BS EN 62361-100:2016

BSI Standards Publication

Power systems management and associated information exchange — Interoperability in the long term

Part 100: CIM profiles to XML schema mapping

National foreword

This British Standard is the UK implementation of EN 62361-100:2016. It is identical to IEC 62361-100:2016.

The UK participation in its preparation was entrusted to Technical Committee PEL/57, Power systems management and associated information exchange.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

ISBN 978 0 580 73737 4
ICS 33.200

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 September 2016.

Amendments/corrigenda issued since publication

Date	Text affected

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 62361-100

September 2016

ICS 33.200

English Version

Power systems management and associated information exchange - Interoperability in the long term - Part 100: CIM profiles to XML schema mapping (IEC 62361-100:2016)

Gestion des systèmes de puissance et échanges d'informations associés - Interopérabilité à long terme - Partie 100: Mapping des profils CIM avec le schéma XML (IEC 62361-100:2016)

Management von Systemen der Energietechnik und zugehöriger Datenaustausch - Langfristige Interoperabilität - Teil 100: Abbilden von CIM Profilen auf XML Schemen (IEC 62361-100:2016)

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

European foreword

The text of document 57/1704/FDIS, future edition 1 of IEC 62361-100, prepared by IEC/TC 57 "Power systems management and associated information exchange" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 62361-100:2016.

The following dates are fixed:

• latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement	(dop)	2017-04-20
• latest date by which the national standards conflicting with the document have to be withdrawn	(dow)	2019-07-20

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

This document has been prepared under a mandate given to CENELEC by the European Commission and the European Free Trade Association.

Endorsement notice

The text of the International Standard IEC 62361-100:2016 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following notes have to be added for the standards indicated:

IEC 61968	NOTE	Harmonized in EN 61968 series.
IEC 62325	NOTE	Harmonized in EN 62325 series.

Annex ZA
(normative)

Normative references to international publications with their corresponding European publications

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

NOTE 1 When an International Publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

NOTE 2 Up-to-date information on the latest versions of the European Standards listed in this annex is available here: www.cenelec.eu.

Publication	Year	Title	EN/HD	Year
IEC 61968-11	-	Application integration at electric utilities - System interfaces for distribution management - Part 11: Common information model (CIM) extensions for distribution	EN 61968-11	-
IEC/TS 61970-2	-	Energy management system application program interface (EMS-API) - Part 2: Glossary	CLC/TS 61970-2	-
IEC 61970-301	-	Energy management system application program interface (EMS-API) - Part 301: Common information model (CIM) base	EN 61970-301	-
IEC 62325-301	-	Framework for energy market communications - Part 301: Common information model (CIM) extensions for markets	EN 62325-301	-
IEC 62325-450	2013	Framework for energy market communications - Part 450: Profile and context modelling rules	EN 62325-450	2013
W3C XML Schema Part 1	2004	XML Schema - Part 1: Structures	-	-
IETF RFC 3986	2005	Uniform Resource Identifier (URI): Generic Syntax	-	-

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____

POWER SYSTEMS MANAGEMENT AND ASSOCIATED INFORMATION EXCHANGE – INTEROPERABILITY IN THE LONG TERM –

Part 100: CIM profiles to XML schema mapping

FOREWORD

International Standard IEC 62361-100 has been prepared by IEC technical committee 57: Power systems management and associated information exchange.

This is the first edition of the standard.

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

FDIS	Report on voting
57/1704/FDIS	57/1735/RVD

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

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

In this document, the following print types are used:

- Words printed in **Arial Black** apply to terms that are defined as contextual model artefacts in 4.4.2,
- Words printed `Courier New` apply to terms that are used as XML Schema representation (as defined in 4.6) or in XML examples,
- Words printed “between quotes” apply to terms that are used as tokens in the normative clauses or that are defined as CIM artefacts.

A list of all parts of the IEC 62361 series, under the general title: *Power systems management and associated information exchange – Interoperability in the long term*, can be found on the IEC website.

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

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

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

The IEC 62361 series defines standards which address areas of interest that impact multiple standards and provide consistency for implementations.

This part of the IEC 62361 series describes a mapping from CIM profiles to W3C XML Schemas and defines the rules that CIM XML message payloads shall adhere to.

The principle objective of this part of IEC 62361 is to facilitate the exchange of information in the form of XML documents whose semantics are defined by the IEC CIM and whose syntax is defined by a W3C XML schema. This will facilitate the integration of all applications that use the XML Schema message payloads developed by the WGs and implemented independently by different vendors into their systems.

The common information model (CIM) specifies the basis for the semantics for message payload exchanges defined by the IEC. The profile specifications, which are contained in other parts of the IEC 62361 series, specify the content of the message payloads exchanged. The format/syntax of those payloads is specified in this part of IEC 62361.

POWER SYSTEMS MANAGEMENT AND ASSOCIATED INFORMATION EXCHANGE – INTEROPERABILITY IN THE LONG TERM –

Part 100: CIM profiles to XML schema mapping

1 Scope

This part of IEC 62361 describes a mapping from CIM profiles to W3C XML Schemas.

The purpose of this mapping is to facilitate the exchange of information in the form of XML documents whose semantics are defined by the IEC CIM and whose syntax is defined by a W3C XML schema.

2 Normative references

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

IEC 61968-11, *Application integration at electric utilities – System interfaces for distribution management – Part 11: Common information model (CIM) extensions for distribution*

IEC TS 61970-2, *Energy management system application program interface (EMS-API) – Part 2: Glossary*

IEC 61970-301, *Energy management system application program interface (EMS-API) – Part 301: Common information model (CIM) base*

IEC 62325-301*, Framework for energy market communications – Part 301: Common information model (CIM) extensions for markets*

IEC 62325-450:2013*, Framework for energy market communications – Part 450: Profile and context modelling rules*

XML Schema Part 1: Structures Second Edition W3C Recommendation 28 October 2004

IETF RFC 3986 Uniform Resource Identifier (URI): Generic Syntax January 2005

Semantic Annotations for WSDL and XML Schema W3C Recommendation 28 August 2007

3 Terms and definitions

For the purposes of this document, the terms and definitions of IEC TS 61970-2 apply, as well as the following.

