-1, *Communication networks and systems in substations – Part 9-1: Specific Communication Service Mapping (SCSM) – Sampled values over serial unidirectional multidrop point to point link*

IEEE-SA TR 1550:1999, *Utility Communications Architecture (UCA™) Version 2*

UTC (Universal Time Coordinated):
http://en.wikipedia.org/wiki/Coordinated_Universal_Time

Index

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

Tel: + 41 22 919 02 11
Fax: + 41 22 919 03 00
info@iec.ch
www.iec.ch