# IEC TS 61850-7-7

Edition 1.0 2018-03

# TECHNICAL SPECIFICATION

colour
inside

Communication networks and systems for power utility automation –
Part 7-7: Machine-processable format of IEC 61850-related data models for tools

IEC TS 61850-7-7

Edition 1.0 2018-03

# TECHNICAL SPECIFICATION

colour inside

## Communication networks and systems for power utility automation – Part 7-7: Machine-processable format of IEC 61850-related data models for tools

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ICS 33.200 ISBN 978-2-8322-5443-1

## CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____________

# COMMUNICATION NETWORKS AND SYSTEMS FOR POWER UTILITY AUTOMATION –

# Part 7-7: Machine-processable format of IEC 61850-related data models for tools

## 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.

The main task of IEC technical committees is to prepare International Standards. In exceptional circumstances, a technical committee may propose the publication of a technical specification when

- the required support cannot be obtained for the publication of an International Standard, despite repeated efforts, or
- the subject is still under technical development or where, for any other reason, there is the future but no immediate possibility of an agreement on an International Standard.

Technical specifications are subject to review within three years of publication to decide whether they can be transformed into International Standards.

Technical Specification IEC TS 61850-7-7 has been prepared by IEC technical committee 57: Power systems management and associated information exchange.

The text of this technical specification is based on the following documents:

| Enquiry draft | Report on voting |
|---|---|
| 57/1925/DTS | 57/1956/RVDTS |

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

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

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

This IEC standard includes Code Components i.e. components that are intended to be directly processed by a computer. Such content is any text found between the markers <CODE BEGINS> and <CODE ENDS>, or otherwise is clearly labeled in this standard as a Code Component.

The purchase of this IEC standard carries a copyright license for the purchaser to sell software containing Code Components from this standard directly to end users and to end users via distributors, subject to IEC software licensing conditions, which can be found at: http://www.iec.ch/CCv1.

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

- transformed into an International standard,
- 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

Year after year the IEC 61850 data models are extended both in depth with hundreds of new data items, and in width with tens of new parts.

In order to foster an active tool market with good quality, and at the end to improve IEC 61850 interoperability, a machine-processable file, describing data model related parts of the standard as input, is needed. This is the purpose of the new language Name Space Definition (NSD) defined by this part of IEC 61850.

This will avoid the need for any engineering tool related to the IEC 61850 data models to get the content of the standard manually entered, with a high risk of mistakes. This will also help to easily spread any corrections to the data model, as requested to reach interoperability. Tool vendors will be able to integrate NSD in their tools to distribute the standard data models directly to end users.

# COMMUNICATION NETWORKS AND SYSTEMS FOR POWER UTILITY AUTOMATION –

# Part 7-7: Machine-processable format of IEC 61850-related data models for tools

## 1 Scope

### 1.1 General

This part of IEC 61850, which is a Technical Specification, specifies a way to model the code components of IEC 61850 data model (e.g., the tables describing logical nodes, common data classes, structured data attributes, and enumerations) in an XML format that can be imported and interpreted by tools. The following main use cases are supported:

- Generation of SCL data type templates for system specification or ICD files.
- Validation of SCL data type templates.
- Definition of private extensions by following the rules of the standard.
- Adapting rapidly the whole engineering chain as soon as a new version of IEC 61850 data model (an addendum, a corrigenda or a Tissue) affects the content of the standard.
- Provide tool-neutral textual help to users of tools on the data model contents.
- Supporting multi-language publication, i.e., enabling the expression of the data model in different languages, through a machine processable format.

The purpose of this proposal is limited to the publication of the XML format which should support the data model part of any IEC 61850 related standard. The publication of code components themselves will be part of the related IEC 61850 part.

### 1.2 Namespace name and version

The new namespace name and version section is mandatory for any IEC 61850 namespace (as defined by IEC 61850-7-1:2011).

The parameters which identify this new release of the NSD namespace xmlns:nsd="http://www.iec.ch/61850/2016/NSD" are:

- Namespace Version: 2017
- Namespace Revision: A
- Namespace Release: 1
- Namespace release date: 2017/08/28

| Edition | Publication date | Webstore | Namespace |
|---|---|---|---|
| Edition 1.0 | 2017-?? | IEC 61850-7-7:2017 | IEC 61850-7-7:2017A |

The namespace version relates to the edition of the standard: here namespace version 2017 refers to the first edition of this document.

Then, the revision relates to amendments if any: as for the current version of this document, revision A corresponds to the original edition, without amendment. For the first amendment, the revision will be B, etc.

Finally, namespace release indicates an update of the related code component (if any) without publication of a new version or revision of the current document. This could be used for internal release of the code component during development of a new version of the document, or to provide fixes of interoperability tissues without need to enter into a full document update process.

The namespace release date is used for information purpose, to indicate when the namespace has been created.

## 1.3 Code Component distribution

The Code Components included in this document are also available as electronic machine readable files at:
http://www.iec.ch/public/TC57/supportdocuments/IEC_61850-7-7.2017.NSD.2017A.full.zip

The Code Component(s) included in this document are potentially subject to maintenance works and the latest release is available in the repository located at:
http://www.iec.ch/TC57/supportdocuments

The latest version/release of the document will be found by selecting the file IEC_61850-7-7.2017.NSD.{VersionStateInfo}.full.zip with the filed VersionStateInfo of the highest value.

Each Code Component is a ZIP package containing the electronic representation of the Code Component itself, with a file describing the content of the package (IECManifest.xml).

The IECManifest contains different sections giving information on:

– The copyright notice
– The identification of the code component
– The publication related to the code component
– The list of the electronic files which compose the code component
– An optional list of history files to track changes during the evolution process of the code component

The IECManifest related to this publication is:

```
<IECManifest xmlns="http://www.iec.ch/CC/2017/IECManifest" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.iec.ch/CC/2017/IECManifest IECManifest.xsd">
        <Copyright>
                <Notice>
                        COPYRIGHT (c) IEC, 2017. This version of this XSD is part of IEC 61850-7-7:2017; see the IEC 61850-
7-7:2017 for full legal notices. In case of any differences between the here-below code and the IEC published content, the here-
below definition supersedes the IEC publication; it may contain updates. See history files. The whole document has to be taken
into account to have a full description of this code component.
                See www.iec.ch/CCv1 for copyright details.
                </Notice>
                <License uri="www.iec.ch/CCv1">IEC License</License>
        </Copyright>
        <CodeComponent id="IEC_61850-7-7.2017A.NSD.XSD" name="NSD schema 2017A" content="full" date="2017-08-
09">
                <Publication name="IEC 61850-7-7.2017_ed1.0" comment="Machine-processable format of IEC 61850-related
data models for tools"/>
                <File name="NSD.xsd" category="normative" content="full" comment="Schema describing the namespace
files"/>
                <File name="IECCopyright.xsd" category="normative" content="full" comment="Schema included in NSD to
integrate management of IEC Copyright notice"/>
                <File name="NSD.Doc.HTML.zip" category="normative" content="full" comment="Zip archive containing the
HTML documentation of the NSD. Contains the 'NSD.html' file and all related pictures"/>
                <HistoryFile name="History.NSD.v1.0.txt" startingDate="2015-10-12" endingDate="2017-01-25"
startingVersion="NSD.XSD.v0.3" endingVersion="NSD.XSD.v1.0"/>
                <HistoryFile name="History.NSD.v1.1.txt" startingDate="2017-01-25" endingDate="2017-06-23"
startingVersion="NSD.XSD.v1.0" endingVersion="NSD.XSD.v1.1"/>
                <HistoryFile name="History.NSD.2017A.txt" startingDate="2017-06-23" endingDate="2017-08-09"
startingVersion="NSD.XSD.v1.1" endingVersion="NSD.XSD.2017A"/>
```

```
    </CodeComponent>
</IECManifest>
```

The package is identified using the following naming rule:

{RefStandard}.{CodeComponentName}.{VersionRevision}.{LightFull}{PublicationStage}.zip

For current publication, the Code Component package name is:

IEC_61850-7-7.2017.NSD.2017A.full.zip

The life cycle of a code component is not restricted to the life cycle of the related publication. The publication life cycle goes through two stages, Version (corresponding to an edition) and Revision (corresponding to an amendment). A third publication stage (Release) allows publication of Code Component without need to publish an amendment.

This is useful when InterOp Tissues need to be fixed. Then a new release of the Code Component will be released, which supersedes the previous release, and distributed through the IEC web site.

## 2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 TS 61850-2, *Communication networks and systems in substations – Part 2: Glossary*

IEC 61850-6, *Communication networks and systems for power utility automation – Part 6: 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*

IEC 61850-7-2, *Communication networks and systems for power utility automation – Part 7-2: Basic information and communication structure – Abstract communication service interface (ACSI)*

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

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*

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*

ISO 639-1:2002, *Codes for the representation of names of languages – Part 1: Alpha-2 code*

## 3 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC TS 61850-2 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at http://www.iso.org/obp

**3.1**
**data model**
hierarchical structure of data and the related services used to define functions and devices in IEC 61850

Note 1 to entry: The IEC 61850 standard series defines the core data model in IEC 61850-7-2, IEC 61850-7-3 and IEC 61850-7-4, which is also the data model related to Substations. Other domains (like DER, Hydro, Wind, etc.) could define their own data model based on IEC 61850 core data model to be able to use IEC 61850 core parts as a common layer.

**3.2**
**namespace**
identification of a data model of a specific part of the standard

Note 1 to entry: This definition will be used in this document when a specific data model has to be represented.

Note 2 to entry: The W3C consortium has also defined the concept of namespace for XML documents to identify the kind of elements which could be used in a file. This is typically the namespace used to identify an XML schema, as per IEC 61850-6 usage. This is not the definition used within this document.

## 4 Abbreviated terms

In general, the glossary and abbreviated terms defined in IEC 61850-2 apply. The following abbreviated terms are either exclusive to this document or particularly useful for understanding this document and are repeated here for convenience.

| | |
|---|---|
| CDC | Common Data Class |
| DO | Data Object |
| DA | Data Attribute |
| LN | Logical Node |
| NSD | Name Space Definition |
| SCL | System Configuration description Language |
| UML | Unified Modeling Language |
| XML | Extensible Markup Language |
| XSD | XML Schema Definition |

## 5 Use cases

### 5.1 Generation of SCL data type templates for system specification

#### 5.1.1 Description of the use case

##### 5.1.1.1 Name of use case

| ***Use case identification*** | | |
|---|---|---|
| ***ID*** | ***Domain(s)*** | ***Name of use case*** |
| | Administration | Use machine readable format to generate SCL data type templates for system specification |

#### 5.1.1.2 Narrative of use case

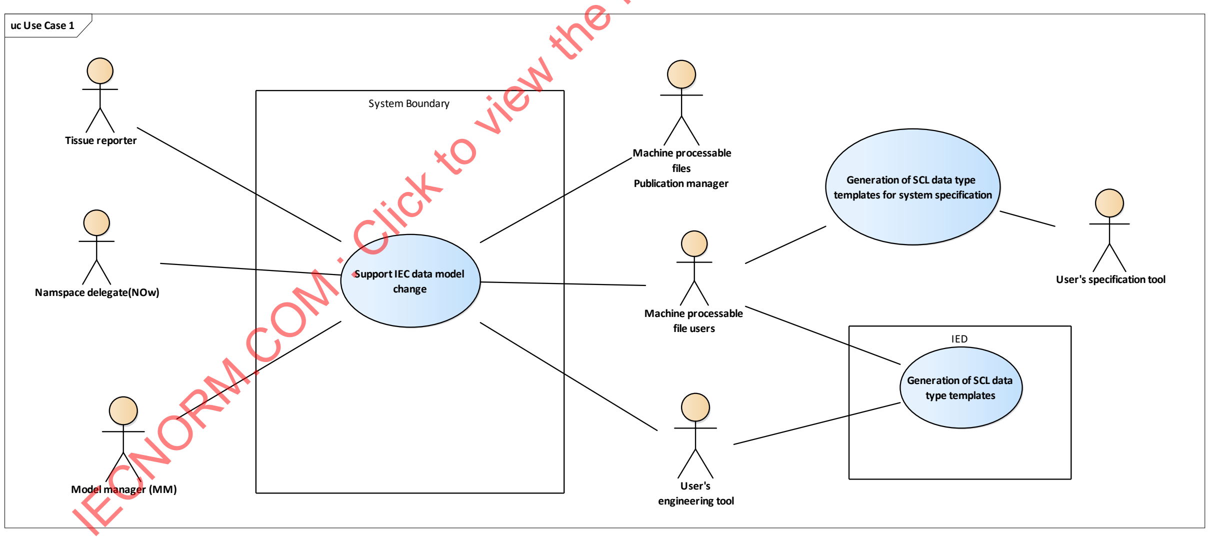
| ***Narrative of use case*** |
|---|
| ***Short description – max 3 sentences*** |
| User take the namespace NSD corresponding to its needs and create data type templates based on it. |
| ***Complete description*** |
| This use case describes how a Machine Processable file user can take benefit of the NSD publication to create datatype templates from the standard description and allow creation of an SSD for the specification of the process.<br>The Figure 1 presents the use case and the corresponding sequence diagram. |

#### 5.1.1.3 General remarks

| ***General remarks*** |
|---|
| The creation of a data model for a process specification shall follow rules from standard definitions of IEC 61850-7-2, IEC 61850-7-3, IEC 61850-7-4 or derived data models for other domains (like Wind farm with IEC 61400-25-2).<br>This creation is done manually by checking against the standard in paper format. This is source of errors.<br>Usage of NSD will allow for a tool to have the list of LNClass and CDCs available in a standard to allow user to use them as input to its definition. |

#### 5.1.1.4 Diagrams of use case

| ***Diagram of use case*** |
|---|
| **The primary use case** |


uc Use Case 1
Tissue reporter
Namspace delegate(NOw)
Model manager (MM)
System Boundary
Support IEC data model change
Machine processable files Publication manager
Machine processable file users
User's engineering tool
Generation of SCL data type templates for system specification
User's specification tool
IED
Generation of SCL data type templates

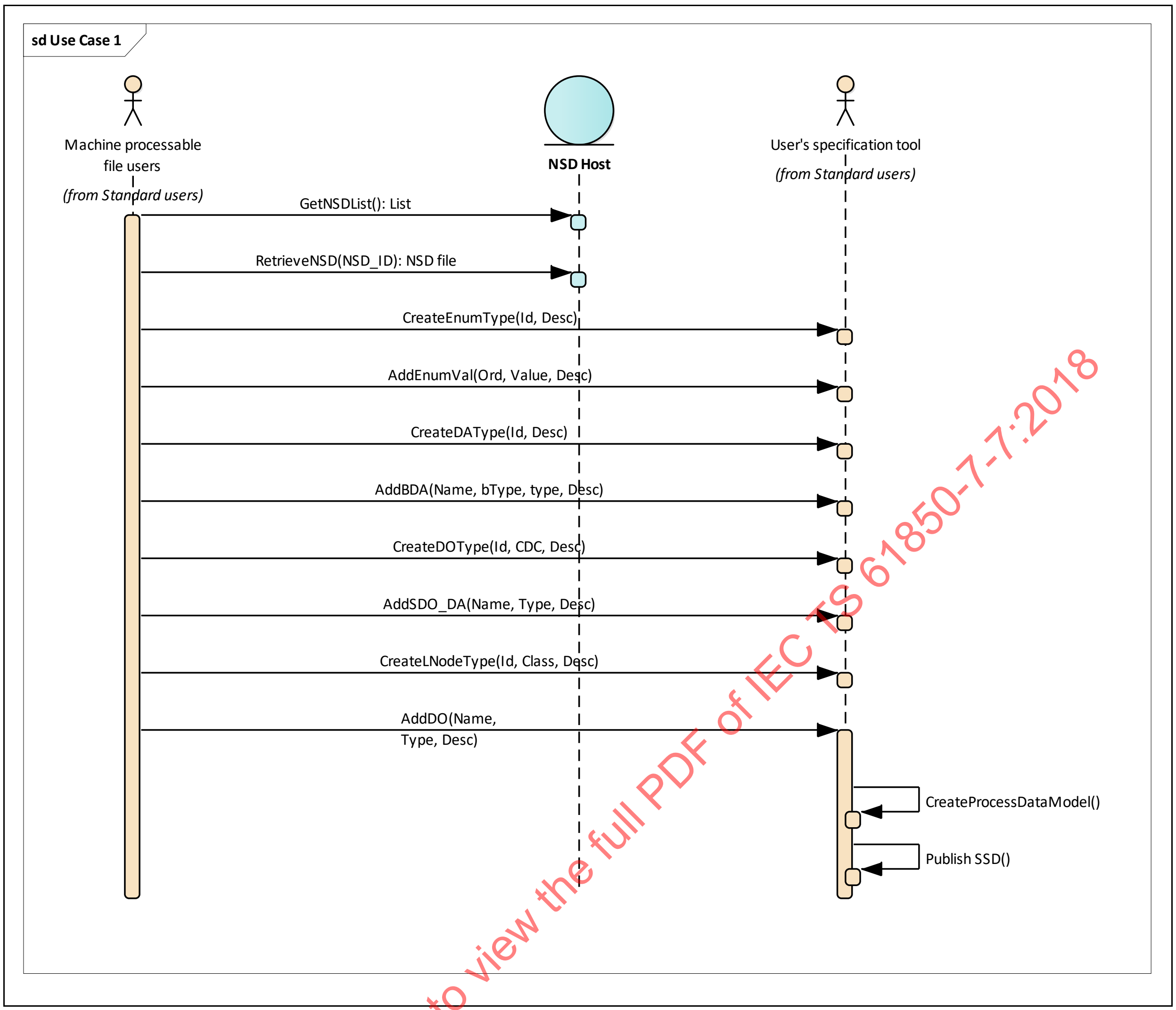

sd Use Case 1
Machine processable file users
(from Standard users)
NSD Host
User's specification tool
(from Standard users)
GetNSDList(): List
RetrieveNSD(NSD_ID): NSD file
CreateEnumType(Id, Desc)
AddEnumVal(Ord, Value, Desc)
CreateDAType(Id, Desc)
AddBDA(Name, bType, type, Desc)
CreateDOType(Id, CDC, Desc)
AddSDO_DA(Name, Type, Desc)
CreateLNodeType(Id, Class, Desc)
AddDO(Name, Type, Desc)
CreateProcessDataModel()
Publish SSD()