NOTE Refer to the International Electrotechnical Vocabulary, IEC 60050, for general glossary definitions.

3.1
artefact
element of a model that represents objects of a given domain and their characteristics

3.2
canonical model
abstract model that represents all the major objects of a given domain (energy, electricity…) with artefacts

3.3
common information model
CIM
canonical model (abstract model) that represents all the major objects in an electric utility enterprise typically needed to model the operational aspects of a utility

Note 1 to entry: CIM is defined in the IEC 61968, IEC 61970 and IEC 62325 series.

Note 2 to entry: This note applies to the French language only.

3.4
contextual model
restricted subset of CIM artefacts

3.5
extensible markup language
XML
markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable

Note 1 to entry: This is defined in the XML Specification produced by the World Wide Web Consortium (W3C).

Note 2 to entry: This note applies to the French language only.

3.6
profile
uniquely named subset of CIM classes, associations and attributes needed to accomplish a specific type of interface

Note 1 to entry: A profile, as used in this document, is defined in IEC 62325-450:2013 and IEC 62361-101[1].

3.7
resource description format
RDF
family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model

Note 1 to entry: This term has come to be used as a general method for conceptual description or modelling of information that is implemented in web resources, using a variety of syntax formats.

Note 2 to entry: This note applies to the French language only.

3.8
semantic annotation for WSDL and XML Schema
SAWSDL
set of extension attributes for the Web Services Description Language (WSDL) and XML Schema definition language

Note 1 to entry: This note applies to the French language only.

1 Under consideration.

3.9
unified modelling language
UML
formal and comprehensive descriptive language with diagramming techniques used to represent software systems, from requirements analysis, through design and implementation, to documentation

Note 1 to entry: UML is a standard defined by the Object Management Group (OMG). UML is used to describe CIM.

Note 2 to entry: This note applies to the French language only.

3.10
uniform resource indicator
URI
string of characters used to identify a name or a resource, enabling interaction with representations of the resource over a network (typically the World Wide Web) using specific protocols

Note 1 to entry: Schemes specifying a concrete syntax and associated protocols define each URI.

Note 2 to entry: This note applies to the French language only.

3.11
XML Schema
family of World Wide Web Consortium (W3C) specifications, used to define the structure, content, and semantics of eXtensible Markup Language (XML) files

Note 1 to entry: XML Schemas are generally found in files with an “xsd” extension. XSD files are used to define inter-application messages.

3.12
Web Ontology Language
OWL
family of knowledge representation languages for authoring ontologies, characterised by formal semantics and RDF/XML-based serializations for the Semantic Web

Note 1 to entry: OWL is endorsed by the World Wide Web Consortium (W3C).

Note 2 to entry: This note applies to the French language only.

4 System context

4.1 Profiling process

The profiling process aim is to define a syntactic model that will govern instance data that are exchanged in a given business context and whose semantic is defined by a canonical model (like CIM). The profiling process is in simple form a two steps process:

- Defining a contextual model that is a subset of the canonical model (subset that could include some restrictions). In this document, there is no assumption about the rules that are used to complete this step, but the contextual model artefacts are described in Table 1.
- Generating a syntactic model in the form of an XML Schema with a defined mapping of contextual model artefacts: this is the purpose of this standard IEC 62361-100.

IEC 62361-100 defines how a contextual model artefact is mapped to XML Schema artefacts (like `element`, `simple` and `complex types`). It does not define a mapping from canonical model artefacts to XML Schema ones, but it keeps the fact that there is relation between these two artefacts.

4.2 CIM

CIM is a canonical model that represents all the major objects in an electric utility enterprise typically needed to model the operational aspects of a utility. This model includes public classes and attributes for these objects, as well as the relationships between them. Classes, attributes, relationships and attribute types like "Primitive", "enumeration", "CIMdatatype" and "Compound" are the main CIM artefacts.

CIM is defined by IEC standards IEC 61968-11, IEC 61970-301 and IEC 62325-301.

The CIM may be augmented with project or application-specific extensions. In that case, the references to the CIM in this subclause can be read as CIM with extensions.

4.3 Contextual model

The concept of a contextual model is borrowed from the UN/CEFACT modelling approach and may be used in CIM standards formation. The contextual model may be any one of several formats including OWL or a UML subset package.

No specific contextual modelling language is assumed by this specification. However, the artefacts defined in Table 1 are used in this document when referring to the contextual model and are assumed to be capable of expression in whichever language is used.

The mapping specifications (see Clause 5) apply to these contextual model artefacts which could be represented in a number of languages. Two possible representations are given in the appendices.

4.4 Contextual model artefacts

4.4.1 Contextual model artefacts and CIM subset

In Table 1, contextual artefacts are defined in relation to CIM artefacts. Here, the term subset is used: a contextual artefact is a subset of some CIM artefact. Subset means that a contextual artefact could have the same characteristics as its CIM counterpart or a subset of these characteristics. Examples:

- “IdentifiedObject” class in CIM has four attributes (“mRID”, “aliasName”, “name” and “description”) and one association “Names”, i.e. its characteristics. “IdentifiedObject” **structured class** in contextual model could have the same characteristics as its CIM counterpart or just some of them: so “IdentifiedObject” contextual artefact is defined as a subset of “IdentifiedObject” CIM artefact.
- “name” attribute of CIM “IdentifiedObject” class has two characteristics: a cardinality that is optional and a type that is a string. In contextual model, “name” **simple property** of “IdentifiedObject” **structured class** could have the same characteristics as its CIM counterpart or some more restricted ones: example, cardinality of “name” could be restricted to mandatory and/or string length could be defined. So “name” contextual artefact is defined as a subset of “name” CIM artefact.
- “Names” association end role name of CIM “IdentifiedObject” has one characteristic: a cardinality that is 0 to many. In contextual model, “Names” **object property** of “IdentifiedObject” **structured class** could have the same characteristic as its CIM counterpart or a more restricted one: example, cardinality of “Names” could be restricted to 1 or 1 to many. So “Names” contextual artefact is defined as a subset of “Names” CIM artefact.

4.4.2 Contextual model artefacts definition

Contextual model artefacts are listed and defined in Table 1.

Table 1 – Contextual model artefacts

Contextual model artefact	Definition
Structured class	subset of a CIM class not stereotyped with "enumeration", "Primitive", "CIMDatatype" or "Compound". A structured class may have zero or more object properties, compound properties and simple properties. Any subclass of a structured class is also a structured class.
Superclass	relative to a given structured class, a more general structured class whose extent is a superset of the given structured class.
Subclass	relative to a given structured class, a more specific structured class whose extent is a subset of the given structured class.
Root class	structured class that may have standalone instances which are not the referent of any object property. A contextual model may assign cardinality bounds to a root class limiting the number of standalone instances that may occur.
Union class	subset of a non-stereotyped CIM superclass defined as a union of (some of) its subclasses. Each member of the union is defined as a structured class. Each of these is a subclass of a single, given CIM class. An instance of a union is an instance of one of its constituent structured classes. Example: in CIM, “RegisteredResource” is a super class of “RegisteredLoad”, “RegisteredTie” and “RegisteredGenerator”. In contextual model, “RegisteredResource” could be a super class of some of these subclasses. When defined as a union, “RegisteredResource” defined the set of the subclasses (“RegisteredLoad”, “RegisteredTie”…) that are going to be used as the referent classes for the “RegisteredResource” **object property** that in this case will be a **union object property** (see below). Note: this feature is used to get all the elements representing subclasses instances in a random order.
Compound class	subset of a CIM class defined as "Compound" with additional restrictions. An instance of a compound class is a structured value. It has one or more properties, but it has no identity distinct from the combination of its property values.
Basic type	CIM class defined as a "Primitive" (include "Integer", "Decimal", "Boolean", "Duration", "DateTime", "Date", "Time", "Float", "String"). A subset of the CIM “Float” class defined as a “Primitive” and marked as “Single” or "Double" precision. A subset of the CIM “String” class defined as a “Primitive” and marked as “normalized”, “token”, “NMTOKEN”, “Name”, “NCName” and “anyURI” . A basic type may be used directly in a contextual model without further definition. The value range of each basic type is assumed to be that of the XML Schema Part II Datatype `integer, decimal, boolean, duration, datetime, date, time, float, double, string, normalizedString, token, MNTOKEN, Name, NCName` and `anyURI` respectively.
Simple type	subset of a CIM class defined as “Primitive” with additional restrictions. As defined above, the value range of such a CIM class is assumed to be one of the XML Schema Part II datatypes defined above. The additional restrictions narrow this value range by defining one or more facets for that datatype (example: “TwentyFourChar_String” is a string whose maximum length is 24 characters). A simple type instance does not have simple properties or object properties and has no identity distinct from its value.

Contextual model artefact	Definition
Data type	subset of a CIM class defined as "CIMDatatype" with additional restrictions. A **data type** is a class whose instances carry a value and other properties that give meaning to this value. Data type value and other data type properties could be restricted by additional constraints. An instance of a **data type** class is a structured value. It has one or more properties, but it has no identity distinct from the combination of its property values.
Enumeration class	subset of an "enumeration" CIM class.
CodeList class	subset of an "enumeration" CIM class and marked as "CodeList". Each instance of the enumeration is associated to a “code” whose type is one of the "**Basic type**".
Simple property	subset of a CIM class attribute with additional restrictions. The type of a **simple property** is a **Simple type**, a **Basic type**, a **Data type** or an **Enumeration** Class.
Compound property	subset of a CIM class attribute whose referent is a class defined as a "Compound".
Object property	subset of a CIM association with additional restrictions and a specific direction from referring class to referent class. The referent of an **object property** is an instance of a **structured class**. The restrictions may narrow the referring or referent classes or place bounds on the cardinality of the object property.
By-reference object property	subset of a CIM association, as per object property, defined as **by-reference**. The referent of a **by-reference object property** is either an instance of a structured class or an external instance. An external instance is assumed to exist but is not described in the present message. Pragmatically, a"**by-reference object property** is implemented by quoting the referent's identifier (example "mRID").
Union object property	object property defined as **union** whose referent class is a super class or an object property whose referent class is a union class. In CIM, “ResourceCapacity” has an association with “RegisteredResource”, super class of “RegisteredLoad”, “RegisteredTie” and “RegisteredGenerator”. The association has two end role names: “ResourceCapacity” and “RegisteredResource”. In contextual model, “ResourceCapacity” could have the **object property** “RegisteredResource” whose referent class is “RegisteredResource”. If this object property is marked as **union** or if the “RegisteredResource” referent class is marked as **union**, then the “RegisteredResource” **object property** is a **union object property**. Note: this feature is used to get all the elements representing subclasses instances in a random order.
Exclusive property group	restriction on a **structured class** with respects to a group of properties such that only one of the properties may appear in a given instance of the class.
BasedOn property	relation between a contextual model artefact (like **structured class**, **property**, **simple type** or **data type**) with its corresponding CIM artefact (like "class", "attribute", "association", "Primitive", "enumeration", "CIMDatatype", "Compound").
Documentation	prose description accompanying a definition in the CIM or the contextual model.
Categorized documentation	prose description accompanying a definition in the CIM or in the contextual model together with some classifying properties which indicate the category and purpose of the description.

Contextual model artefact	Definition
Stereotype	identifier associated with a contextual model class or property that qualifies its usage or semantics, but not its XML Schema mapping. The meaning of each stereotype must be provided in documentation accompanying the contextual model.

4.5 Mapping contextual model to XML schema

4.5.1 General

The mapping determines:

- a single, standalone XML schema for a given contextual model;
- the syntax of the instance XML documents to be exchanged;
- the relationship between definitions in the XML schema and definitions in the contextual model;
- the relationship between elements in the XML documents exchanged and the definitions in the CIM.

The mapping is applied at design time to map a CIM profile to a W3C XML schema.

In this mapping, a profile defines the semantics of a single type of message payload that will be encoded in XML.