**Figure 1 – Generation of SCL data type templates for system specification diagrams**

### 5.1.2 Technical details

#### 5.1.2.1 Actors: People, systems, applications, databases, the power system, and other stakeholders

| ***Actors*** | | | |
|---|---|---|---|
| ***Grouping (community)*** | | ***Group description*** | |
| | | | |
| ***Actor name see actor list*** | ***Actor type see actor list*** | ***Actor description see actor list*** | ***Further information specific to this use case*** |
| **Machine readable file users** | people | user of machine readable file, involved in system engineering | |
| **User's engineering system** | system | engineering system of the Machine readable file users | IED specification tool |

#### 5.1.2.2 Preconditions, assumptions, post condition, events

| *Use case conditions* | | | |
|---|---|---|---|
| ***Actor/System/Information/Contract*** | ***Triggering event*** | ***Pre-conditions*** | ***Assumption*** |
| NSD Host | | | The latest NSD of the IEC 61850 namespaces are hosted on the NSD host. |
| | | | |

### 5.1.3 Information exchanged

| *Information exchanged* | | |
|---|---|---|
| ***Name of information exchanged*** | ***Description of information exchanged*** | ***Requirements to information data R-ID*** |
| NSD_ID | Identifier of an NSD, based on the namespace identifier based on the publication number, version and revision | |
| NSD list | List of available NSD on the NSD Host with a description | |
| NSD file | The file representing a Namespace data model to be used by a tool | |

## 5.2 Generation of SCL data type templates for ICD files

### 5.2.1 Description of the use case

#### 5.2.1.1 Name of use case

| *Use case identification* | | |
|---|---|---|
| ***ID*** | ***Domain(s)*** | ***Name of use case*** |
| | Administration | Use machine readable format to generate SCL data type templates for ICD files |

#### 5.2.1.2 Narrative of use case

| *Narrative of use case* |
|---|
| ***Short description – max 3 sentences*** |
| User take the namespace NSD corresponding to its needs and create data type templates based on it. |
| ***Complete description*** |
| This use case describes how a Machine Processable file user can take benefit of the NSD publication to create datatype templates from the standard description and allow creation of an ICD for a specific IED.<br>The Figure 2 presents the use case and the corresponding sequence diagram. |

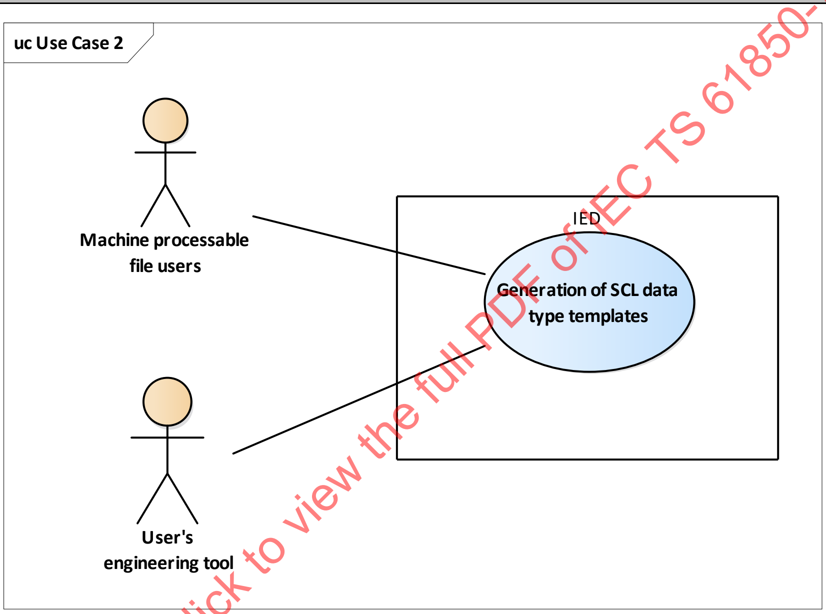
#### 5.2.1.3 General remarks

| ***General remarks*** |
|---|
| The creation of a data model for an implementation in a device shall follow rules from the standard definition of IEC 61850-7-2, IEC 61850-7-3, IEC 61850-7-4 or derived data models for other domains (like Wind farm with IEC 61400-25-2).<br>This creation is done manually by checking against the standard in paper format. This is source of errors.<br>Usage of NSD will allow for a tool to have the list of LNClass and CDCs available in a standard to allow user to use them as input to its definition. |

#### 5.2.1.4 Diagrams of use case

***Diagram of use case***

**The primary use case**


uc Use Case 2
Machine processable file users
User's engineering tool
IED
Generation of SCL data type templates

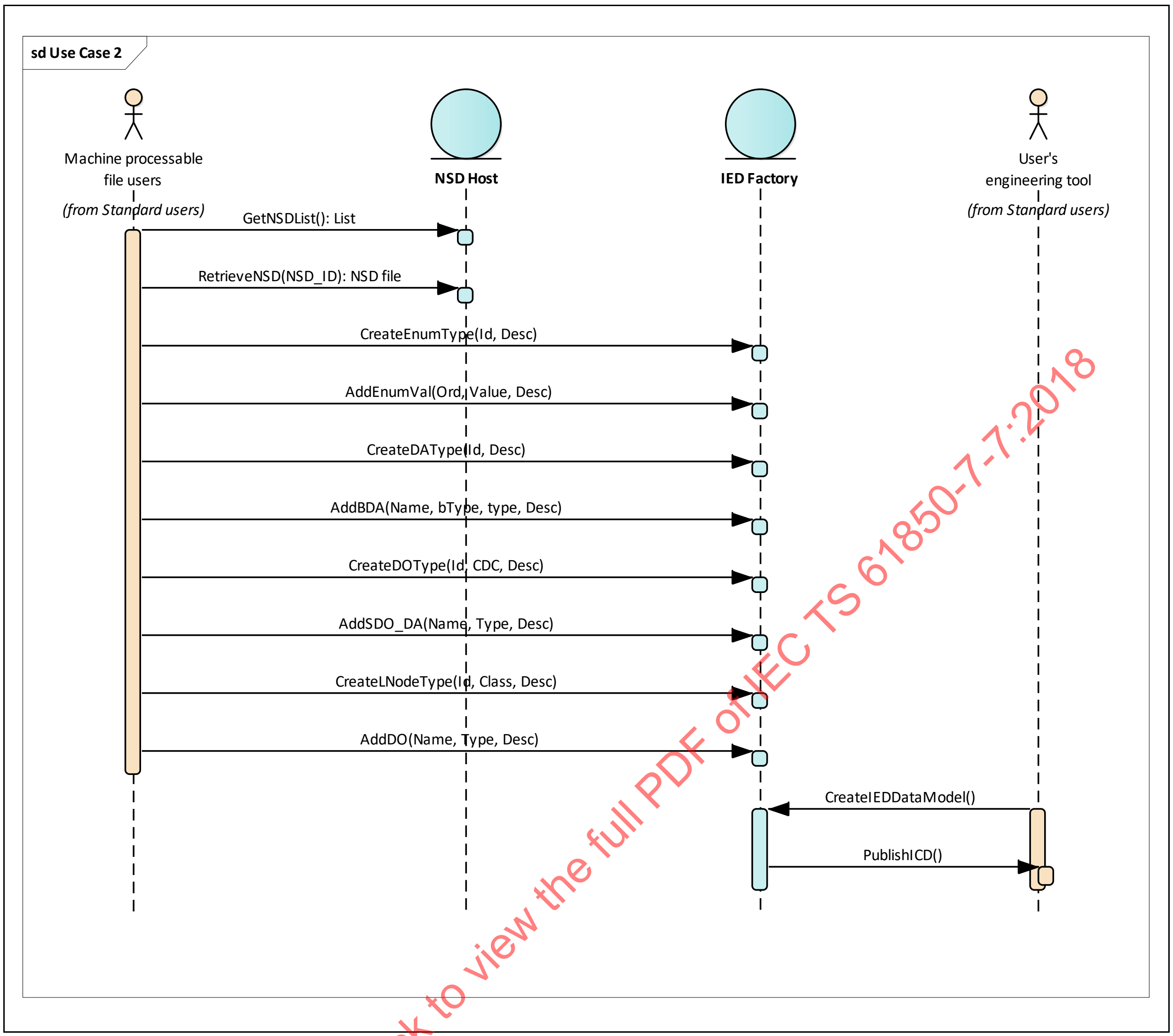


**Figure 2 – Generation of SCL data type templates for ICD files diagrams**

### 5.2.2 Technical details

#### 5.2.2.1 Actors: People, systems, applications, databases, the power system, and other stakeholders

| ***Actors*** | | | |
|---|---|---|---|
| ***Grouping (community)*** | | ***Group description*** | |
| | | | |
| ***Actor name see Actor list*** | ***Actor type see Actor list*** | ***Actor description see Actor list*** | ***Further information specific to this use case*** |
| **Machine readable file users** | people | user of machine readable file, involved in system engineering | |
| **User's engineering system** | system | engineering system of the Machine readable file users | IED engineering tool |

#### 5.2.2.2 Preconditions, assumptions, post condition, events

| *Use case conditions* | | | |
|---|---|---|---|
| ***Actor/System/Information/Contract*** | ***Triggering event*** | ***Pre-conditions*** | ***Assumption*** |
| NSD Host | | | The latest NSD of the IEC 61850 namespaces are hosted on the NSD host. |

### 5.2.3 Information exchanged

| *Information exchanged* | | |
|---|---|---|
| ***Name of information exchanged*** | ***Description of information exchanged*** | ***Requirements to information data R-ID*** |
| NSD_ID | Identifier of an NSD, based on the namespace identifier based on the publication number, version and revision | |
| NSD list | List of available NSD on the NSD Host with a description | |
| NSD file | The file representing a Namespace data model to be used by a tool | |

## 5.3 Validation of data model conformity

### 5.3.1 Description of the use case

#### 5.3.1.1 Name of use case

| *Use case identification* | | |
|---|---|---|
| ***ID*** | ***Domain(s)*** | ***Name of use case*** |
| | Administration | Use machine readable format to validate conformity of a data model to the standard |

#### 5.3.1.2 Narrative of use case

| *Narrative of use case* |
|---|
| ***Short description – max 3 sentences*** |
| User takes the namespace NSD and SCL schema to validate syntactically and semantically a data model |
| ***Complete description*** |
| This use case describes how a Machine Processable file user can take benefit of the NSD publication to validate the conformity of a given Data Model created with a IED Configuration Tool or manually edited.<br>The validation process includes<br>– Syntax validation of the SCL file against the IEC 61850-6 schema<br>– A semantic validation<br>For instance:<br>– Is the description of an LNodeType conforms to the LN NSD content?<br>– Are all mandatory DOs (resp. DAs) present in a LNodeType (resp. DOType)?<br>– Etc.<br>The Figure 3 presents the use case and the corresponding sequence diagram. |

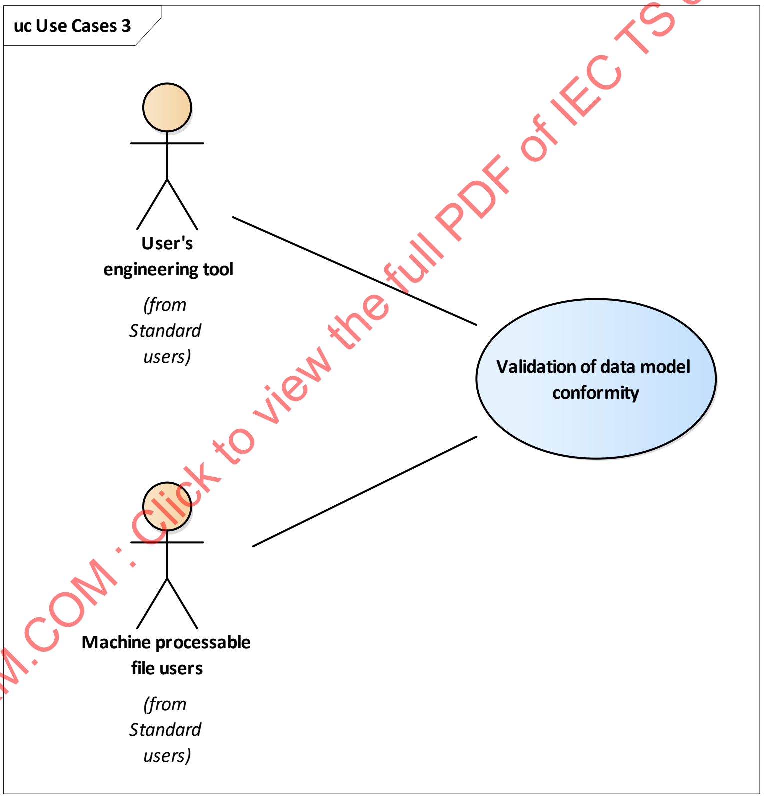
#### 5.3.1.3 General remarks

| ***General remarks*** |
|---|
| The existing data model to be validated shall follow rules from standard definition of IEC 61850-7-2, IEC 61850-7-3, IEC 61850-7-4 or derived data models for other domains (like Wind farm with IEC 61400-25-2).<br><br>This creation is done either by use of a configuration tool or done manually by checking against the standard in paper format. It can be optionally manually edited to solve some found issues. In any circumstances this process is prone to errors and the final result should be checked to verify its conformity.<br><br>The syntactic conformity is checked thanks to the IEC 61850-6 schema. To check the semantic conformity (excluding private extensions) the validation tool will make use of the NSD files. |

### 5.3.2 Diagrams of use case

***Diagram of use case***

**The primary use case**



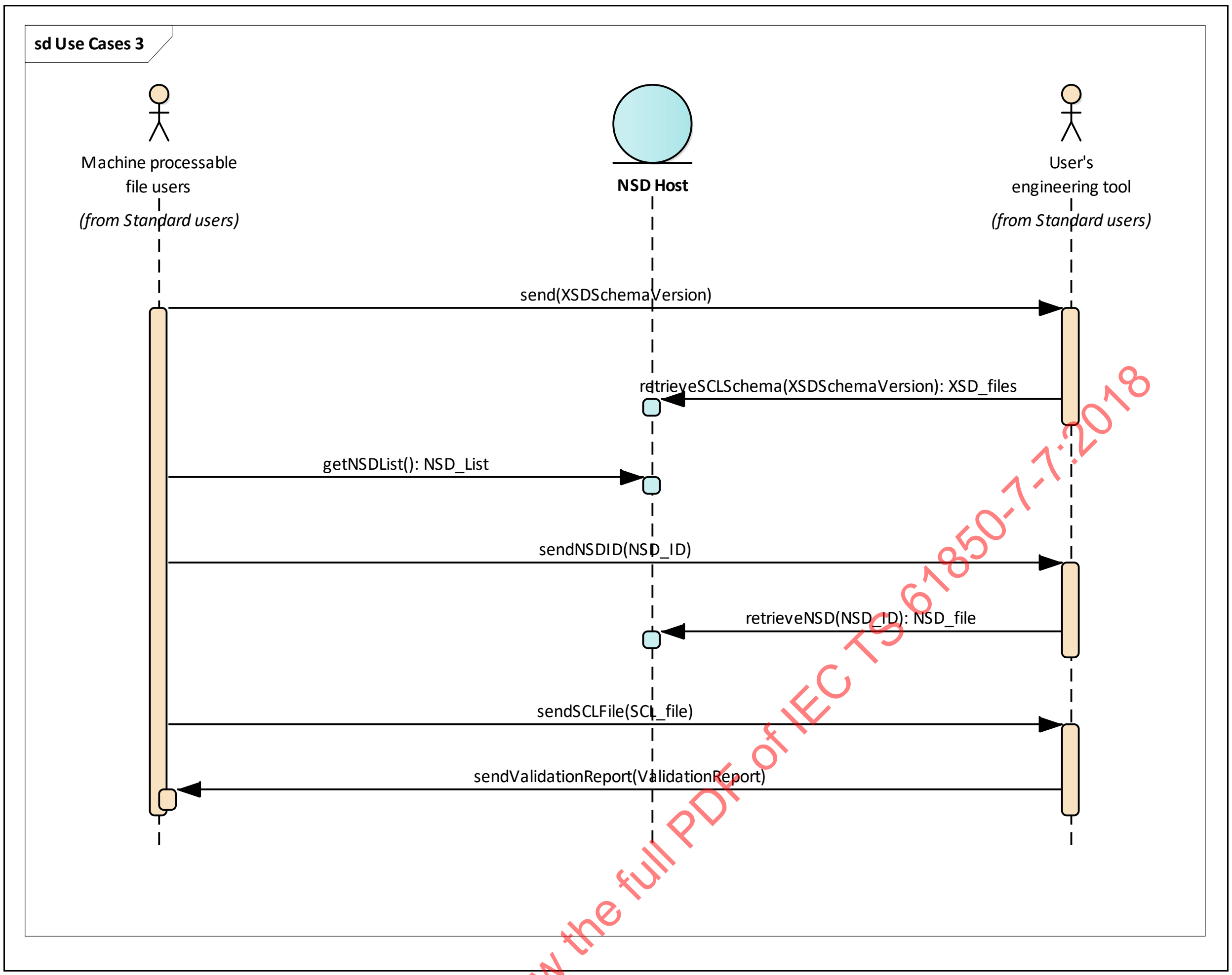


**Figure 3 – Validation of data model conformity diagrams**

### 5.3.3 Technical details

#### 5.3.3.1 Actors: People, systems, applications, databases, the power system, and other stakeholders

| ***Actors*** | | | |
|---|---|---|---|
| ***Grouping (community)*** | | ***Group description*** | |
| | | | |
| ***Actor name see Actor list*** | ***Actor type see Actor list*** | ***Actor description see Actor list*** | ***Further information specific to this use case*** |
| **Machine processable file users** | people | user of machine readable file, involved in system engineering | |
| **User’s engineering tool** | system | engineering system of the Machine processable file users | IED Conformity Validation tool |