Therefore:

- The syntax or semantics of any headers that may be added to the instance documents when they are exchanged is not specified.
- Both the contextual model and its mapped XML schema are design artefacts and are not necessarily required at the time instance XML documents are exchanged. But the corresponding XML schema must be agreed and shared before the exchange.
- The method used to exchange instance XML documents is not specified.

4.5.2 Traceability

In this mapping, the relationships, between elements in the XML documents exchanged and the definitions in the canonical model of which the contextual model is a subset, are expressed. To keep track of these relationships, the mapping uses semantic annotation as defined in "Semantic Annotations for WSDL and XML Schema W3C Recommendation". The mapping to these semantic annotations is done with the contextual model artefact "**BasedOn**" property.

4.6 XML Schema Representation

The XML Schema mappings are presented using the “XML Representation Summary Notation” used in the W3C specification: "XML Schema Part 1: Structures Second Edition". The following example is abstracted from section 1.3 of this W3C specification:

```
<example
  count = integer
  size = (large | medium | small): medium>
  Content: (annotation, (all | any*))
</example>
```

The notation consists of an outline of an XML Schema construct with the following conventions:

- Mandatory attributes are shown in bold, e.g. **`count`**
- Optional attributes are shown in standard, e.g. `size`
- Literal attribute values are shown in italics e.g. *`medium`*
- Alternatives attribute values are shown in brackets separated by vertical bars. e.g. (*`large`* `|` *`medium`* `|` *`small`*) and if there is a default value it is shown after a colon, e.g.: `medium`
- The content of the schema element is introduced by `Content:`
- Content grammar is enclosed in brackets with a separating comma for concatenation or vertical bar for alternatives. e.g. `(annotation, (all | any*))`
- The Kleenex operators; **`?`**, **`+`**, and **`*`** are used for at most one, at least one, and any number of repetitions respectively
- Simple words in plain face refer to definitions elsewhere. (Unlike the XML Schema Recommendation, these are not hyper-linked in this document.) e.g. `annotation`

4.7 Namespaces

XML namespace definitions are not explicitly shown in the mapping definitions. The choice of namespace prefixes is outside the scope of the mapping specification. However, for the purpose of this document and examples it is assumed that:

- The default namespace is the same as the schema's target namespace, which is designated as `namespace-uri` below
- The prefix `xs:` stands for the standard XML Schema namespace, `http://www.w3.org/2001/XMLSchema`
- The prefix `sawsdl:` stands for the Semantic Annotations for WSDL namespace `http://www.w3.org/ns/sawsdl`
- The prefix `p:` stands for profile extension name space `http://iec.ch/TC57/<year>/<Profile>`

5 Mapping specifications

5.1 General

Subclauses 5.2 through 5.17 define the mapping of a generic contextual model to an XML Schema.

5.1.1 Example

Figure 1 serves as example to illustrate certain constructs introduced in Subclauses 5.2 through 5.17. It shows one kind of mapping using an envelope and an alphabetical order for the properties. Other schema designs are possible such as not using an envelope (see 5.2.3.3) or not using an alphabetical order (see 5.16.1)

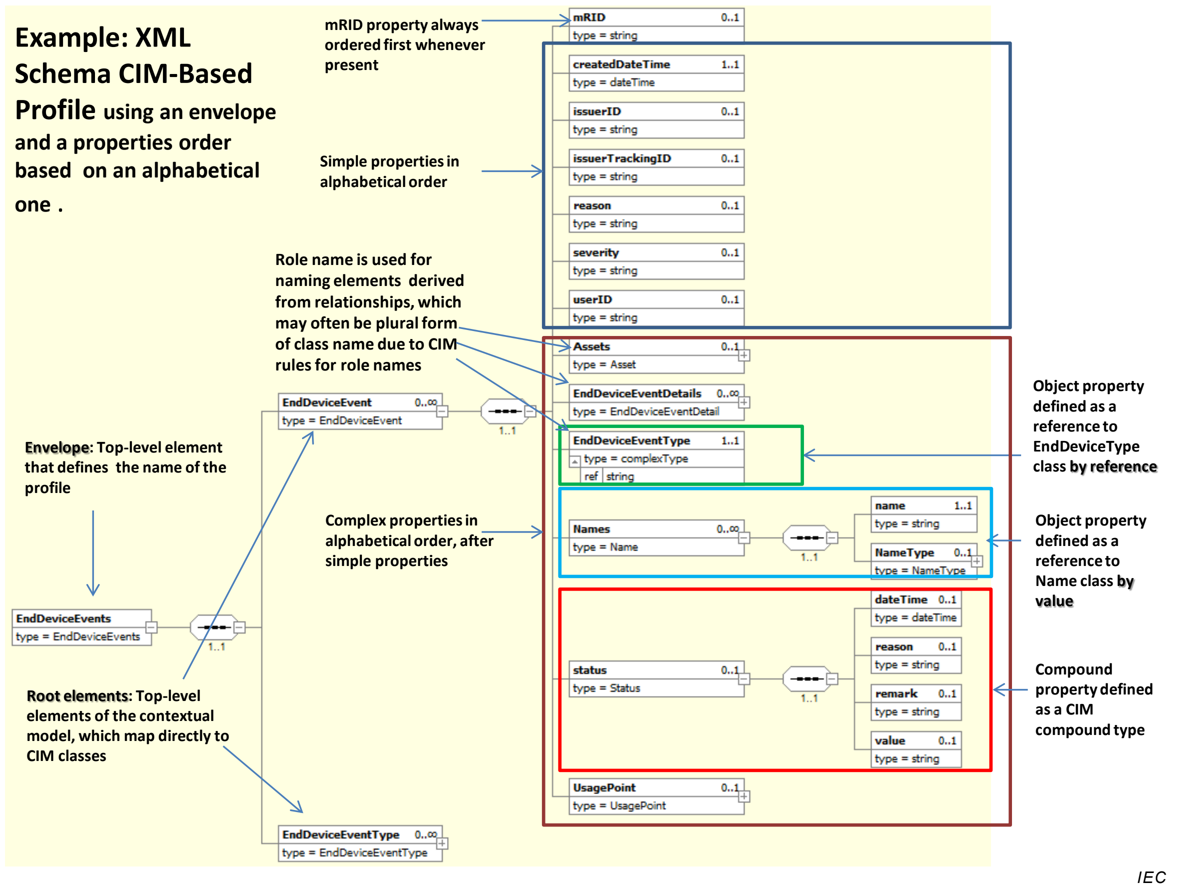
Example: XML Schema CIM-Based Profile using an envelope and a properties order based on an alphabetical one .
mRID property always ordered first whenever present
Simple properties in alphabetical order
Role name is used for naming elements derived from relationships, which may often be plural form of class name due to CIM rules for role names
Envelope: Top-level element that defines the name of the profile
Complex properties in alphabetical order, after simple properties
Root elements: Top-level elements of the contextual model, which map directly to CIM classes
Object property defined as a reference to EndDeviceType class by reference
Object property defined as a reference to Name class by value
Compound property defined as a CIM compound type
EndDeviceEvents
type = EndDeviceEvents
1..1
EndDeviceEvent 0..∞
type = EndDeviceEvent
EndDeviceEventType 0..∞
type = EndDeviceEventType
mRID 0..1
type = string
createdDateTime 1..1
type = dateTime
issuerID 0..1
type = string
issuerTrackingID 0..1
type = string
reason 0..1
type = string
severity 0..1
type = string
userID 0..1
type = string
Assets 0..1
type = Asset
EndDeviceEventDetails 0..∞
type = EndDeviceEventDetail
EndDeviceEventType 1..1
type = complexType
ref string
Names 0..∞
type = Name
name 1..1
type = string
NameType 0..1
type = NameType
status 0..1
type = Status
dateTime 0..1
type = dateTime
reason 0..1
type = string
remark 0..1
type = string
value 0..1
type = string
UsagePoint 0..1
type = UsagePoint
IEC

Figure 1 – Example XML Schema CIM-based profile

5.1.2 Mapped name

The mapping process defines how a contextual model artefact is mapped to an XSD artefact. All artefacts have a name. One of the steps of the mapping is to define the name of the XSD artefact in relation with the contextual model artefact name: in the following clause, this is defined as the mapped name of the contextual model artefact. Usually, the mapped name is the name of the contextual model artefact except when the changing name rule described in 5.17 is applied.

5.2 Profile mapping

5.2.1 General

A single profile is mapped to a single XML Schema with the following form:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
      (xmlns:prefix = namespace-uri)*
      targetNamespace = namespace-uri
      elementFormDefault = qualified
      attributeFormDefault = unqualified
      version = version-string >
      Content: (annotation, ((envelope-elem, envelope-type) |
      (root-elem, root-type)), (complex-type | simple-type)*)
</xs:schema>
```

`xmlns` attributes specify the namespaces that are used in the schema:

- The namespaces defined in 4.7 are the ones that are always used in the schema,
- It is a good practice to include the name space of the CIM canonical model that is used for defining the contextual model. Note: in case of a CIM extended canonical model, the extensions are defined in other namespaces that could be included too.

5.2.2 Namespace and version

The `namespace-uri` identifies the profile. The namespace for each IEC standard profile is allocated by the IEC in the `iec.ch` domain.

The `version` attribute may be used by an implementer or during the definition of a draft standard profile.

The form of the `version` attribute in these applications is outside the scope of this specification.

5.2.3 Schema top level element

5.2.3.1 General

The schema top level element should be either an `Envelope` or a `Root` element.

5.2.3.2 Envelope element

When used, the `Envelope` element is an element that characterizes the profile (it is the name of the profile). It is used as the top level element of the schema, when the name of the profile is used to define the payload. It is not defined in the contextual model and therefore must not have the name used by any class from the contextual model. In Figure 1, it is the `EndDeviceEvents` element.

The `envelope-elem` is the single top-level element definition in the schema, as follows:

```
<xs:element
      Name = envelope-name
      Type = envelope-name>
</xs:element>
```

The `envelope-name` is chosen such that the combination of `namespace-uri` and envelope-name is unique among all published XML schema.

The `envelope-type` is a `global` type definition for the `envelope` as follows:

```
</xs:complexType>
      name = envelope-name>
      <xs:sequence>
            Content: (root-elem*)
      </xs:sequence>
</xs:complexType>
```

The content consists of one `local` element definition, `root-elem`, for each root class in the profile in alphanumeric order by element name or in an order defined by the business context

The contextual model could define several root classes. In Figure 1, for example, it is the **EndDeviceEvent** and **EndDeviceEventType** elements.

5.2.3.3 Root element

Root class is the top element of the contextual model (there could be several root classes in a contextual model). Usually, it is used as a schema top level element, according to a business context, when the payload is not defined by the profile name.

The `root-elem` is defined as follows:

```
<xs:element
      name = root-name
      type = root-name
      minOccurs = min
      maxOcurs = max>
</xs:element>
```

The `root-name` is the mapped name of the root class in the contextual model.

The value `min` is the minimum cardinality of the class as defined in the contextual model. The value `max` is the maximum cardinality defined in the contextual model. (If `max` or `min` equals 1, the corresponding `maxOccurs` or `minOccurs` attribute can be omitted, according to XML schema specification.)

5.2.4 Types

`complex-type` and `simple-type` are the XML "complexTypes" and "simpleTypes" defined in the following clauses.

5.2.5 Semantic annotation

In order to insure traceability, for each XML schema element, `complexType` and `simpleType` defined in the following clauses, a semantic annotation `modelReference` attribute is defined as an absolute URIRef that is defined as follows:

- the namespace URI of the information model,
- the character #,
- and the name of the corresponding CIM artefact (like class, attribute, association, datatypes....

The name of the corresponding CIM artefact is given by the **BasedOn** relationship. The attribute definition is as follows:

```
sawsdl:modelReference = (class-ref | prop-ref | type-ref | enum-ref |
codelist-ref | enum-value-ref)
```

where,:

- `class-ref` is a URIRef designating the structured or compound class (CIM class) of which this contextual model class is a subset. In Figure 1, `Name` is a `complexType`, mapped from the contextual **structured class** "**Name**", that itself is a subset of the CIM class "Name", so `class-ref` for `complexType` `"Name"` is for example: `"http://iec.ch/TC57/2013/Cim-schema-cimXX#Name"`.
- `prop-ref` is the URIRef designating the property definition (CIM attribute or association) of which this contextual model property is a subset. In Figure 1, `name` is an `element`, mapped from the contextual **simple property** "**name**", that itself is a subset of CIM attribute "name", so `prop-ref` for `element` `"name"` is for example: `http://iec.ch/TC57/2013/Cim-schema-cimXX#Name.name`. In Figure 1, `Names` is

an `element`, mapped from the contextual **object property "Names"**, that is a subset of CIM "Names" end role name, so `prop-ref` for `element` "`Names`" is for example: "`http://iec.ch/TC57/2013/Cim-schema-cimXX#IdentifiedObject.Names`".

- `type-ref` is a URIRef designating the class (CIM "Primitive" or "CIMDatatype" class) of which this type is a subset. In Figure 1, `xs:dateTime` is the type of `element` `"createdDateTime"`, `xs:dateTime` is mapped from contextual **basic type "DateTime"**, which corresponds to CIM Primitive "DateTime" so `type-ref` for `xs:dateTime` is for example: "`http://iec.ch/TC57/<year>/Cim-schema-cimXX#DateTime`".
- `enum-ref` is a URIRef designating the enumeration (CIM "enumeration") of which this contextual model **enumeration** is a subset. Example: `UsagePointConnectedKind` is a `simpleType`, mapped from contextual **enumeration class "UsagePointConnectedKind"**, which is a subset of CIM enumeration "UsagePointConnectedKind", so `enum-ref` for `UsagePointConnectedKind` element is for example: "`http://iec.ch/TC57/<year>/Cim-schema-cimXX#UsagePointConnectedKind`".
- `enum-value-ref` is the URIRef designating the "enumeratedLiteral" of the CIM "enumeration" of which this contextual model **enumeration** is a subset. Example: `connected` is one of the "`UsagePointConnectedKind"` `simpleType` enumeration values, mapped from contextual **"connected"** enumeration value of contextual UsagePointConnectedKind **enumeration class**, which is one of the subset of CIM "UsagePointConnectedKind" enumerated literals, so `enum-value-ref` for "connected" enumeration value is for example: "`http://iec.ch/TC57/<year>/Cim-schema-cimXX#UsagePointConnectedKind.connected`".
- `codelist-ref` is a URIRef designating the "enumeration" of which this contextual model class is a subset.

URIRef are compliant with IETF RFC 3986.

5.3 Structured classes

Each **structured class** in the contextual model, including each **root class** is mapped to a `global complex-type` definition as follows:

```
<xs:complexType
      name = class-name
      sawsdl:modelReference = class-ref>
      Content: (annotation, (whole-class | derived-class))
</xs:complexType>
```

The `class-name` is the mapped name of the contextual model class.

The `class-ref` is a URIRef designating the class (CIM class) of which this contextual model class is a subset.

The `annotation` content carries documentation from the contextual model and the CIM. This is detailed in the Annotation section.

The remaining content depends on whether the contextual model class is a subclass that has a contextual model super class. If the contextual class is a **subclass**, the `derived-class` form applies as follows:

```
<xs:extension
      base = super-name>
      Content: whole-class
</xs:extension>
```

The `super-name` is the mapped name of the class's **super class** defined in the contextual model.

The whole-class form is as follows:

```
<xs:sequence>
      Content:((simple-elem | typed-elem | ref-elem | union-elem
       | compound-elem | exclusive-elem)*)
</xs:sequence>
```

The content consists of one `local` element definition for each property defined in the contextual model as a member of the class.

The XML element definitions appear in a given order as specified in 5.16 . However, if one of the XML elements is the result of the mapping of contextual model **mRID simple property** this XML element appears first.

The form of each XML element definition depends on the type of the property and is described in the following clauses.

5.4 Compound classes

Each **compound class** is mapped to a `global complex-type` definition as follows:

```
<xs:complexType
      name = class-name
      sawsdl:modelReference = class-ref>
      Content: (annotation, whole-compound)
</xs:complexType>
```

The `class-name` is the mapped name of the contextual model compound class.

The `class-ref` is a URIRef designating the class (CIM class) of which this contextual model compound class is a subset.

The `annotation` content carries documentation from the contextual model and the CIM. This is detailed in the Annotation section.

The `whole-compound` form is as follows:

```
<xs:sequence>
      Content:((simple-elem | typed-elem | compound-elem )*)
</xs:sequence>
```

The content consists of one `local` element definition for each property defined in the contextual model as a member of the compound class.

The XML element definitions appear in a given order as specified in 5.16.

The form of each XML element definition depends on the type of the property and is described in the following clauses.

5.5 Basic types

The **basic types** listed in Table 2 have fixed mappings to W3C XML Schema 1.1 Part II Datatypes as shown. No definitions are created for these types in the XML schema.

Table 2 – Basic Types

Basic Type	Mapped Type
Boolean	`xs:boolean`
Date	`xs:date`
Datetime	`xs:datetime`
Decimal	`xs:decimal`
Duration	`xs:duration`
SingleFloat	`xs:float`
DoubleFloat	`xs:double`
Integer	`xs:integer`
String	`xs:string`
Normalized String	`xs:normalizedString`
Token String	`xs:token`
NMToken String	`xs:NMTOKEN`
Name String	`xs:Name`
NCName String	`xs:NCName`
AnyURI String	`xs:anyURI`
Time	`xs:time`

5.6 Simple types

5.6.1 Mapping rules

Each **simple type** defined in the contextual model is mapped to a global `simple-type` definition as follows:

```
<xs:simpleType
      name = type-name
      sawsdl:modelReference = type-ref>
      Content: (annotation, base-type)
</xs:simpleType>
```

The `type-name` is the mapped name of the contextual model **simple type**.

The `type-ref` is a URIRef designating the class (CIM class) of which this type is a subset.

The `base-type` is as follows:

```
<xs:restriction
      base = xstype
      Content: (facet*)
</xs:restriction>
```

The `xs:type` is the qualified name of an *XML Schema Part II Datatype* specified in the contextual model (see 5.5).

Each `facet` is an XML Schema facet restriction corresponding to a facet restriction given in the contextual model for this **simple type**.

The `facet` definition has the form:

```
<xs:facet-name
      value = facet-value>
</xs:facet-name>
```

The `facet-name` is one of the allowed facet name as defined by *XML Schema Part II datatype* and described in Table 3.

Example:

```
<xs:simpleType
      name="NonNegativeInteger"
      sawsdl:modelReference="http://iec.ch/TC57/2013/CIM-schema-
cimXX#Integer">
      <xs:restriction
            base="xs:integer">
                  <xs:minInclusive
                        value="0"/>
      </xs:restriction>
</xs:simpleType>
```

This shows the `minInclusive` facet which takes a non-negative integer as the `facet-value`.

5.6.2 Possible facets

The possible facets are listed for each **basic type** in Table 3.

Table 3 – Facets

Basic Type	Facet
Boolean	*No facet*
String Normalized String Token String NMToken String Name String NCName String AnyURI String	length
	minLength
	maxLength
	pattern
	whiteSpace
	enumeration
Integer	totalDigits
	minInclusive
	maxInclusive
	minExclusive
	maxExclusive
	enumeration
SingleFloat DoubleFloat DateTime Date Time Duration	minInclusive
	maxInclusive
	minExclusive
	maxExclusive
	enumeration
	pattern
Decimal	totalDigits
	fractionalDigits
	minInclusive
	maxInclusive
	minExclusive
	maxExclusive
	enumeration

These facets are defined in W3C XML Schema 1.1 Part II Datatypes.

5.7 Data Types mapping

Each **data type** defined in the contextual model is mapped to a global `complexType` definition as follows:

```
<xs:complexType
      name= datatype-name
      sawsdl:modelReference = datatype-ref>
      Content: (annotation, simple-content)
</xs:complexType>
```

The `datatype-name` is the mapped name of the **data type** name.

The `datatype-ref` is a URIRef of the "CIMDatatype" class (CIM class) of which this contextual model datatype is a subset.

EXAMPLE: in CIM "ActivePower" is a "CIMDatatype". In a contextual model a corresponding artefact is **ActivePower data type**, that will be mapped to a complexType whose `datatype-name` will be "ActivePower" and `datatype-ref` will be "http://iec.ch/TC57/2013/CIM-schema-cimXX#ActivePower".

The `simple-content` is as follows:

```
<xs:simpleContent>
      <xs:extension
            base = base-type>
            Content: (attribute-elem*)
</xs:simpleContent>
```

The "`base-type`" depends on the referent of the datatype value property as follows:

- If the referent is one of the **basic types** listed in 5.5 the `base-type` is the given XML Schema Part II **data type** name.
- If the referent is an **enumeration** class then the `base-type` is its mapped name. This designates the corresponding type definition described in 5.8.
- If the referent is one of the **basic types** listed in 5.5 with additional restrictions, then the `base-type` is a global simple type whose mapped name is the **data type** name appended with the term "-base". The definition of the simple type is constructed as described in 5.6, except that the `annotation` and `type-ref` are omitted.

Example: in CIM, "ActivePower" is a "CIMDatatype", that has a "value" attribute whose value space is defined by a type that is the CIM "Primitive" Float. In contextual model, the corresponding artefact is "ActivePower" **data type**, that has a "value" property and some other properties ("unit" and "multiplier"). The mapping of the "value" property depends on its type. "Value" property type could be of three kinds, resulting in three kinds of mapping:

- "value" property value space or type is the **basic type** "SingleFloat", the simpleContent `base-type` will be "xs:float".
- "value" property value space or type is restricted to a named **enumeration** for example "ActivePowerValueKind", the simpleContent `base-type` will be "ActivePowerValueKind".
- "value" property value space or type is restricted to a "SingleFloat" that is positive (facet "minInclusive" = "0"). The simpleContent `base-type` will be "ActivePower-base". This base-type is a `simpleType` whose name is "ActivePower-base". The `simpleType` is a restriction of `xs:float` with a "minInclusive" restriction whose value is "0" (see simpleType mapping clause and example below).

The `attribute-elem` definition (as a local attribute definition) is as follows:

```
<xs:attribute>
      name = prop-name
      type = type-name
      default = string
      fixed = string
      use = (required | optional | prohibited): optional>
      Content: annotation
</xs:attribute>
```

The `prop-name` is the mapped name of the datatype property (other than the "value" one). Example "unit", "multiplier"…

The form of the `type-name` depends on the referent of the **data type** property as follows:

- If the referent is one of the **basic types** listed in 5.5, the `type-name` is the given XML Schema Part II datatype name.

- If the referent is an **enumeration** class then the `type-name` is its mapped name. This designates the corresponding type definition described in 5.8.

The `default` string is the value of the default value set for the contextual model **data type** property.

The `fixed` string is the fixed value set for the contextual model **data type** property.

The two `fixed` and `default` attributes are exclusive if they are used.

The `use` value is `optional` if the **data type** property is optional, `required` if it is mandatory or `prohibited` if it is not allowed in the contextual model. In the first case `optional` it could be omitted.

Example: in contextual model "ActivePower" **data type** has a value property whose value space is a "SingleFloat" **basic type** restricted to positive value. "ActivePower **data type** has also two other properties "unit" and "multiplier". "Unit" property has a fixed value "W", and "multiplier" has a default value of "none". The result mapping will be the following.