#### 5.3.3.2 Preconditions, assumptions, post condition, events

| *Use case conditions* | | | |
|---|---|---|---|
| ***Actor/System/Information/Contract*** | ***Triggering event*** | ***Pre-conditions*** | ***Assumption*** |
| NSD Host | | | The latest NSD of the IEC 61850 namespaces are hosted on the NSD host. And the latest SCL Schema |
| User's engineering tool | | | For validation process it is assumed that all the validation semantic rules are documented |

#### 5.3.3.3 Information exchanged

| *Information exchanged* | | |
|---|---|---|
| ***Name of information exchanged*** | ***Description of information exchanged*** | ***Requirements to information data R-ID*** |
| NSD_ID | Identifier of an NSD, based on the namespace identifier based on the publication number, version and revision | |
| NSD_list | List of available NSD on the NSD Host with a description | |
| NSD_file | The file representing a Namespace data model to be used by a tool | |
| XSDSchemaVersion | The last version of WG10 7-6 SCL XSD Schema | |
| XSD_files | The XSD SCL Files | As provided by IEC<br>If allows to publish a syntactic validation in the validation report |
| SCL_file | The SCL file to be validated | |
| ValidationReport | The file which is the result of the validation process of the DataModelFile | All the validations according to documented specified rules |

## 5.4 Definition of private data model extensions by following the rules of the standard

### 5.4.1 Description of the use case

#### 5.4.1.1 Name of use case

| *Use Case identification* | | |
|---|---|---|
| ***ID*** | ***Domain(s)*** | ***Name of use case*** |
| | Administration | Use machine readable format to create extension namespace for a manufacturer specific data model |

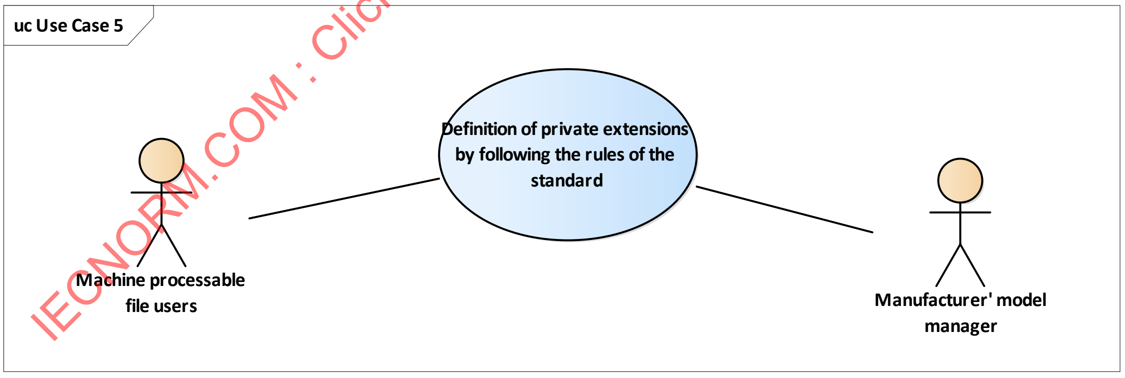
### 5.4.1.2 Narrative of use case

| ***Narrative of use case*** |
|---|
| ***Short description – max 3 sentences*** |
| User takes the namespace NSD corresponding to its needs and extend existing LNClass, or create new LNClass. Then user is able to add new DataObject based on standard CDC. |
| ***Complete description*** |
| This use case describes how a manufacturer model manager can take benefit of the NSD publication to create a private extension of a standard namespace by extending standard LNClass or by creating new LNClass.<br>For extended or new LNClass, user will create new DataObject based on standard CDC.<br>User is also able to create private enumeration for new DataObject.<br>Then user will be able to define its own namespace ID and create the NSD file for its private extension, known as private namespace.<br>The Figure 4 presents the use case and the corresponding sequence diagram. |

### 5.4.1.3 General remarks

| ***General remarks*** |
|---|
| A private namespace is an NSD file used to represent specific data model of a manufacturer device.<br>The creation of a private namespace shall follow rules from standard definition of IEC 61850-7-1. A private namespace is allowed only to extend existing LNClass or create new LNClass (with a different name than existing LNclass) and add DataObject to them. Creations or modifications of CDC, ConstructedAttribue or BasicTypes are forbidden. Enumerations can be created for DataObjects added into private namespace.<br>A new DataObject shall not reuse an existing DataObject name if semantic is different. standard NSD will help in searching such cases. |

### 5.4.2 Diagrams of use case

| ***Diagram of use case*** |
|---|
| **The primary use case** |
|  uc Use Case 5<br>Definition of private extensions by following the rules of the standard<br>Machine processable file users<br>Manufacturer' model manager |

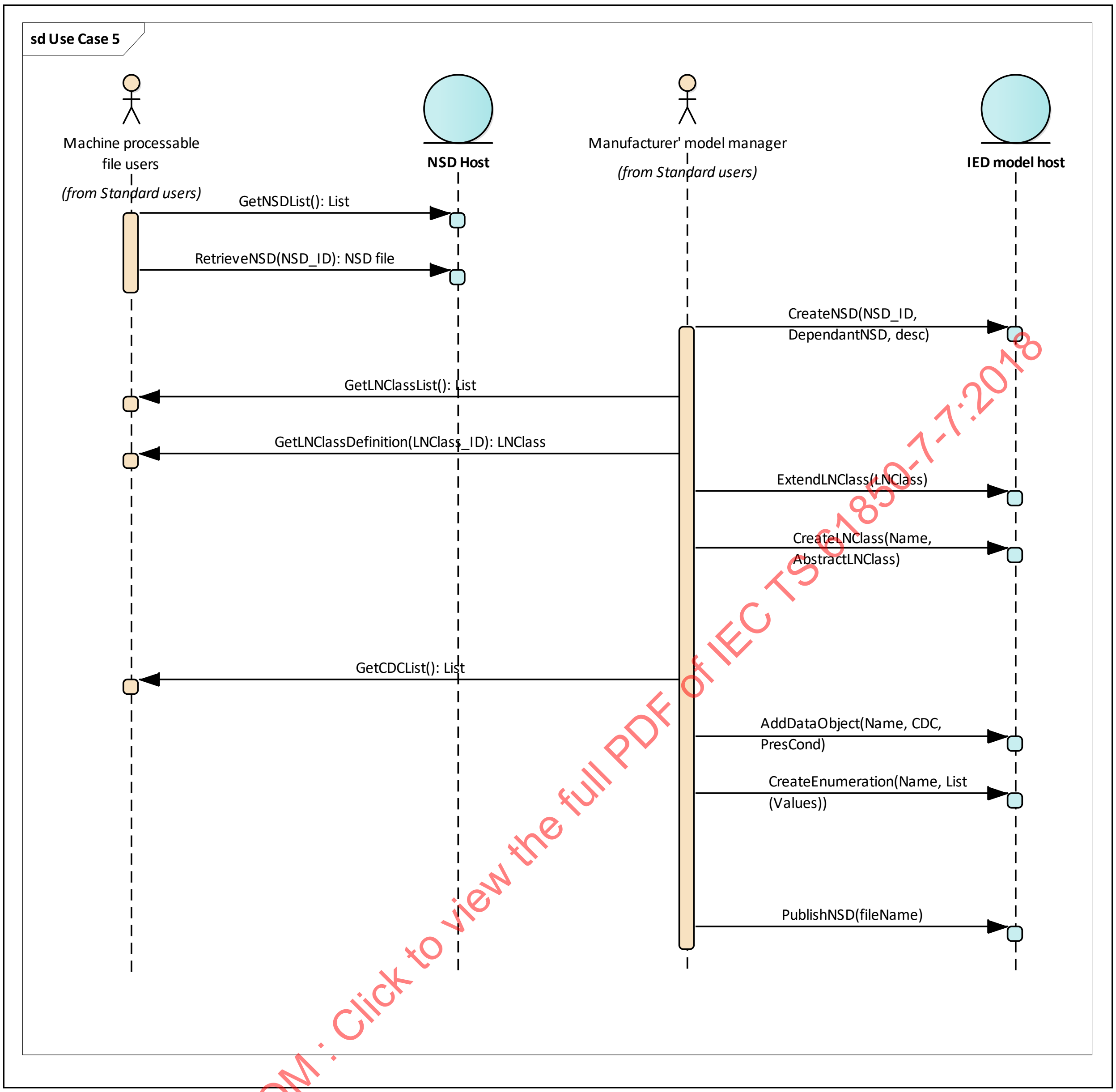

sd Use Case 5
Machine processable file users
(from Standard users)
NSD Host
Manufacturer' model manager
(from Standard users)
IED model host
GetNSDList(): List
RetrieveNSD(NSD_ID): NSD file
CreateNSD(NSD_ID, DependantNSD, desc)
GetLNClassList(): List
GetLNClassDefinition(LNClass_ID): LNClass
ExtendLNClass(LNClass)
CreateLNClass(Name, AbstractLNClass)
GetCDCList(): List
AddDataObject(Name, CDC, PresCond)
CreateEnumeration(Name, List (Values))
PublishNSD(fileName)

**Figure 4 – Definition of private data model extensions by following the rules of the standard diagrams**

### 5.4.3 Technical details

#### 5.4.3.1 Actors: People, systems, applications, databases, the power system, and other stakeholders

| *Actors* | | | |
|---|---|---|---|
| *Grouping (community)* | | *Group description* | |
| | | | |
| *Actor name see Actor list* | *Actor type see Actor list* | *Actor description see Actor list* | *Further information specific to this use case* |
| **Machine readable file users** | people | user of machine readable file, involved in system engineering | |
| **Manufacturer's model manager** | people | User responsible of definition of the private namespace with control of coherency with NSD | |

#### 5.4.3.2 Preconditions, assumptions, post condition, events

| *Use case conditions* | | | |
|---|---|---|---|
| ***Actor/System/Information/Contract*** | ***Triggering event*** | ***Pre-conditions*** | ***Assumption*** |
| NSD Host | | | The latest NSD of the IEC 61850 namespaces are hosted on the NSD host. |
| IED model host | | | Repository for manufacturer's NSD |

### 5.4.4 Information exchanged

| *Information exchanged* | | |
|---|---|---|
| ***Name of information exchanged*** | ***Description of information exchanged*** | ***Requirements to information data R-ID*** |
| NSD_ID | Identifier of an NSD, based on the namespace identifier based on the publication number, version and revision | |
| NSD list | List of available NSD on the NSD Host with a description | |
| NSD file | The file representing a Namespace data model to be used by a tool | |
| LNClass_ID | Name of an existing LNClass from an NSD | |
| LNClass list | List of LNClass name existing in an NSD | |
| LNClass | Description of an LNClass | |
| CDC list | List of CDC from an NSD | |

## 5.5 Supporting the IEC 61850 data model change management

### 5.5.1 Description of the use case

#### 5.5.1.1 Name of use case

| *Use case identification* | | |
|---|---|---|
| ***ID*** | ***Domain(s)*** | ***Name of use case*** |
| | Administration | Use machine readable format as a support to reflect Namespace content modifications and Tissue process consolidated impacts |

#### 5.5.1.2 Narrative of use case

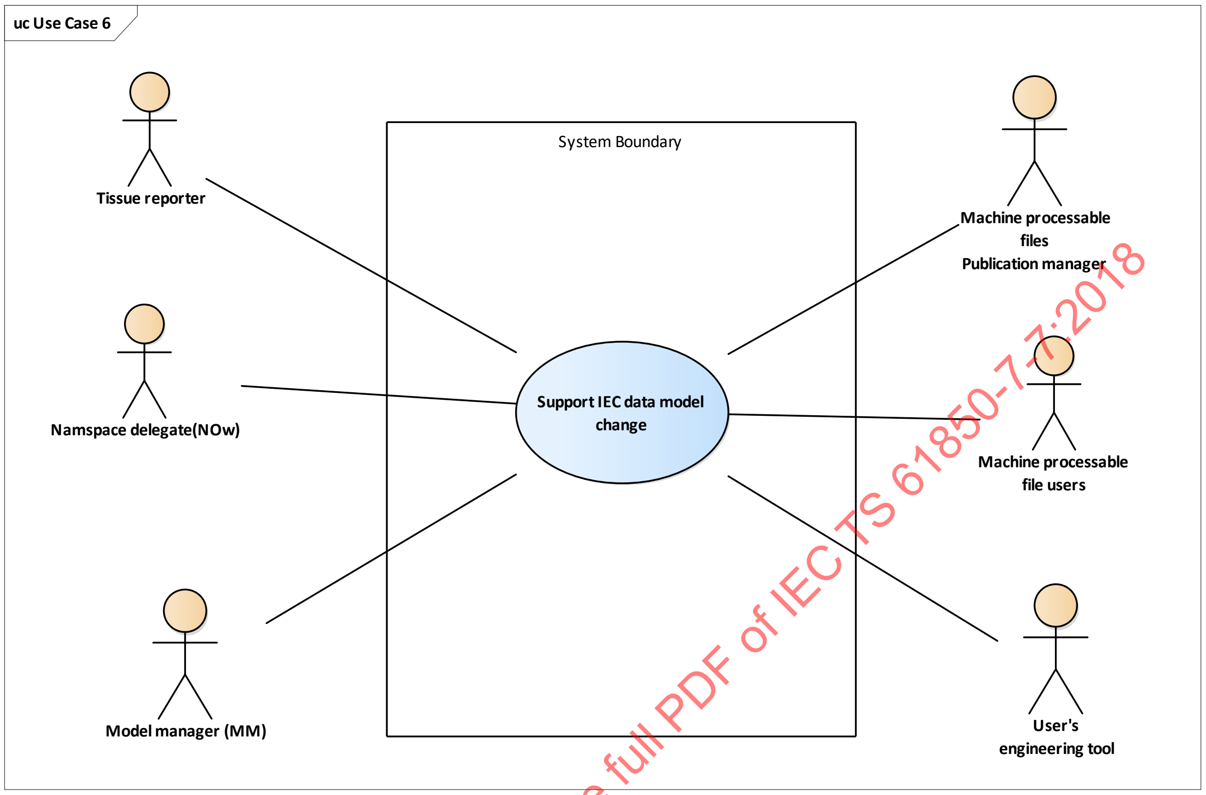
| *Narrative of use case* |
|---|
| ***Short description – max 3 sentences*** |
| Tissue reporters issue Tissue (Tissue: "technical issue which can affect interoperability and/or the published standard document") which at the end lead to the production of a new release of machine readable format |
| ***Complete description*** |
| This Use Case describe the path between Tissues as reported by Tissues reporters, the UML model where all impact to the model are reflected, the published machine processable file, and finally the usage by standard user of the updated machine processable file. The Figure 5 presents the use case and the corresponding sequence diagram. |

#### 5.5.1.3 General remarks

| ***General remarks*** |
|---|
| The Management of corrections of the published standard is done through the Tissue process (refer to the IEC 61850 part 1 for further explanations)<br><br>Tissues may or may not have an impact on the data model, and therefore in case the model is impacted, it shall be reflected within the UML model.<br><br>Each Tissue follows its own resolution process, and reports on its status (Red, Orange, Green, Blue). This is not the purpose of this use case to describe it, however it is a key pre-process to consider for the use case. That’s why only a simplified description is provided here.<br><br>Under the decision of the namespace delegate, green Tissues can be packed together as a single namespace update pack, which will go through a validation process. Then the namespace delegate may ask to publish a new release of machine processable files, which would launch a publication process thread.<br><br>The NSD publication manager will be requested to make public the considered pack of NSD files associated to a namespace and a namespace version.<br><br>Once publically available, notification may be sent to standards users, in order to indicate the availability of a new release.<br><br>The user may decide to upload the corresponding new release.<br><br>It is envisaged that such notification and upload are directly managed by machine to machine information exchange. |

### 5.5.2 Diagrams of use case

| ***Diagram of use case*** |
|---|
| **The primary use case** |



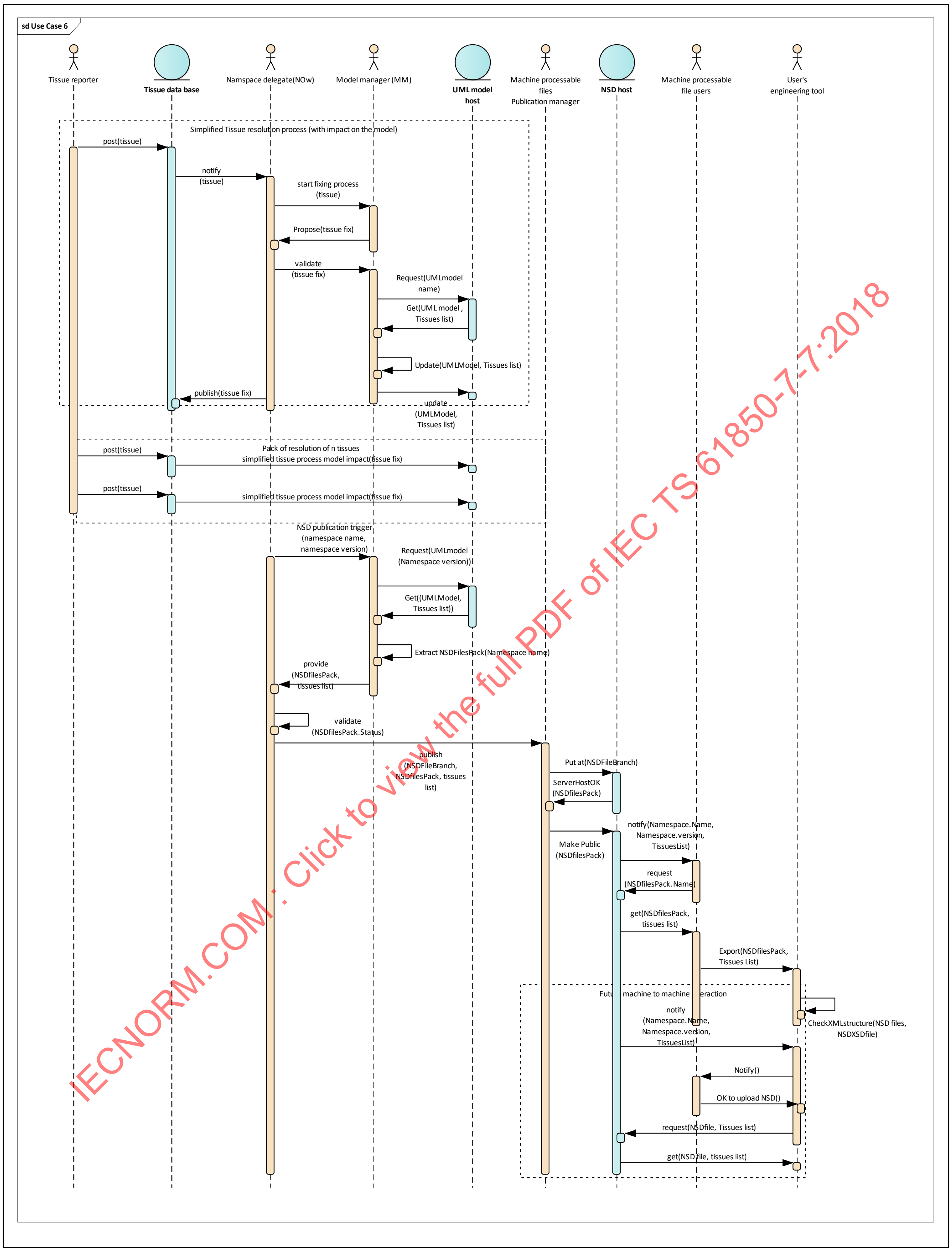

sd Use Case 6
Tissue reporter
Tissue data base
Namspace delegate(NOw)
Model manager (MM)
UML model host
Machine processable files Publication manager
NSD host
Machine processable file users
User's engineering tool
Simplified Tissue resolution process (with impact on the model)
post(tissue)
notify (tissue)
start fixing process (tissue)
Propose(tissue fix)
validate (tissue fix)
Request(UMLmodel name)
Get(UML model , Tissues list)
Update(UMLModel, Tissues list)
publish(tissue fix)
update (UMLModel, Tissues list)
Pack of resolution of n tissues
post(tissue)
simplified tissue process model impact(tissue fix)
post(tissue)
simplified tissue process model impact(tissue fix)
NSD publication trigger (namespace name, namespace version)
Request(UMLmodel (Namespace version))
Get((UMLModel, Tissues list))
Extract NSDFilesPack(Namespace name)
provide (NSDfilesPack, tissues list)
validate (NSDfilesPack.Status)
publish (NSDFileBranch, NSDfilesPack, tissues list)
Put at(NSDFileBranch)
ServerHostOK (NSDfilesPack)
Make Public (NSDfilesPack)
notify(Namespace.Name, Namespace.version, TissuesList)
request (NSDfilesPack.Name)
get(NSDfilesPack, tissues list)
Export(NSDfilesPack, Tissues List)
Future machine to machine interaction
CheckXMLstructure(NSD files, NSDXSDfile)
notify (Namespace.Name, Namespace.version, TissuesList)
Notify()
OK to upload NSD()
request(NSDfile, Tissues list)
get(NSDfile, tissues list)

**Figure 5 – Supporting the IEC 61850 data model change management**

### 5.5.3 Technical details

#### 5.5.3.1 Actors: People, systems, applications, databases, the power system, and other stakeholders

| *Actors* | | | |
|---|---|---|---|
| *Grouping (community)* | | *Group description* | |
| | | | |
| *Actor name see Actor list* | *Actor type see Actor list* | *Actor Description see Actor list* | *Further information specific to this use case* |
| **Model manager** | people | Model manager (MM) of the standard models. A UML specialist in charge of maintaining the overall consistency of the UML model of one or many IEC 61850 namespaces and who is part of the WG as well as the 61850 UML *Modeling Team* in charge of managing model. | |
| **Namespace delegate (NOw)** | people | each namespace is under the responsibility of one Namespace delegate (NOw), by delegation of the TC secretary or WG convenor (quite often, this is the convenor himself). Each delegate is in charge of validating the content of the namespace (as well as the NSD representation of the content), in co-ordination with the other delegates of namespace interacting with his own one. Namespace delegates are members of TC57 WGs but can also be members of other TC of the IEC. | |
| **TISSUE reporter** | people | expresses a feedback to the standard editorial team of one part of the standard about a technical issue in the published content – may be anybody who has logged into the Tissue database | |
| **Machine readable file users** | people | user of machine readable file, involved in system engineering | |
| **User's engineering system** | system | engineering system of the Machine readable file users | |
| **NSD Publication manager (PM)** | people | Create/update "public" content from the publishing artifacts (NSD files) generated from the UML model referenced master. Publication responsibility is under the IEC Central Office leadership. | |

#### 5.5.3.2 Preconditions, assumptions, post condition, events

| ***Use case conditions*** | | | |
|---|---|---|---|
| ***Actor/System/Information/Contract*** | ***Triggering event*** | ***Pre-conditions*** | ***Assumption*** |
| Model manager, UML Model | | | The latest UML model of the IEC 61850 namespaces (all packages merged into one UML model) is hosted on the UML model host. |

### 5.5.4 Information exchanged

| ***Information exchanged*** | | |
|---|---|---|
| ***Name of information exchanged*** | ***Description of information exchanged*** | ***Requirements to information data R-ID*** |
| UMLModelName | Unique name of the model which hosts the considered namespace | |
| NSD files branch | URL Branch of the tree where the NSD files pack needs to be attached to. This is needed to manage properly NSD file versioning.<br>A branch will contain all subsequent releases of the NSD files of the same namespace. | uniquely derived from the explicit reference to the UML model version used for producing the content<br>explicit reference of the tools and the version of tools used for producing the content |
| NSDXSDfile | XSD file enabling the check of the NSD file structure | URL is hosted by IEC.<br>XSD file should be available at the URL indicated in the XML NSD file |
| NSD files pack | pack of files reflecting the content of an IEC 61850 namespace at a given time. Seems to comprise at least 3 separated files:<br>– NSDXSDfile<br>– The data model structure of the considered namespace<br>– The textual elements of the same namespace<br>– The formal description of the relationships between this namespace and the other namespaces | Explicit namespace ID including explicit unique versioning information.<br><br>Explicit reference to the UML model version used for producing the content<br>explicit reference of the tools and the version of tools used for producing the content, to allow the tracking of changes of the associated source, and of the transformation processes.<br><br>Reference to textual element (possibly hosted in a separated file) shall be persistent, shall not change for a given object, to allow the tracking of any changes, only the real changes. |
| NSDfilesPacks.status | A status indicates the status of the pack content:<br>– Draft<br>– Reviewed<br>– Final | |

| ***Information exchanged*** | | |
|---|---|---|
| ***Name of information exchanged*** | ***Description of information exchanged*** | ***Requirements to information data R-ID*** |
| Tissues list | incremental list of tissues having been considered for producing the content of the considered NSD file, considering as the reference, a named previous version of NSD file | the basis from which the increment is built shall be explicitly given |
| Namespace.Name Namespace.Version | Attribute of NSD files which should be made public for identification and search | |

### 5.5.5 Common terms and definitions

| ***Common terms and definitions*** | |
|---|---|
| ***Term*** | ***Definition*** |
| Tissue | “technical issue which can affect interoperability and/or the published standard document” |

## 6 Namespace file breakdown

### 6.1 General

This document defines one namespace per file, and an additional file expressing the links between all these.

In this way, there will be one namespace file for each core part of the IEC 61850 series, for extensions and other domains.

### 6.2 Namespace dependencies

The dependency between namespaces is also described in machine-processable format, allowing defining a kind of usage between each part. This dependency is used to inherit definition from a namespace into another one to avoid duplication of information. Figure 6 presents an exemple of namespace inheritance.

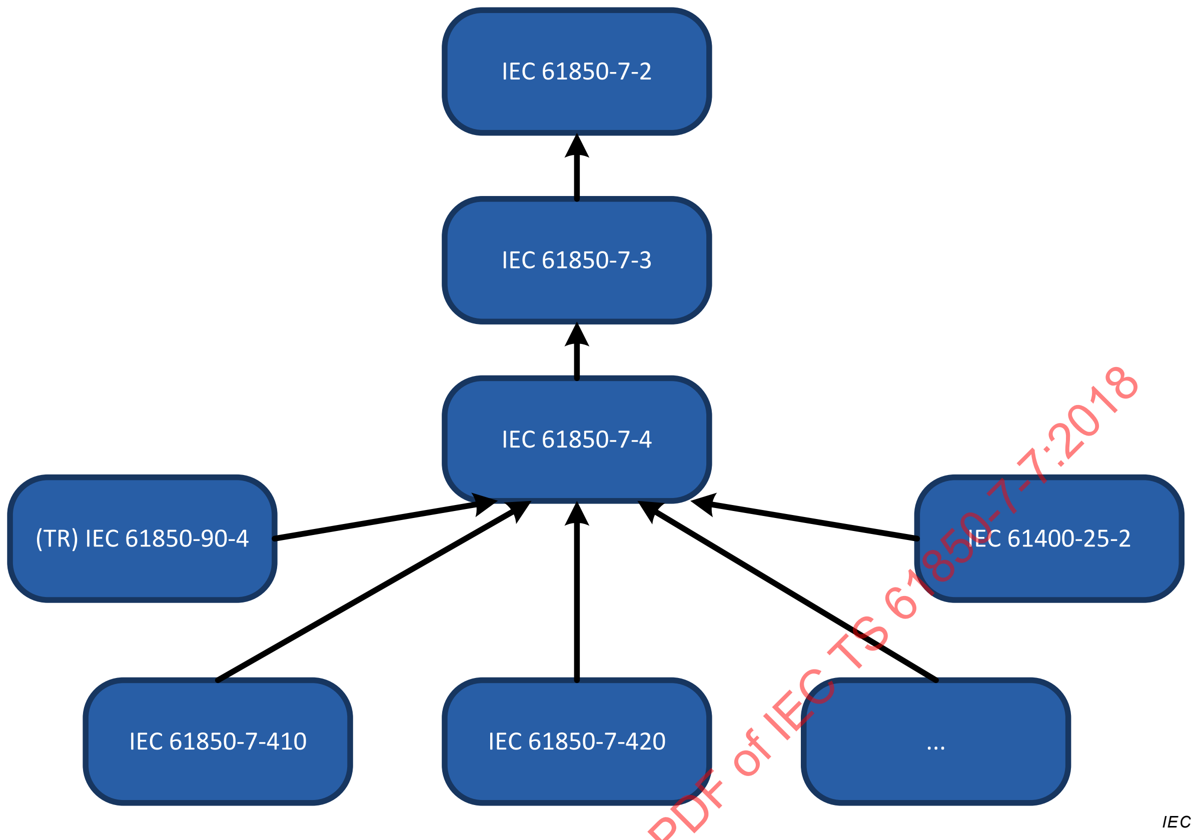


**Figure 6 – Namespace dependencies**

Inheritance between namespaces allows usage of objects defined in another namespace. All objects defined in namespaces are identified by a unique name for the type of object and reference is done using this name. It is not allowed to define same name twice for a given type of object (except for extension).

The other kind of link between namespaces is the service usage, to define which kind of service implementation could be used by other namespaces. This link is not represented by a dependency as it is not one-to-one link.

The goal of this association is to be able to define service namespaces (MMS, WebServices, etc.) independently from other namespaces and identify usage of service namespaces by other namespaces, using an association file. Figure 7 present usage links between namespaces and service namespaces.

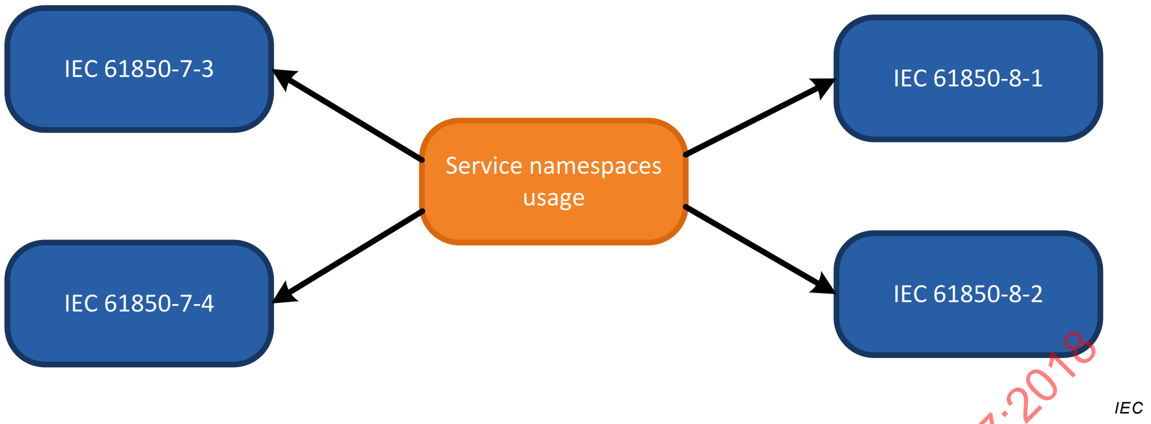


**Figure 7 – Service Namespace usage**

## 6.3 Namespace types

### 6.3.1 General

Namespaces are used to define data model for specific usage. Different types of namespaces can be identified, depending on this usage.

### 6.3.2 Core namespace

Core namespaces are related to original data model from IEC 61850. This covers: IEC 61850-7-2, IEC 61850-7-3 and IEC 61850-7-4.

These namespaces define bricks for interoperability, covering basic and constructed types, functional constraints and presence condition, common data classes and common logical node classes.

IEC 61850-7-4 also covers substation automation domain logical node classes.

Only core namespaces are allowed to define bricks for interoperability. This means that no other namespace will be allowed to contain its own definition of basic and constructed types, functional constraints, presence conditions and CDCs.

### 6.3.3 Domain namespace

A domain namespace defines logical node classes related to a specific domain different from substation automation. Such namespace could be part of IEC 61850 or any other IEC standard.

Domain namespaces cover for example hydroelectric power plants (IEC 61850-7-410), distributed energy resources (IEC 61850-7-420) or wind power plants (IEC 61400-25).

### 6.3.4 Technical report namespace

A technical report namespace defines logical node classes for technical work. They are transitional before integration into a future edition of the related standard.

This is the only category of namespace other than core which may include new CDCs as a transitional process before integration into the core namespace.

Technical report namespaces cover for example condition monitoring diagnosis and analysis (IEC TR 61850-90-3) or network engineering (IEC TR 61850-90-4).

### 6.3.5 Private namespace

A private namespace defines logical node classes for a usage outside standardization. This could cover either manufacturer or user needs.

A manufacturer will be able with a private namespace to define the extension to the standard integrated in their devices.

A user will be able with a private namespace to define their needs in term of data and functions not present in the standard. This will be useful for a customer to specify to a manufacturer specific data required in his applications, or to model a proposal for evolution to a standard.

## 7 Overview of the format

### 7.1 General

Like SCL format (defined in IEC 61850-6), the machine-processable format is based on Extensible Markup Language (XML) standard de represent namespaces. The format itself is call Namespace Definition (NSD).

A grammar has been written to verify content of such format, using XML Schema Definition (XSD) which is interpretable by common XML processing tools to automatically validate NSD files.

The character encoding used for NSD is UTF-8.

To fulfil the different use cases, the NSD format has to present different kind of information:

- Definition of data model namespaces as per definition of related standard (like IEC 61850-7-4).
- Definition of service namespaces as per definition of related standard (like IEC 61850-8-1).
- Definition of service namespace usage by data model namespaces
- Definition of translatable documentation for each namespace

These four kinds of information are split in different file types, for which the content is defined in one single XSD: NSD.xsd, given in Annex A.

In Subclause 7.2, each type will be presented with its specific root, the global architecture, the philosophy and the concepts of the file. For the detailed description of each element and attribute possible in the file, the XSD is fully documented with detailed description. These details will not be reproduced in this document, except in annexes.

### 7.2 File types

#### 7.2.1 NSD

##### 7.2.1.1 General

The first type of file is NSD which represent the data model namespace as defined in IEC 61850-7-x or other domain related standards.

One file represents one namespace, for core namespaces (IEC 61850-7-2, IEC 61850-7-3, IEC 61850-7-4), for other domain namespaces (IEC 61850-7-410, IEC 61850-7-420, IEC 61400-25-2, etc.) and for technical report namespaces (IEC TR 61850-90-3, IEC TR 61850-90-4, etc.).

The NSD file contains the formal description of the namespace's data model, with definition of all hierarchical elements used in the data model. Each element will have different attributes to characterize it, like a name or an identifier, and other kind of attributes used to associate documentation to the element.

The documentation defined in this kind of file is a unique identifier used in the dedicated documentation file type. No textual documentation will be defined in the NSD file.

### 7.2.1.2 Root element

The NSD file has the root element NS to identify the namespace represented with the ID, version revision and release (both are optional):

```
<NS id="IEC 61850-7-4" version="2007" revision="B" umlDate="2016-04-29T12:00:00Z">
```

The umlDate indicates when the source of this namespace has been updated. It is informative and optional.

### 7.2.1.3 Version information elements

The root element contains two elements allowing description of additional information related to version of the namespace.

The first element, Changes, identifies the previous version of the namespace and the tissues which has been resolved in the current version:

```
<Changes version="2007" revision="A" tissues="650, 671"/>
```

The second element, DependsOn, defines another namespace which is used as a base to produce current namespace (like IEC 61850-7-3 defines elements used by IEC 61850-7-4):

```
<DependsOn id="IEC 61850-7-3" version="2007" revision="B"/>
```

### 7.2.1.4 Functional constraints and presence condition elements

The root element then contains two specific elements to describe data related to the services.

The first element is FunctionalConstraints which contains the functional constraints as per the definition in IEC 61850-7-2, with the services applicable to each of them:

```
<FunctionalConstraints>
    <FunctionalConstraint abbreviation="ST" titleID="IEC 61850-7-2:FunctionalConstraints.ST.title">
        <ApplicableServices>
            <Service name="GetDataValues"/>
        </ApplicableServices>
    </FunctionalConstraint>
</FunctionalConstraints>
```

The second element is PresenceConditions which defines the presence conditions used to define content of a data model:

```
<PresenceConditions>
    <PresenceCondition name="M" descID="IEC 61850-7-2:PresenceConditions.M.desc"/>
<PresenceConditions>
```

### 7.2.1.5 Abbreviation elements

The root element also contains a list of Abbreviations used within the namespace to build data model names to give the textual description attached.

```
<Abbreviations>
    <Abbreviation name="A" descID="Abbr.A"/>
    <Abbreviation name="St" descID="Abbr.St"/>
    <Abbreviation name="Tms" descID="Abbr.Tms"/>
</Abbreviations>
```

### 7.2.1.6 Attribute types elements (Basic, Enumerated and Constructed)

Attributes are part of a data model as the smallest entity representing a data in a function. These attributes could be of different type:

- Basic types
- Enumerated types
- Constructed types

The attribute types are identified uniquely by their name which will be used by attribute elements to reference it.

Basic types are defined in the NSD under the BasicTypes element as per the definition in IEC 61850-7-2:

```
<BasicTypes>
        <BasicType name="BOOLEAN" descID="IEC 61850-7-2:BasicType.BOOLEAN.desc"/>
</BasicTypes>
```

Enumerated types are defined in NSD by the element Enumeration:

```
<Enumerations>
        <Enumeration name="BehaviourModeKind" titleID="IEC 61850-7-4:BehaviourModeKind.title">
                <Literal name="on" literalVal="1"/>
                <Literal name="blocked" literalVal="2"/>
                <Literal name="test" literalVal="3"/>
                <Literal name="test/blocked" literalVal="4"/>
                <Literal name="off" literalVal="5"/>
        </Enumeration>
</Enumerations>
```

An enumeration is defined by a list of literals with a name and a numerical value.

Constructed types are defined in NSD by the element ConstructedAttribute:

```
<ConstructedAttributes>
        <ConstructedAttribute name="Originator" titleID="IEC 61850-7-2:S_Originator.title">
                <SubDataAttribute name="orCat" type="OriginatorCategoryKind" typeKind="ENUMERATED"
presCond="M"/>
                <SubDataAttribute name="orIdent" type="Octet64" presCond="M"/>
        </ConstructedAttribute>
</ConstructedAttributes>
```

A constructed attribute is composed of sub attributes which could be of the three types defined previously (basic, enumerated or constructed) and with a presence condition used to indicate in which case these sub attributes shall be defined in the real data model.

### 7.2.1.7 CDC elements

Data are part of the data model to represent process/function data by grouping attributes in a meaningful way. The types of these data are defined in IE 61850-7-3 as Common Data Class (CDC) and represented in NSD by CDC elements:

```
<CDCs>
        <CDC name="SPS" titleID="IEC 61850-7-3:SPS.title">
                <DataAttribute name="stVal" fc="ST" type="BOOLEAN" dchg="true" presCond="M"/>
                <DataAttribute name="q" fc="ST" type="Quality" qchg="true" presCond="M"/>
        </CDC>
</CDCs>
```

The CDCs are identified uniquely by their “name” which will be used by DataObject elements to reference it.

A CDC can contain three kinds of sub elements:

- DataAttribute
- SubDataObject

- ServiceParameter

The DataAttribute represents attributes of the CDC with a specific type as defined in 7.2.1.6:

```
<DataAttribute name="ctlModel" fc="CF" type="CtlModelKind" typeKind="ENUMERATED" dchg="true" presCond="M"/>
```

The “typeKind” attribute indicates if it is a basic type (when not defined), an enumerated or a constructed type, and then the “type” attribute indicates the corresponding type. The “presCond” attribute gives the reason of inclusion of the attribute and reference one of the PresenceCondition defined previously.

The SubDataObject represents the sub element of a composed CDC (like WYE):

```
<SubDataObject name="phsA" type="CMV" presCond="AtLeastOne" presCondArgs="1"/>
```

The “type” attributes reference another CDC.

The ServiceParameter represents the parameter which has to be used for the specific control service used on the given CDC:

```
<ServiceParameter name="ctlVal" type="BOOLEAN"/>
```

Some specific CDCs (as settings) could be defined with different variants (SP/SG/SE for settings). To represent this in NSD, a CDC could be declared multiple times with a “variant” attribute allowing distinguishing them:

```
<CDC name="SPG" variant="SP ">
<CDC name="SPG" variant="SG">
<CDC name="SPG" variant="SE">
```

And then the content could differ for the variant part.

### 7.2.1.8 Logical node class elements

Logical nodes are the highest element of the data model hierarchy which could be described in NSD. A logical node is a function composed of different data objects which represent data involved if the function. They are represented by LNClass and AbstractLNClass.

```
<LNClasses>
    <AbstractLNClass name="DomainLN" titleID="IEC 61850-7-4:DomainLN.title">
    <LNClass name="LLN0" titleID="IEC 61850-7-4:LLN0.title" canHaveLOG="true">
</LNClasses>
```

The AbstractLNClass allow definition of a set of data which could be reused by different logical nodes inheriting the AbstractLNClass. In this way an inheritance hierarchy could be used to reuse same functional concepts in different logical nodes as per the example in Figure 8:

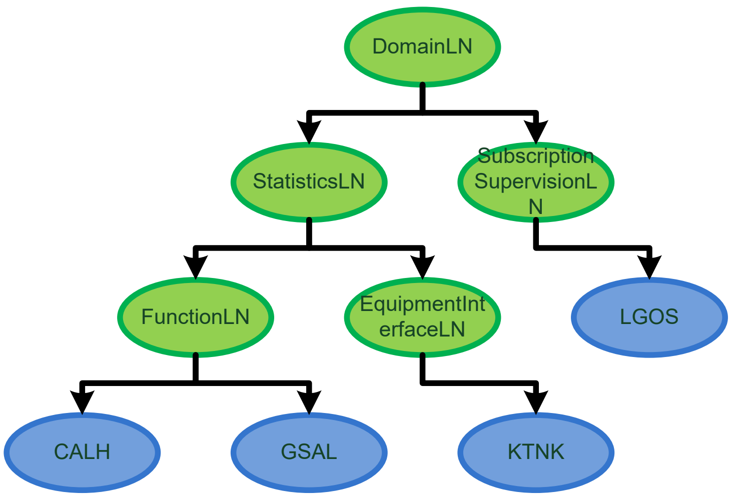


**Figure 8 – Abstract LNClass hierarchy example**

An AbstractLNClass shall not be used by a real data model, and an LNClass shall not be inherited by another LNClass.

Except for this specificity, AbstractLNClass and LNCass contents are identical and both contain a list of DataObject:

```
<DataObject name="Pos" type="DPC" presCond="M" dsPresCond="F"/>
```

The attribute “type” references a CDC defined previously.

### 7.2.2 SNSD

#### 7.2.2.1 General

The second type of file is SNSD which represent the service namespace as defined in IEC 61850-8-x or other specific implementation standards. As per NSD, one SNSD file represents one namespace.

The core namespace IEC 61850-7-2 defines Abstract Communication Service Interface (ACSI) with the type of data and services needed by a data model. When the ACSI is implemented on a real protocol, some additional information may have to be described for the correct implementation. The additional information will be described in an SNSD file.

#### 7.2.2.2 Root element

The root element of an SNSD file will be ServiceNS:

```
<ServiceNS id="IEC 61850-8-1" version="2003">
```

As per NSD, the service namespace is identified by its id, version, revision and release (both are optional).

The ServiceNS contains elements already defined for NSD:

- Changes
- FunctionalConstraints
- PresenceConditions

Then ServiceNS contains also specific elements dedicated to service implementation

#### 7.2.2.3 Type implementation element

Some attribute types defined in ACSI may have a specific definition based on the implementation. The realization of a type is done with ServiceTypeRealization element:

```
<ServiceTypeRealizations>
    <ServiceTypeRealization titleID="IEC 61850-8-1:2003.PhyComAddr" name="PhyComAddr">
        <SubDataAttribute name="Addr" typeKind="BASIC" type="Octet64" presCond="M"/>
    </ServiceTypeRealization>
</ServiceTypeRealizations>
```

This type realization is equivalent to a constructed attribute type as per definition in 7.2.1.6, with a list of sub attributes, and will correspond to an attribute in CDC from IEC 61850-7-3.

#### 7.2.2.4 Constructed attribute element

For the implementation, some additional attribute types may be required for specific services. This is described by ServiceConstructedAttribute element:

```
<ServiceConstructedAttributes>
    <ServiceConstructedAttribute titleID="IEC 61850-8-1:2003.Oper" name="Oper" typeKindParameterized="true">
        <SubDataAttribute name="ctlVal" typeKind="undefined" presCond="M"/>
    </ServiceConstructedAttribute>
</ServiceConstructedAttributes>
```

This constructed type definition is equivalent to a constructed attribute type as per definition in 7.2.1.6, with a list of sub attributes.

#### 7.2.2.5 Service data element

For controllable data, the implementation may require additional attributes. The element ServiceCDC is used for this purpose:

```
<ServiceCDCs>
        <ServiceCDC cdc="SPC">
                <ServiceDataAttribute name="Oper" typeKind="CONSTRUCTED" type="Oper"
underlyingTypeKind="BASIC" underlyingType="BOOLEAN" fc="CO" presCond="MOctrl"/>
        </ServiceCDC>
</ServiceCDCs>
```

The “cdc” attribute indicates an existing CDC from core name space and sub element ServiceDataAttribute extend it specific attributes in the context of this implementation only. This is the element where ServiceConstructedAttribute and FunctionalConstraint defined in this ServiceNS could be used.

### 7.2.3 AppNS

#### 7.2.3.1 General

The third type of file is AppNS which represent the associations between service namespaces and data model namespaces.

This file should be unique and define the data model namespaces which could use the implementation namespaces. This file is named “ApplicableServiceNS.AppNS”

#### 7.2.3.2 Root element

The root element is ApplicableServiceNS:

```
<ApplicableServiceNS version="1" date="2016-04-29T12:00:00Z">
```

It has no identifier as it is a unique file which will list all possible association

#### 7.2.3.3 Service usage element

The only goal of this file is to lists all service namespace defined in the standard and provides for each of the data model namespaces which could use it. This element is ServiceNsUsage:

```
<ServiceNsUsage id="IEC 61850-8-1" version="2003">
        <AppliesTo id="IEC 61850-7-4" version="2003"/>
        <AppliesTo id="IEC 61850-7-4" version="2007" revision="A"/>
</ServiceNsUsage>
```

It indicates a service namespace identified by its “id”, “version” and “revision”.

Then the data model namespaces which use it are listed by element AppliesTo, with namespace “id”, “version” and “revision” as identifier.

In the given example, current MMS implementation (IEC 61850-8-1:2003) is usable by core data model namespaces in edition 1 and edition 2 (IEC 61850-7-4:2003 and IEC 61850-7-4:2007A)

On publication of a new NSD or SNSD, only this file will be updated to introduce new namespaces usage.

### 7.2.4 NSDoc

#### 7.2.4.1 General

The fourth type of file is NSDoc which represent the textual documentation associated to namespaces.

Among all previous NSD and SNSD files, different documentation has been introduced by creating a documentation ID in dedicated attributes (like “titleID” and “descID”). All these IDs will be listed in an NSDoc file with the corresponding textual documentation for a specific language.

#### 7.2.4.2 Root element

The root element is NSDoc:

```
<NSDoc id="IEC 61850-7-4" version="2007" revision="B" lang="en">
```

This element identifies the corresponding namespace by its “id”, “version” and “revision”, and an additional attribute “lang” indicates the language used for this file.

The language value is based on ISO 639-1 which defined codes for representation of languages with 2 characters.

This language attribute allows definition of one file by language to have translation of the same documentation in different languages.

#### 7.2.4.3 Documentation element

The NSDoc element will contain a list of Doc elements:

```
<Doc id="IEC 61850-7-4:LLN0.title"><![CDATA[Logical device LN]]></Doc>
```

This element associates to an “id” (used in an NSD or SNSD file) a textual value represented within the CDATA section which is an XML notation to represent plain text.

The text could be formatted with HTML tag, like <br>, <p>, etc.

In NSD, elements may have different kind of documentation:

- A title to have a short description of the element. This documentation is identified by the attribute “titleId”, available for top elements of a namespace (LNClass, CDC, ConstructedAttribute, etc.).
- A description to have a detailed description. This documentation is identified by the attribute “descId”, available for all elements of a namespace (LNClass, DataObject, CDC, DataAttribute, etc.).
- A presence condition argument description to give more information on a specific case of inclusion for optional elements belonging to another element (DataObject of LNClass, DataAttribute of CDC, etc.). This documentation is identified by the attributes “presCondArgsId” and “dsPresCondArgsId”.

### 7.3 NSD usage convention

#### 7.3.1 General

There are specificities which are not obvious in NSD. The goal of this subclause is to detail them.

### 7.3.2 Element identification

Identification of elements of the NSD is done by means of their name. This means that this name has to be unique for the type of element, i.e. there could be only one CDC named SPS and only one LNClass named XCBR.

This name will be used by other elements to reference this element, i.e. LNClass CSWI contains one DO Pos with attribute “type” set to DPC which correspond to the CDC named DPC.

Then, in each element, sub element will have a name to identify it inside its parent. As per the definition of IEC 61850-7-1, a semantic is attached to these names which means that in two different elements if both have a child with same name, semantic shall remain the same (i.e. Pos data object will indicate position of the device managed by the LNClass containing this data object).

### 7.3.3 Inheritance and extension

#### 7.3.3.1 LNClass

As defined in 7.2.1.8, an inheritance hierarchy of AbstractLNClass and LNClass will simplify the definition of reusable common sets of DataObject. Only LNClass represents a real function which will be integrated into a system, and no LNClass can inherit from another LNClass.

Inheritance of an AbstractLNClass is allowed in a different namespace as the one which define it, if this other namespace depends on it. AbstractLNClass could also be defined in other namespaces than core.

For a namespace which depends on another one, there is also the possibility to extend LNClass to add specificities to an existing function. This will be the only case where a LNClass can reuse the same name as another one, by adding the XML attribute isExtension="true" to the extension LNClass.

Then, this new definition of the LNClass will replace the previous one for the whole namespace which has defined it. The base LNClass shall be defined in the namespace dependency hierarchy.

The LNClass name shall be unique for the whole hierarchy of inheritance of LNClass (see 7.2.1.8), for abstract and real LNClass. The only exception is for the extension which is clearly identified.

As an example, here is the definition of LPHD for IEC TR 61850-90-4 which added new DataObjects:

```
<LNClass titleID="(Tr)IEC 61850-90-4:LPHDExt.title" name="LPHD" isExtension="true">
```

When an LNClass extends/inherits another one, this LNClass can also override the definition of an existing DataObject. This is possible by redefining the given DataObject in the extended/inheriting LNClass.

DataObject overriding is only allowed to change an enumeration (which must be an extension as defined in 7.3.3.2) or to refine a description. A structure of a DataObject shall not be modified as per namespace extension rule defined in IEC 61850-7-1).

#### 7.3.3.2 Enumeration

As per LNClass, it is possible to define an extension of an enumeration in an inherited namespace to avoid redefinition of existing literals.

This extension is indicated by XML attribute inheritedFrom="originalEnumerationName" on an enumeration into a namespace derived from the one containing the “originalEnumerationName”.

Definition in NSD for namespace IEC 61850-7-4:2007B

```
<Enumeration name="AffectedPhasesKind">
        <Literal name="PhaseA" literalVal="1"/>
        <Literal name="PhaseB" literalVal="2"/>
        <Literal name="PhaseAB" literalVal="3"/>
        <Literal name="PhaseC" literalVal="4"/>
        <Literal name="PhaseAC" literalVal="5"/>
        <Literal name="PhaseBC" literalVal="6"/>
        <Literal name="PhaseABC" literalVal="7"/>
        <Literal name="None" literalVal="8"/>
</Enumeration>
```

Definition in NSD for namespace IEC TR 61850-90-17:2015

```
<Enumeration name="AffectedPhases90-17Kind" inheritedFrom="AffectedPhasesKind">
        <Literal name="PhaseABN" literalVal="9"/>
        <Literal name="PhaseACN" literalVal="10"/>
        <Literal name="PhaseBCN" literalVal="11"/>
        <Literal name="PhaseABCN" literalVal="12"/>
</Enumeration>
```

### 7.3.4 DataObject parameterization

As per their usage in IEC 61850-7-3, DataAttributes of DataObjects have a type definition which could be a basic type, a structure or an enumeration.

This type definition uses two XML attributes in NSD in DataAttribute element:

– typeKind which indicate if type is BASIC, ENUMERATED or CONSTRUCTED,

– type which indicate the name of the type, depending of the typeKind.

In some cases, NSD will offer the possibility to define them later, by mean of two specific XML attributes of CDC element:

– when typeKindParameterized="true" is defined, this indicates that the type is not known in NSD and has to be defined on usage (typical case of CTS CDC) , and so typeKind shall be “undefined” and type shall not be defined in DataAttribute

– when enumParameterized="true" is defined, this indicates that the enumeration shall be defined on usage (typical case of enumerated CDCs like ENS), and so typeKind shall be “ENUMERATED” and type shall not be defined in DataAttribute

Then in both cases, the values will be defined in LNClass which use this CDC or if not, in the LN in SCL which is based on this LNClass. This will be done in LNClass by two specific XML attributes at DataObject level (or SubDataObject level in case of composite CDC):

– underlyingTypeKind attribute allows definition of the typeKind (BASIC, ENUMERATED or CONSTRUCTED) for a CDC with typeKindParameterized="true"

– underlyingType attribute allows definition of the type (depending of the typeKind) for a CDC with typeKindParameterized="true" or enumParameterized="true"

As per example for typeKind parameterization, here is an extract of a definition of CTS CDC and its usage on LTRK LNClass:

```
<CDC name="CTS" titleID="IEC 61850-7-3:CTS.title" typeKindParameterized="true">
        <DataAttribute name="ctlVal" fc="SR" typeKind="undefined" presCond="M"/>
</CDC>

<LNClass name="LTRK" titleID="IEC 61850-7-4:LTRK.title" base="DomainLN">
        <DataObject name="SpcTrk" type="CTS" underlyingTypeKind="BASIC" underlyingType="BOOLEAN"/>
</LNClass>
```

And as per example for enumeration parameterization, here is an extract of a definition of ENS CDC and its usage on LPHD LNClass:

```
<CDC name="ENS" titleID="IEC 61850-7-3:ENS.title" enumParameterized="true">
        <DataAttribute name="stVal" fc="ST" typeKind="ENUMERATED" dchg="true" dupd="true" presCond="M"/>
</CDC>

<LNClass name="LPHD" titleID="IEC 61850-7-4:LPHD.title">
        <DataObject name="PhyHealth" type="ENS" underlyingType="HealthKind"/>
</LNClass>
```

There exists also a specific case for parameterized enumeration where the enumeration will be resolved at implementation and not in the NSD itself. To address this case, the specific keyword is “EnumDA”. The following example shows usage of this keyword on FSCH LNClass:

```
<LNClass name="FSCH" titleID="IEC61850_7_4.LNGroupF::FSCH.__cl.title" base="DomainLN">
        <DataObject name="ValENS" type="ENS" underlyingType="EnumDA"/>
</LNClass>
```

### 7.3.5 CDC variants

For some cases, a CDC may have different variant to allow the definition of the same DataAttribute in different ways (like for settings where value could have different functional constraints depending on the usage, SP, SG or SE).

This will be addressed by defining same CDC multiple times (exception to the uniqueness rule for names) with another XML attribute indicating the variant.

This variant will be used to decide in a real project which one to use in an instance LN, by means of the configuration in SCL.

As an example, here is extract of definition of SPG CDC:

```
<CDC name="SPG" variant="SP" titleID="IEC 61850-7-3:SPG_SP.title">
        <DataAttribute name="setVal" fc="SP" type="BOOLEAN" dchg="true" presCond="M"/>
        <DataAttribute name="d" fc="DC" type="VisString255" presCond="O"/>
</CDC>
<CDC name="SPG" variant="SG" titleID="IEC 61850-7-3:SPG_SG.title">
        <DataAttribute name="setVal" fc="SG" type="BOOLEAN" presCond="M"/>
        <DataAttribute name="d" fc="DC" type="VisString255" presCond="O"/>
</CDC>
<CDC name="SPG" variant="SE" titleID="IEC 61850-7-3:SPG_SE.title">
        <DataAttribute name="setVal" fc="SE" type="BOOLEAN" presCond="M"/>
        <DataAttribute name="d" fc="DC" type="VisString255" presCond="O"/>
</CDC>
```

## 7.4 Naming convention

### 7.4.1 File extension

The four types of files will be identified by their file extension:

- For data model namespace: .nsd
- For service namespace: .snsd
- For documentation: .nsdoc
- For service usage, the whole name of the file is fixed: ApplicableServiceNS.AppNS

### 7.4.2 File name

For NSD and SNSD, the file name is the namespace identifier, defined with following rules:

- NamespaceID + ‘_‘ + version [+ revision [+ release]]
- Revision mandatory if not A
- Release mandatory if not 1 and then revision must be specified.

- The namespace ID is prefixed by “eTr” for technical reports IDs, or eTS for technical specifications IDs
- The column (‘:’), brackets (‘(‘ and ‘)’) and white space (‘ ‘) characters cannot be part of a file name and will be replaced by underscore (‘_’).

The version revision release will be used to identify edition of the namespace.

Here are some examples of NSD/SNSD file names:

- IEC 61850-7-4 edition 2: “IEC_61850-7-4_2007A.nsd”
- IEC 61850-7-4 edition 2 amendment 1: “IEC_61850-7-4_2007B.nsd”
- IEC 61850-8-1 edition 1: “IEC_61850-8-1_2003.NSD”
- Technical report IEC TR 61850-90-4 edition 1: “eTr_IEC61850-90-4_2012B.nsd”

For NSDoc, the file name will be same as the corresponding NSD or SNSD with the language added, separated by a dash (‘-‘), like in the following examples:

- “IEC_61850-7-4_2007A-en.nsd”
- “IEC_61850-7-4_2007A-fr.nsd”
- “IEC_61850-8-1_2003-en.nsd”

## 7.5 Example

### 7.5.1 General

The following examples are not taken from any real namespace.

### 7.5.2 NSD file example

```
<NS xmlns="http://www.iec.ch/61850/2016/NSD" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="MyNS"
version="2015" xsi:schemaLocation="http://www.iec.ch/61850/2016/NSD NSD.xsd" publicationStage="CDV">
        <Copyright>
                <Notice>Copyright 2017 sample</Notice>
                <License uri="EULA.txt" kind="Standard">IEC License</License>
        </Copyright>
        <Changes version="2014" revision="B" tissues="12, 13"/>
        <DependsOn id="IEC 61850-7-4" version="2007" revision="B"/>
        <Enumerations>
                <Enumeration name="MyBeh" titleID="MyNS:MyBeh">
                        <Literal name="on" literalVal="1"/>
                        <Literal name="blocked" literalVal="2"/>
                </Enumeration>
        </Enumerations>
        <LNClasses>
                <LNClass name="XCBR" isExtension="true">
                        <DataObject name="MyDO" type="ENS" underlyingType="BreakerOpCapabilityKind" presCond="O"/>
                </LNClass>
                <LNClass name="LAAA" base="DomainLN" titleID="MyNS:LAAA">
                        <DataObject name="FirstData" type="SPS" presCond="M"/>
                        <DataObject name="SecData" type="SPC" presCond="O"/>
                </LNClass>
        </LNClasses>
</NS>
```

### 7.5.3 SNSD file example

```
<ServiceNS xmlns="http://www.iec.ch/61850/2016/NSD" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="MyServiceNS" version="2015" xsi:schemaLocation="http://www.iec.ch/61850/2016/NSD NSD.xsd">
        <FunctionalConstraints>
                <FunctionalConstraint abbreviation="CO"/>
        </FunctionalConstraints>
        <ServiceConstructedAttribute titleID="ID" name="MyOper" typeKindParameterized="true">
                <SubDataAttribute descID="ID" name="ctlVal" typeKind="undefined" presCond="M"/>
        </ServiceConstructedAttribute>
        <ServiceConstructedAttribute titleID="ID" name="MyCancel" typeKindParameterized="true">
                <SubDataAttribute descID="ID" name="ctlVal" typeKind="undefined" presCond="M"/>
```

```
		</ServiceConstructedAttribute>
		<ServiceCDC cdc="SPC">
			<ServiceDataAttribute name="MyOper" typeKind="CONSTRUCTED" type="Oper" underlyingTypeKind="BASIC" 
underlyingType="BOOLEAN" fc="CO"/>
		</ServiceCDC>
		<ServiceCDC cdc="INC">
			<ServiceDataAttribute name="MyOper" typeKind="CONSTRUCTED" type="Oper" underlyingTypeKind="BASIC" 
underlyingType="INT32" fc="CO"/>
			<ServiceDataAttribute name="MyCancel" typeKind="CONSTRUCTED" type="Cancel" 
underlyingTypeKind="BASIC" underlyingType="INT32" fc="CO"/>
		</ServiceCDC>
</ServiceNS>
```

### 7.5.4 AppNS file example

```
<ApplicableServiceNS xmlns="http://www.iec.ch/61850/2016/NSD" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" 
version="1" date="2016-04-29T12:00:00Z" xsi:schemaLocation="http://www.iec.ch/61850/2016/NSD NSD.xsd">
		<ServiceNsUsage id="MyServiceNS" version="2015">
			<AppliesTo id="MyNs" version="2015"/>
			<AppliesTo id="MyOtherNs" version="2016" revision="A"/>
		</ServiceNsUsage>
</ApplicableServiceNS>
```

### 7.5.5 NSDoc file example

```
<NSDoc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.iec.ch/61850/2016/NSD 
NSD.xsd" xmlns="http://www.iec.ch/61850/2016/NSD" id="MyNS" version="2015" lang="en">
		<Doc id="MyNS:MyBeh"><![CDATA[<p>documentation example for MyBeh enumeration.</p>]]></Doc>
		<Doc id="MyNS:LAAA"><![CDATA[<p>documentation example LAAA logical node.</p>]]></Doc>
</NSDoc>
```

## 8 File usage

### 8.1 General

An NSD file can be used for many purpose depending of the user needs and the tool which use it. This clause will give some examples of possible usage, based on already identified use cases.

These examples are not an exhaustive list of possible usage.

### 8.2 System specification creation

A System Specification Tool can take benefit of the NSD files to give to a user the possibility to select a namespace related to its needs (substation automation, hydroelectric power plants, wind power plants, distributed energy resources, customer private domain, etc.) and give him the possibility to explore the whole related data model elements (Logical nodes, Data objects, Common data classes, Data attributes, etc.).

With this capability, a user will be able to create his system specification using an SSD more precisely when he will identify required functions by adding Logical nodes to its primary system.

The user will have the list of all logical nodes of the desired domains, with possibly a description, and then he will have the possibility to define precisely the data objects and data attributes required by its project.

For example, in an SSD, he will have the possibility to add in a Circuit Breaker (in Substation section, a conducting equipment with the type “CBR”) a logical node CSWI (in the conducting equipment, an element LNode with lnClass “CSWI”) to identify the control of the device. This example is shown in Figure 9:

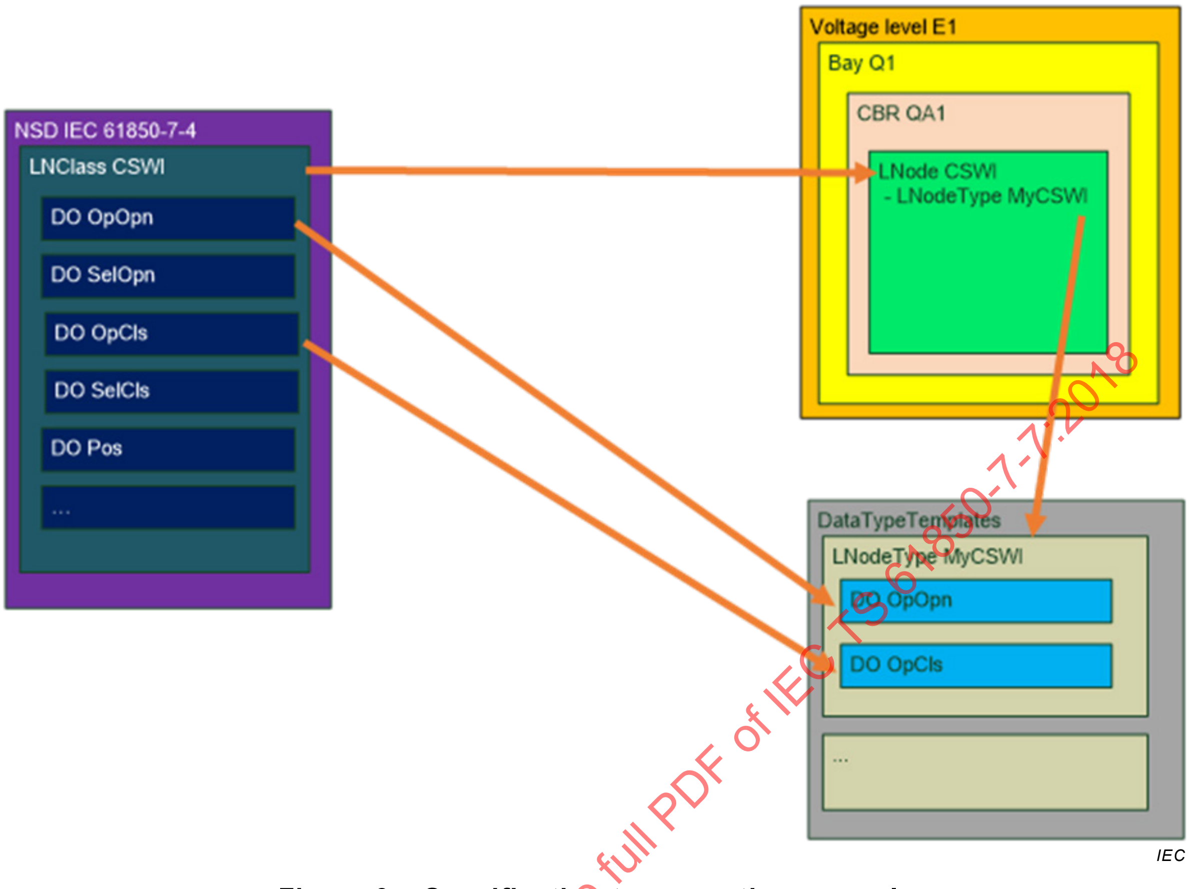


**Figure 9 – Specification type creation example**

Then the user will create a logical node type (in the Data Type Template section, an element LNodeType with an ID to be referenced by previously created LNode) for this CSWI and add the required data objects OpOpn and OpCls, to control the XCBR throw gooses (adding two DO elements with name OpOpn and OpCls within CSWI LNodeType created before).

All this creation process will be done according to the standard by accessing IEC 61850-7-4 in an electronic format.

## 8.3 Manufacturer IED creation

A manufacturer tool may take benefit of the NSD to choose the right LNode and Data object when it creates the modelling of an IED.

As per system specification creation, the user will create data type templates based on NSD definition, with the help of the description to know which element to use.

This will provide a manufacturer data model in line with the standard.

## 8.4 Creation of a non-standard name space

When a manufacturer creates a data model for a product, a standard data model may not be enough and additional data may be required for specific feature of the device.

These additional data are defined as a namespace extension which needs to follow IEC 61850-7-x rules. NSD will help in defining an extension by giving existing LNode classes and data object names to avoid reuse an existing one, as it is forbidden in extension rules.

The user will be able to use an existing LNClass to extend it, or to create a new one. Then he will be able to create new data object with a CDC already defined by the standard.

Definition of an extension could serve manufacturers to specify the data required by their devices or by customers to specify their specific needs.

The output will be a new NSD file usable by other tools compliant with NSD format.

## 8.5 IED data model validation

When a manufacturer provides a new device to a test/certification team, the team could take benefit of the NSD to compare the implemented data model with the standard in an automatic way.

The team will be able to look at standard LN class, data objects and CDC if the implementation is conforming to the corresponding standard NSD. It will also be able to verify the extension NSD.

## 8.6 Data model update management

The NSD will be available as code component of the related part and will be available for download on IEC website.

When an update of a standard data model is needed, the corresponding NSD will be updated accordingly and published before the related part. Then purchasers of the related part will be able to download the corrected NSD.

This will help to distribute Tissues fixes faster.

# Annex A
(normative)

## Schema NSD.xsd

<CODE BEGINS>

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.iec.ch/61850/2016/NSD" xmlns:nsd="http://www.iec.ch/61850/2016/NSD"
xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.iec.ch/61850/2016/NSD"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="2017A">
	<xs:annotation>
		<xs:documentation xml:lang="en">
			COPYRIGHT (c) IEC, 2017. This version of this XSD is part of IEC 61850-7-7:2017; see the IEC 61850-
7-7:2017 for full legal notices. In case of any differences between the here-below code and the IEC published content, the IEC
published content remains the reference to be considered. The whole document has to be taken into account to have a full
description of this code component.
		See www.iec.ch/CCv1 for copyright details.
		</xs:documentation>
		<xs:documentation xml:lang="en">
			NameSpace Definition file syntax (NSD.xsd) for the machine processable format for tools.
			2017-08-28 (version 2017A).

			Describes in a machine-processable way selected components from the IEC 61850 data model.
			Four elements are defined:
			- NS: shall hold the namespace definition, all its documentation strings to be resolved in the sibling file
(i.e., with same name but file extension "NSDOC") with NSDoc as root. Shall have as file extension "NSD".
			- ServiceNS: shall hold a service namespace definition, all its documentation strings to be resolved in
the sibling file (i.e., with same name but file extension "NSDOC") with NSDoc as root. Shall have as file extension "SNSD".
			- NSDoc: shall hold all documentation strings defined in the sibling NS file. Shall have as file extension
"NSDOC".
			- ApplicableServiceNS: shall hold definition of link between NS and applicable ServiceNS. Shall have as
file extension "AppNS".
		</xs:documentation>
	</xs:annotation>
	<xs:include schemaLocation="IECCopyright.xsd"/>
	<!-- ============================================================================ -->
	<xs:simpleType name="tDocID">
		<xs:annotation>
			<xs:documentation>Identifier referring to a documentation string, available in a sibling
file.</xs:documentation>
		</xs:annotation>
		<xs:restriction base="xs:normalizedString">
			<xs:minLength value="1"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="tIec61850Name">
		<xs:restriction base="xs:Name">
			<xs:minLength value="1"/>
			<xs:pattern value="[\p{IsBasicLatin}\p{IsLatin-1Supplement}]+"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="tEmptyString">
		<xs:annotation>
			<xs:documentation>The empty string.</xs:documentation>
		</xs:annotation>
		<xs:restriction base="xs:normalizedString">
			<xs:maxLength value="0"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="tIec61850NameString">
		<xs:restriction base="xs:normalizedString">
			<xs:pattern value="[\p{IsBasicLatin}\p{IsLatin-1Supplement}]+"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="tUMLVersion">
		<xs:annotation>
			<xs:documentation>Version of the UML model used to generate this NSD.</xs:documentation>
		</xs:annotation>
		<xs:restriction base="xs:Name">
			<xs:minLength value="1"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="tNSIdentifier">
```

```
		<xs:annotation>
			<xs:documentation>Identifier of a namespace.</xs:documentation>
		</xs:annotation>
		<xs:restriction base="xs:normalizedString">
			<xs:pattern value="\p{IsBasicLatin}+"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="tNSVersion">
		<xs:annotation>
			<xs:documentation>Version of a namespace (a year between 2002 and 2099).</xs:documentation>
		</xs:annotation>
		<xs:restriction base="xs:unsignedShort">
			<xs:minInclusive value="2002"/>
			<xs:maxInclusive value="2099"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="tNSRevision">
		<xs:annotation>
			<xs:documentation>Revision of a namespace (a basic latin upper-case letter), distinguishing between
revisions of a same namespace version.</xs:documentation>
		</xs:annotation>
		<xs:restriction base="xs:token">
			<xs:pattern value="[A-Z]"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="tNSRelease">
		<xs:annotation>
			<xs:documentation>Release of a namespace (number between 1 and 255), distinguishing between
releases of a same namespace version and revision.</xs:documentation>
		</xs:annotation>
		<xs:restriction base="xs:unsignedByte">
			<xs:minExclusive value="0"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="tPubStage">
		<xs:annotation>
			<xs:documentation>Publication stage of the namespace.</xs:documentation>
		</xs:annotation>
		<xs:restriction base="xs:token">
			<xs:enumeration value="WD">
				<xs:annotation>
					<xs:documentation>Working Draft</xs:documentation>
				</xs:annotation>
			</xs:enumeration>
			<xs:enumeration value="CD">
				<xs:annotation>
					<xs:documentation>Committee Draft</xs:documentation>
				</xs:annotation>
			</xs:enumeration>
			<xs:enumeration value="CDV">
				<xs:annotation>
					<xs:documentation>Committee Draft for Vote</xs:documentation>
				</xs:annotation>
			</xs:enumeration>
			<xs:enumeration value="DTS">
				<xs:annotation>
					<xs:documentation>Draft Technical Specification</xs:documentation>
				</xs:annotation>
			</xs:enumeration>
			<xs:enumeration value="DTR">
				<xs:annotation>
					<xs:documentation>Draft Technical Report</xs:documentation>
				</xs:annotation>
			</xs:enumeration>
			<xs:enumeration value="FDIS">
				<xs:annotation>
					<xs:documentation>Final Draft International Standard</xs:documentation>
				</xs:annotation>
			</xs:enumeration>
			<xs:enumeration value="TS">
				<xs:annotation>
					<xs:documentation>Technical Specification</xs:documentation>
				</xs:annotation>
			</xs:enumeration>
			<xs:enumeration value="TR">
				<xs:annotation>
					<xs:documentation>Technical Report</xs:documentation>
```

```
                    </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="IS">
                    <xs:annotation>
                        <xs:documentation>International Standard</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tDefinedAttributeTypeKind">
            <xs:annotation>
                <xs:documentation>Type kind of a data attribute.</xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:token">
                <xs:enumeration value="BASIC">
                    <xs:annotation>
                        <xs:documentation>The type of the data attribute is a basic one (e.g., BOOLEAN,
INT32), including CODED ENUMs (e.g., Dbpos, Tcmd) and PACKED LISTs (Timestamp, Quality, OptFlds).</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="ENUMERATED">
                    <xs:annotation>
                        <xs:documentation>The type of the data attribute is an
enumeration.</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="CONSTRUCTED">
                    <xs:annotation>
                        <xs:documentation>The type of the data attribute is a constructed (composed)
one.</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tUndefinedAttributeTypeKind">
            <xs:annotation>
                <xs:documentation>Type kind for a data attribute which has its type "undefined".</xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:token">
                <xs:enumeration value="undefined">
                    <xs:annotation>
                        <xs:documentation>Indicates that the type is not defined, and shall be done so where it
is used.</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tAttributeTypeKind">
            <xs:annotation>
                <xs:documentation>Type kind of a data attribute, which may be "undefined".</xs:documentation>
            </xs:annotation>
            <xs:union memberTypes="tDefinedAttributeTypeKind tUndefinedAttributeTypeKind"/>
        </xs:simpleType>
        <xs:simpleType name="tFCAbbreviation">
            <xs:annotation>
                <xs:documentation>Functional Constraint abbreviation.</xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:token">
                <xs:minLength value="1"/>
                <xs:pattern value="[\p{IsBasicLatin}]+"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tLiteralName">
            <xs:annotation>
                <xs:documentation>Name of an enumeration literal. Maybe the empty string.</xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:normalizedString">
                <xs:maxLength value="127"/>
                <xs:pattern value="[\p{IsBasicLatin}\p{IsLatin-1Supplement}]*"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tAttributeName">
            <xs:annotation>
                <xs:documentation>Name of a Data Attribute.</xs:documentation>
            </xs:annotation>
            <xs:restriction base="tIec61850Name"/>
        </xs:simpleType>
```

```
<xs:simpleType name="tSubDataObjectName">
        <xs:annotation>
                <xs:documentation>Name of a Sub-Data Object.</xs:documentation>
        </xs:annotation>
        <xs:restriction base="tIec61850Name"/>
</xs:simpleType>
<xs:simpleType name="tCDCName">
        <xs:annotation>
                <xs:documentation>Name of a Common Data Class (CDC).</xs:documentation>
        </xs:annotation>
        <xs:restriction base="tIec61850Name">
                <xs:minLength value="1"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tDataObjectName">
        <xs:annotation>
                <xs:documentation>Name of a Data Object.</xs:documentation>
        </xs:annotation>
        <xs:restriction base="tIec61850Name">
                <xs:maxLength value="12"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tLNClassName">
        <xs:annotation>
                <xs:documentation>Name of a (non-abstract) logical node class.</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:Name">
                <xs:pattern value="LLN0"/>
                <xs:pattern value="[A-Z]{4}"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tAbstractLNClassName">
        <xs:annotation>
                <xs:documentation>Name of an abstract logical node.</xs:documentation>
        </xs:annotation>
        <xs:restriction base="tIec61850Name"/>
</xs:simpleType>
<xs:simpleType name="tPresenceConditionName">
        <xs:annotation>
                <xs:documentation>Name of a presence condition of a child.</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:normalizedString"/>
</xs:simpleType>
<xs:simpleType name="tPresenceConditionArgument">
        <xs:annotation>
                <xs:documentation>Argument to (some) presence conditions.</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:normalizedString"/>
</xs:simpleType>
<xs:simpleType name="tAbbreviationName">
        <xs:annotation>
                <xs:documentation>Name of an abbreviation.</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:normalizedString"/>
</xs:simpleType>
<xs:simpleType name="tCBKind">
        <xs:annotation>
                <xs:documentation>Enumeration of control block kinds.</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:normalizedString">
                <xs:enumeration value="RCB"/>
                <xs:enumeration value="LCB"/>
                <xs:enumeration value="GoCB"/>
                <xs:enumeration value="SVCB"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tBasicTypeName">
        <xs:annotation>
                <xs:documentation>Type describing the name of a basic type of a data attribute.</xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:token">
                <xs:minLength value="1"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tACSIServicesKind">
        <xs:annotation>
```

```
				<xs:documentation>Enumeration holding the names of all ACSI services, as defined in Part 7-2. Is
conceptually identical to the 7-2 ServiceNameKind enumeration (except the Unknown).</xs:documentation>
			</xs:annotation>
			<xs:restriction base="xs:token">
				<xs:enumeration value="Associate"/>
				<xs:enumeration value="Abort"/>
				<xs:enumeration value="Release"/>
				<xs:enumeration value="GetServerDirectory"/>
				<xs:enumeration value="GetLogicalDeviceDirectory"/>
				<xs:enumeration value="GetAllDataValues"/>
				<xs:enumeration value="GetDataValues"/>
				<xs:enumeration value="SetDataValues"/>
				<xs:enumeration value="GetDataDirectory"/>
				<xs:enumeration value="GetDataDefinition"/>
				<xs:enumeration value="GetDataSetValues"/>
				<xs:enumeration value="SetDataSetValues"/>
				<xs:enumeration value="CreateDataSet"/>
				<xs:enumeration value="DeleteDataSet"/>
				<xs:enumeration value="GetDataSetDirectory"/>
				<xs:enumeration value="SelectActiveSG"/>
				<xs:enumeration value="SelectEditSG"/>
				<xs:enumeration value="SetEditSGValue"/>
				<xs:enumeration value="ConfirmEditSGValues"/>
				<xs:enumeration value="GetEditSGValue"/>
				<xs:enumeration value="GetSGCBValues"/>
				<xs:enumeration value="Report"/>
				<xs:enumeration value="GetBRCBValues"/>
				<xs:enumeration value="SetBRCBValues"/>
				<xs:enumeration value="GetURCBValues"/>
				<xs:enumeration value="SetURCBValues"/>
				<xs:enumeration value="GetLCBValues"/>
				<xs:enumeration value="SetLCBValues"/>
				<xs:enumeration value="QueryLogByTime"/>
				<xs:enumeration value="QueryLogAfter"/>
				<xs:enumeration value="GetLogStatusValues"/>
				<xs:enumeration value="SendGOOSEMessage"/>
				<xs:enumeration value="GetGoCBValues"/>
				<xs:enumeration value="SetGoCBValues"/>
				<xs:enumeration value="GetGoReference"/>
				<xs:enumeration value="GetGOOSEElementNumber"/>
				<xs:enumeration value="SendMSVMessage"/>
				<xs:enumeration value="GetMSVCBValues"/>
				<xs:enumeration value="SetMSVCBValues"/>
				<xs:enumeration value="SendUSVMessage"/>
				<xs:enumeration value="GetUSVCBValues"/>
				<xs:enumeration value="SetUSVCBValues"/>
				<xs:enumeration value="Select"/>
				<xs:enumeration value="SelectWithValue"/>
				<xs:enumeration value="Cancel"/>
				<xs:enumeration value="Operate"/>
				<xs:enumeration value="CommandTermination"/>
				<xs:enumeration value="TimeActivatedOperate"/>
				<xs:enumeration value="GetFile"/>
				<xs:enumeration value="SetFile"/>
				<xs:enumeration value="DeleteFile"/>
				<xs:enumeration value="GetFileAttributeValues"/>
				<xs:enumeration value="TimeSynchronization"/>
				<xs:enumeration value="InternalChange"/>
				<xs:enumeration value="GetLogicalNodeDirectory"/>
				<xs:enumeration value="GetMsvReference"/>
				<xs:enumeration value="GetMSVElementNumber"/>
				<xs:enumeration value="GetUsvReference"/>
				<xs:enumeration value="GetUSVElementNumber"/>
			</xs:restriction>
		</xs:simpleType>
		<!-- ============================================================================ -->
		<xs:attributeGroup name="agNSIdentification">
			<xs:annotation>
				<xs:documentation>Full identification of a namespace.</xs:documentation>
			</xs:annotation>
			<xs:attribute name="id" type="tNSIdentifier" use="required">
				<xs:annotation>
					<xs:documentation>Identifier of the namespace, e.g., "IEC 61850-7-4", "(Tr)IEC 61850-90-
4".</xs:documentation>
				</xs:annotation>
			</xs:attribute>
			<xs:attribute name="version" type="tNSVersion" use="required">
```

```
				<xs:annotation>
					<xs:documentation>Version (year) of the namespace.</xs:documentation>
				</xs:annotation>
			</xs:attribute>
			<xs:attribute name="revision" type="tNSRevision" use="optional" default="A">
				<xs:annotation>
					<xs:documentation>Revision of the namespace version, by default 'A'.</xs:documentation>
				</xs:annotation>
			</xs:attribute>
			<xs:attribute name="release" type="tNSRelease" use="optional" default="1">
				<xs:annotation>
					<xs:documentation>Release of the namespace version and revision, by default
1.</xs:documentation>
				</xs:annotation>
			</xs:attribute>
			<xs:attribute name="publicationStage" type="tPubStage" use="optional" default="IS">
				<xs:annotation>
					<xs:documentation>The publication stage of the namespace, by default IS.</xs:documentation>
				</xs:annotation>
			</xs:attribute>
		</xs:attributeGroup>
		<xs:attributeGroup name="agPresenceCondition">
			<xs:annotation>
				<xs:documentation>Presence condition definition.</xs:documentation>
			</xs:annotation>
			<xs:attribute name="presCond" type="tPresenceConditionName" use="optional" default="M">
				<xs:annotation>
					<xs:documentation>Presence condition of the element. By default 'M'
(mandatory).</xs:documentation>
				</xs:annotation>
			</xs:attribute>
			<xs:attribute name="presCondArgs" type="tPresenceConditionArgument" use="optional">
				<xs:annotation>
					<xs:documentation>Optional argument to the presence condition. Can be a sibling element or a
group number.</xs:documentation>
				</xs:annotation>
			</xs:attribute>
			<xs:attribute name="presCondArgsID" type="tDocID" use="optional">
				<xs:annotation>
					<xs:documentation>Optional argument to the presence condition: a documentation identifier
referring to some free text.</xs:documentation>
				</xs:annotation>
			</xs:attribute>
		</xs:attributeGroup>
		<xs:attributeGroup name="agPresenceConditionDerivedStatistics">
			<xs:annotation>
				<xs:documentation>Presence condition definition for the derived statistics context.</xs:documentation>
			</xs:annotation>
			<xs:attribute name="dsPresCond" type="tPresenceConditionName" use="optional" default="M">
				<xs:annotation>
					<xs:documentation>Presence condition of the element. By default 'M'
(mandatory).</xs:documentation>
				</xs:annotation>
			</xs:attribute>
			<xs:attribute name="dsPresCondArgs" type="tPresenceConditionArgument" use="optional">
				<xs:annotation>
					<xs:documentation>Optional argument to the presence condition. Can be a sibling element or a
group number.</xs:documentation>
				</xs:annotation>
			</xs:attribute>
			<xs:attribute name="dsPresCondArgsID" type="tDocID" use="optional">
				<xs:annotation>
					<xs:documentation>Optional argument to the presence condition: a documentation identifier
referring to some free text.</xs:documentation>
				</xs:annotation>
			</xs:attribute>
		</xs:attributeGroup>
		<xs:attributeGroup name="agArray">
			<xs:annotation>
				<xs:documentation>Array definition.</xs:documentation>
			</xs:annotation>
			<xs:attribute name="isArray" type="xs:boolean" use="optional" default="false">
				<xs:annotation>
					<xs:documentation>Flag indicating whether the element is an array. By default, not an
array.</xs:documentation>
				</xs:annotation>
			</xs:attribute>
```

```
        <xs:attribute name="minIndex" type="xs:unsignedInt" use="optional" default="0">
            <xs:annotation>
                <xs:documentation>Lowest index of the array.
Is relevant information if and only if isArray=true.
By default 0.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="sizeAttribute" type="tAttributeName" use="optional">
            <xs:annotation>
                <xs:documentation>Sibling data attribute name holding the size of the array.
Is relevant information if and only if isArray=true. One and only one of sizeAttribute and maxIndexAttribute shall be
used.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="maxIndexAttribute" type="tAttributeName" use="optional">
            <xs:annotation>
                <xs:documentation>Name of the Attribute holding the maximal index of the array.
Is relevant information if and only if isArray=true. One and only one of sizeAttribute and maxIndexAttribute shall be
used.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:attributeGroup>
    <xs:attributeGroup name="agTrgOp">
        <xs:annotation>
            <xs:documentation>Triggering conditions definition.</xs:documentation>
        </xs:annotation>
        <xs:attribute name="dchg" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="qchg" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="dupd" type="xs:boolean" use="optional" default="false"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="agAttributeType">
        <xs:annotation>
            <xs:documentation>Definition of the type of a data attribute.</xs:documentation>
        </xs:annotation>
        <xs:attribute name="typeKind" type="tAttributeTypeKind" use="optional" default="BASIC">
            <xs:annotation>
                <xs:documentation>The kind of the data attribute's type. By  default,
"BASIC".</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="type" type="tIec61850Name" use="optional">
            <xs:annotation>
                <xs:documentation>The type of the data attribute, e.g., the name of a basic type like INT32, the
name of an enumeration of a constructed data attribute – depending on the sibling typeKind. If not provided, shall be defined by
the "parent" element.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:attributeGroup>
    <xs:attributeGroup name="agAttributeTypeAndValues">
        <xs:annotation>
            <xs:documentation>Definition of the type of a data attribute.</xs:documentation>
        </xs:annotation>
        <xs:attributeGroup ref="agAttributeType"/>
        <xs:attribute name="defaultValue" type="xs:normalizedString" use="optional">
            <xs:annotation>
                <xs:documentation>Default value for the data attribute. May only be defined for data attributes
with typeKind="BASIC" or "ENUMERATED".</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="minValue" type="xs:decimal" use="optional">
            <xs:annotation>
                <xs:documentation>Allowed minimal value (inclusive) for the data attribute. If not provided the
minimal value according to the type is allowed. May only be defined for data attributes with typeKind="BASIC" and
corresponding to a number.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="maxValue" type="xs:decimal" use="optional">
            <xs:annotation>
                <xs:documentation>Allowed maximal value (inclusive) for the data attribute. If not provided the
minimal value according to the type is allowed. May only be defined for data attributes with typeKind="BASIC" and
corresponding to a number.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:attributeGroup>
    <xs:attributeGroup name="agUnderlyingType">
        <xs:attribute name="underlyingTypeKind" type="tDefinedAttributeTypeKind" use="optional">
            <xs:annotation>
```

```
                <xs:documentation>The typeKind to be used for all "undefined" attributes of the CDC of this 
DataObject.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="underlyingType" type="tIec61850Name" use="optional">
            <xs:annotation>
                <xs:documentation>Type to be used for type-open CDCs, e.g., enumeration to be used for 
enumeration-based CDCs (e.g., ENS, ENC, ENG).</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:attributeGroup>
    <xs:attributeGroup name="agUML">
        <xs:attribute name="umlVersion" type="tUMLVersion" use="optional">
            <xs:annotation>
                <xs:documentation>Version of UML from which this namespace definition file was generated 
from.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="umlDate" type="xs:dateTime" use="optional">
            <xs:annotation>
                <xs:documentation>UTC Date and time of the UML version from which this namespace 
definition file was generated. Shall be provided if umlVersion is present. Format: YYYY-MM-DDThh:mm:ssZ, where: YYYY 
indicates the year, MM indicates the month, DD indicates the day, T indicates the start of the required time section, hh indicates 
the hour, mm indicates the minute, ss indicates the second, and Z the UTC time stamp indication.
</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:attributeGroup>
    <xs:attributeGroup name="agNSdesc">
        <xs:annotation>
            <xs:documentation>Description of a namespace (typically the application domain)</xs:documentation>
        </xs:annotation>
        <xs:attribute name="descID" type="tDocID" use="optional">
            <xs:annotation>
                <xs:documentation>Documentation identifier referring to a full description of this 
NS.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:attributeGroup>
    <!-- ============================================================================ -->
    <xs:complexType name="tDocumentedClass">
        <xs:annotation>
            <xs:documentation>A class with description.</xs:documentation>
        </xs:annotation>
        <xs:attribute name="descID" type="tDocID" use="optional">
            <xs:annotation>
                <xs:documentation>Documentation identifier referring to a full description of this 
object.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="informative" type="xs:boolean" use="optional" default="false">
            <xs:annotation>
                <xs:documentation>Flag indicating whether this object is classified as informative or not. By 
default not informative.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="deprecated" type="xs:boolean" use="optional" default="false">
            <xs:annotation>
                <xs:documentation>Flag indicating whether this object is classified as deprecated or not. By 
default not deprecated.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="tTitledClass">
        <xs:annotation>
            <xs:documentation>A tDocumentedClass with title.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="tDocumentedClass">
                <xs:attribute name="titleID" type="tDocID" use="required">
                    <xs:annotation>
                        <xs:documentation>Documentation identifier referring to the title (alias) of this 
object.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tChanges">
        <xs:annotation>
            <xs:documentation>Namespace history, e.g., which version it is based on and which TISSUES it 
includes since then.</xs:documentation>
        </xs:annotation>
        <xs:attribute name="version" type="tNSVersion" use="required">
            <xs:annotation>
                <xs:documentation>Version of the namespace this NSD file is based on.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="revision" type="tNSRevision" use="optional" default="A">
            <xs:annotation>
                <xs:documentation>Revision of the namespace this NSD file is based on, per default 
'A'.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="release" type="tNSRelease" use="optional" default="1">
            <xs:annotation>
                <xs:documentation>Release of the namespace this NSD file is based on, per default 
1.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="tissues" use="optional">
            <xs:annotation>
                <xs:documentation>Comma-separated list of TISSUE numbers that were implemented since 
the previous namespace version/revision/release.</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:normalizedString">
                    <xs:minLength value="1"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="changesID" type="tDocID" use="optional">
            <xs:annotation>
                <xs:documentation>Documentation identifier referring to a textual description of changes (not 
TISSUE list). Shall only be used for private namespaces.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="tBasicType">
        <xs:annotation>
            <xs:documentation>Definition of a basic type for a data attribute.</xs:documentation>
        </xs:annotation>
        <xs:attribute name="name" type="tBasicTypeName" use="required">
            <xs:annotation>
                <xs:documentation>Name of the basic type (e.g., INT32), as used in SCL.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="descID" type="tDocID" use="optional">
            <xs:annotation>
                <xs:documentation>Documentation identifier referring to a full description of this 
object.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="tBasicTypes">
        <xs:annotation>
            <xs:documentation>List of basic types for data attributes introduced by a 
namespace.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="BasicType" type="tBasicType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="tDataSetMemberOf">
        <xs:annotation>
            <xs:documentation>Indication of a given object can be a data set member of a given control 
block.</xs:documentation>
        </xs:annotation>
        <xs:attribute name="cb" type="tCBKind" use="required">
            <xs:annotation>
                <xs:documentation>Control block kind.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>
```

```
	<xs:complexType name="tApplicableServices">
		<xs:annotation>
			<xs:documentation>Applicable services for a given functional constraint.</xs:documentation>
		</xs:annotation>
		<xs:sequence>
			<xs:element name="Service" minOccurs="0" maxOccurs="unbounded">
				<xs:annotation>
					<xs:documentation>If present, indicates that the service with given name applies to
attributes with the specified FC (otherwise it may not be used).</xs:documentation>
				</xs:annotation>
				<xs:complexType>
					<xs:attribute name="name" type="tACSIServicesKind" use="required"/>
				</xs:complexType>
			</xs:element>
			<xs:element name="DataSetMemberOf" type="tDataSetMemberOf" minOccurs="0"
maxOccurs="unbounded">
				<xs:annotation>
					<xs:documentation>An attribute of this FC can be member of a dataset for a control
block type indicated by attribute cb if and only if this element is present.</xs:documentation>
				</xs:annotation>
			</xs:element>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="tFunctionalConstraint">
		<xs:annotation>
			<xs:documentation>Definition of a Functional Constraint.</xs:documentation>
		</xs:annotation>
		<xs:sequence>
			<xs:element name="ApplicableServices" type="tApplicableServices" minOccurs="0">
				<xs:unique name="uniqueDataSetMemberOf">
					<xs:selector xpath="nsd:DataSetMemberOf"/>
					<xs:field xpath="@cb"/>
				</xs:unique>
				<xs:unique name="uniqueService">
					<xs:selector xpath="nsd:Service"/>
					<xs:field xpath="@name"/>
				</xs:unique>
			</xs:element>
		</xs:sequence>
		<xs:attribute name="abbreviation" type="tFCAbbreviation" use="required">
			<xs:annotation>
				<xs:documentation>Abbreviated name of the FC (e.g., ST, MX, etc.).</xs:documentation>
			</xs:annotation>
		</xs:attribute>
		<xs:attribute name="titleID" type="tDocID" use="optional">
			<xs:annotation>
				<xs:documentation>Documentation identifier referring to the title (alias) of this
object.</xs:documentation>
			</xs:annotation>
		</xs:attribute>
		<xs:attribute name="descID" type="tDocID" use="optional">
			<xs:annotation>
				<xs:documentation>Documentation identifier referring to a full description of this
object.</xs:documentation>
			</xs:annotation>
		</xs:attribute>
	</xs:complexType>
	<xs:complexType name="tFunctionalConstraints">
		<xs:annotation>
			<xs:documentation>List of Functional Constraints introduced by a namespace.</xs:documentation>
		</xs:annotation>
		<xs:sequence>
			<xs:element name="FunctionalConstraint" type="tFunctionalConstraint" maxOccurs="unbounded">
				<xs:annotation>
					<xs:documentation>Definition of a Functional Constraint.</xs:documentation>
				</xs:annotation>
			</xs:element>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="tPresenceCondition">
		<xs:annotation>
			<xs:documentation>Definition of a presence condition.</xs:documentation>
		</xs:annotation>
		<xs:attribute name="name" type="tPresenceConditionName" use="required">
			<xs:annotation>
				<xs:documentation>Name of the presence condition, as used in IEC 61850 data
models.</xs:documentation>
```

```
					</xs:annotation>
				</xs:attribute>
				<xs:attribute name="argument" type="tPresenceConditionArgument" use="optional">
					<xs:annotation>
						<xs:documentation>The presence condition argument, if any.</xs:documentation>
					</xs:annotation>
				</xs:attribute>
				<xs:attribute name="titleID" type="tDocID" use="optional">
					<xs:annotation>
						<xs:documentation>Documentation identifier referring to the title (alias) of this
object.</xs:documentation>
					</xs:annotation>
				</xs:attribute>
				<xs:attribute name="descID" type="tDocID" use="optional">
					<xs:annotation>
						<xs:documentation>Documentation identifier referring to a full description of this
object.</xs:documentation>
					</xs:annotation>
				</xs:attribute>
			</xs:complexType>
			<xs:complexType name="tPresenceConditions">
				<xs:annotation>
					<xs:documentation>List of presence conditions which are introduced by a
namespace.</xs:documentation>
				</xs:annotation>
				<xs:sequence>
					<xs:element name="PresenceCondition" type="tPresenceCondition" maxOccurs="unbounded">
						<xs:annotation>
							<xs:documentation>Definition of a presence condition.</xs:documentation>
						</xs:annotation>
					</xs:element>
				</xs:sequence>
			</xs:complexType>
			<xs:complexType name="tAbbreviation">
				<xs:annotation>
					<xs:documentation>Definition of an abbreviation.</xs:documentation>
				</xs:annotation>
				<xs:attribute name="name" type="tAbbreviationName" use="required">
					<xs:annotation>
						<xs:documentation>Name of the abbreviation, as used in IEC 61850 data
models.</xs:documentation>
					</xs:annotation>
				</xs:attribute>
				<xs:attribute name="descID" type="tDocID" use="optional">
					<xs:annotation>
						<xs:documentation>Documentation identifier referring to a full description of this
abbreviation.</xs:documentation>
					</xs:annotation>
				</xs:attribute>
			</xs:complexType>
			<xs:complexType name="tAbbreviations">
				<xs:annotation>
					<xs:documentation>List of abbreviationss which are introduced by a namespace.</xs:documentation>
				</xs:annotation>
				<xs:sequence>
					<xs:element name="Abbreviation" type="tAbbreviation" maxOccurs="unbounded">
						<xs:annotation>
							<xs:documentation>Definition of an abbreviation.</xs:documentation>
						</xs:annotation>
					</xs:element>
				</xs:sequence>
			</xs:complexType>
			<xs:complexType name="tLiteral">
				<xs:annotation>
					<xs:documentation>Definition of a literal of an enumeration.</xs:documentation>
				</xs:annotation>
				<xs:complexContent>
					<xs:extension base="tDocumentedClass">
						<xs:attribute name="name" type="tLiteralName" use="required">
							<xs:annotation>
								<xs:documentation>Name of the enumeration's literal.</xs:documentation>
							</xs:annotation>
						</xs:attribute>
						<xs:attribute name="literalVal" type="xs:int" use="required">
							<xs:annotation>
								<xs:documentation>Enumeration literal integer value.</xs:documentation>
							</xs:annotation>
```

```
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tEnumeration">
        <xs:annotation>
            <xs:documentation>Definition of an enumeration.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="tTitledClass">
                <xs:sequence>
                    <xs:element name="Literal" type="tLiteral" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="name" type="tIec61850Name" use="required">
                    <xs:annotation>
                        <xs:documentation>Name of the enumeration.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="inheritedFrom" type="tIec61850Name" use="optional">
                    <xs:annotation>
                        <xs:documentation>Name of the enumeration which is extended by the current
enumeration.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tEnumerations">
        <xs:annotation>
            <xs:documentation>List of Enumerations introduced by a namespace.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="Enumeration" type="tEnumeration" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Definition of an enumeration.</xs:documentation>
                </xs:annotation>
                <xs:unique name="uniqueLiteralName">
                    <xs:annotation>
                        <xs:documentation>For an Enumeration, there shall not be two Literal sub-
elements with same name.</xs:documentation>
                    </xs:annotation>
                    <xs:selector xpath="nsd:Literal"/>
                    <xs:field xpath="@name"/>
                </xs:unique>
                <xs:unique name="uniqueLiteralVal">
                    <xs:annotation>
                        <xs:documentation>For an Enumeration, there shall not be two Literal sub-
elements with same liiteralVal.</xs:documentation>
                    </xs:annotation>
                    <xs:selector xpath="nsd:Literal"/>
                    <xs:field xpath="@literalVal"/>
                </xs:unique>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="tSubDataAttribute">
        <xs:annotation>
            <xs:documentation>Definition of a Sub Data Attribute (within a constructed data
attribute).</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="tDocumentedClass">
                <xs:attribute name="name" type="tAttributeName" use="required">
                    <xs:annotation>
                        <xs:documentation>Name of the data attribute.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attributeGroup ref="agAttributeTypeAndValues"/>
                <xs:attributeGroup ref="agPresenceCondition"/>
                <xs:attributeGroup ref="agArray"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tConstructedAttribute">
        <xs:annotation>
            <xs:documentation>Definition of a constructed (structured) data attribute.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="tTitledClass">
                <xs:sequence>
                    <xs:element name="SubDataAttribute" type="tSubDataAttribute"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="name" type="tIec61850Name" use="required">
                    <xs:annotation>
                        <xs:documentation>Name of the constructed (structured) data
attribute.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tConstructedAttributes">
        <xs:annotation>
            <xs:documentation>List of Constructed Attributes introduced by a namespace.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="ConstructedAttribute" type="tConstructedAttribute" minOccurs="0"
maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Definition of a constructed (structured) data
attribute.</xs:documentation>
                </xs:annotation>
                <xs:unique name="uniqueSubDataAttribute">
                    <xs:annotation>
                        <xs:documentation>For a ConstructedAttribute, there shall not be two
SubDataAttribute sub-elements with same name.</xs:documentation>
                    </xs:annotation>
                    <xs:selector xpath="nsd:SubDataAttribute"/>
                    <xs:field xpath="@name"/>
                </xs:unique>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="tDataAttribute">
        <xs:annotation>
            <xs:documentation>Definition of a Data Attribute.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="tDocumentedClass">
                <xs:attribute name="name" type="tAttributeName" use="required">
                    <xs:annotation>
                        <xs:documentation>Name of the data attribute.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attributeGroup ref="agAttributeTypeAndValues"/>
                <xs:attribute name="fc" type="tFCAbbreviation" use="required">
                    <xs:annotation>
                        <xs:documentation>Functional constraint of the data
attribute.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attributeGroup ref="agTrgOp"/>
                <xs:attributeGroup ref="agPresenceCondition"/>
                <xs:attributeGroup ref="agArray"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tServiceParameter">
        <xs:annotation>
            <xs:documentation>Definition of the service parameter for control services.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="tDocumentedClass">
                <xs:attribute name="name" type="tAttributeName" use="required">
                    <xs:annotation>
                        <xs:documentation>Name of the data attribute.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attributeGroup ref="agAttributeTypeAndValues"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tSubDataObject">
```