```
<xs:simpleType
      name="ActivePower-base"
      <xs:restriction
            base="xs:float">
                  <xs:minInclusive
                        value="0"/>
      </xs:restriction>
</xs:simpleType>

<xs:complexType
      name= "ActivePower"
      sawsdl:modelReference = "http://iec.ch/TC57/2013/CIM-schema-
cimXX#ActivePower">
      <xs:simpleContent>
            <xs:extension
                  base = "ActivePower-base">
                  <xs:attribute
                        name = unit
                        type = UnitSymbolKind
                        fixed = W
                        use = (required, optional, prohibited) optional>
                  </xs:attribute>
                  <xs:attribute
                        name = multiplier
                        type = UnitMultiplierKind
                        default = none
                        use = (required, optional, prohibited) optional>
                  </xs:attribute>
      </xs:simpleContent>
</xs:complexType>
```

5.8 Enumeration classes mapping

Each **enumeration class** defined in the contextual model is mapped to an `enum-type` global definition as follows:

```
<xs:simpleType
      name= enum-name
      sawsdl:modelReference = enum-ref>
      Content: (annotation, enum-values)
</xs:simpleType>
```

The `enum-name` is the mapped name of the contextual model class.

Then `enum-ref` is a URIRef designating the enumeration (CIM enumeration) of which this contextual model **enumeration** is a subset.

The `enum-values` is as follows:

```
<xs:restriction
      base= xs:string>
      Content: (enum-value*)
</xs:restriction>
```

Each `enum-value` represents one value of the enumeration defined in the contextual model as follows:

```
<xs:enumeration

      value = enum-value

      sawsdl:modelReference = enum-value-ref>

      Content: annotation

</xs:enumeration>
```

The `enum-value` is the mapped name of the enumeration value.

The `enum-value-ref` is the URIRef designating the enumeration value of the "enumeration" of which this contextual model **enumeration** is a subset.

5.9 CodeList classes mapping

Each **CodeList class** defined in the contextual model is mapped to a `codelist-type` global definition as follows:

```
<xs:simpleType
      name= codelist-name
      sawsdl:modelReference = codelist-ref>
      Content: (annotation, codelist-values)
</xs:simpleType>
```

The `codelist-name` is the mapped name of the contextual model class.

Then `codelist-ref` is a URIRef designating the "enumeration" or "CodeList" of which this contextual model class is a subset.

The `codelist-value` is as follows:

```
<xs:restriction
      Base = codelist-type>
      Content: (codelist-value*)
</xs:restriction>
```

The `codelist-type` is the mapped name of the code type.

Each codelist-value represents one value of the code defined in the contextual model as follows:

```
<xs:enumeration
      value = codelist-value
      sawsdl:modelReference = enum-value-ref>
      Content: annotation
</xs:enumeration>
```

The codelist-value is the mapped name of the code value.

The enum-value-ref is the URIRef designating the corresponding enumeration value in the "CodeList" class of which this **CodeList class** is a subset.

The codelist-value annotation is defined as follows (see 5.14.3):

```
<xs:annotation>
      Content: catdocument-elem
</xs:annotation>
```

The catdocument-elem is defined as follows as follows:

```
<xs:documentation
      xml:lang = language
      p:category: enumValue
      p:notes: xs:string>
      content: xs:string
</xs:documentation>
```

p:notes is the description associated to the value of the enumeration.

content is the mapped name of the associated value of the enumeration.

5.10 Simple properties mapping

Each **simple property** in the contextual model is mapped (as a local element definition) to a simple-elem as follows:

```
<xs:element
      name = prop-name
      type = type-name
      minOccurs = min
      sawsdl:modelReference = prop-ref>
      Content: annotation
</xs:element>
```

In Figure 1, examples of **simple properties** are the elements "createdDateTime" (whose type is a “Datetime” **basic Type**, here mapped to `xs:datetime`) and "reason" (whose type is a “String” **BasicType**, here mapped to `xs:string`).

The `prop-name` is the mapped name of the property.

The value `min` for minOccurs attribute is 0 if the property is optional in the contextual model or 1 if mandatory. (In the latter case the `minOccurs` attribute can be omitted, according to XML Schema specification.)

The `prop-ref` is the URIRef designating the property definition (CIM property) of which this contextual model property is a subset – Example in Figure 1: "createdDateTime" is an element of "EndDeviceEvent" element. In contextual model, "createdDateTime" is a **simple property** of "EndDeviceEvent" class. This **simple property** is related to a CIM corresponding attribute. In CIM, the corresponding attribute is “createdDateTime” from “ActivityRecord” class and this attribute is inherited by “EndDeviceEvent”. Thus the prop-ref is the URIRef: `http://iec.ch/TC57/2013/Cim-schemacimXX#ActivityRecord.createdDatetime`.

The form of the `type` attribute depends on the referent of the simple property as follows:

- If the referent is one of the **basic types** listed in 5.5, there is no simpleType definition in the XML schema and the `type-name` is the given XML Schema Part II datatype name. Example: in contextual model, "issuerID" type is a “String” **basic type**. So, the mapped `type-name` is `xs:string`.
- If the referent is a **simple type** then the `type-name` is its mapped name. This designates the corresponding type definition described in 5.6. Example: if in contextual model, "issuerID" type was a simple type name “Char24_String” (for example a String of 24 characters), the mapped `type-name` will be “`Char24_String`”.
- If the referent is a **data type** then the `type-name` is its mapped name. This designates the corresponding type definition described in 5.7. Example: in CIM, “EndDeviceInfo" attribute “ratedVoltage” has CIMDatatype “Voltage” for type. In contextual model, “ratedVoltage” **simple property** will have “Voltage” **data type** for type. Thus, the mapped `type-name` will be “`Voltage`”.
- If the referent is an **enumeration class** then the `type-name` is its mapped name. This designates the corresponding type definition described in 5.8. Example: in CIM, “ComFunction” “technology” attribute has “ComTechnologyKind” enumeration for type. In contextual model, “technology” **simple property** will have “ComTechnologyKind” **enumeration** type. Thus, the mapped `type-name` will be “`ComTechnologyKind`”.
- If the referent is a **codelist class** then the `type-name` is its mapped name. This designates the corresponding type definition described in 5.9.

5.11 Compound properties mapping

Each **compound property** in the contextual model has for referent a **compound class** and is mapped (as a local element definition) to a `compound-elem` as follows:

```
<xs:element
      name = prop-name
      type = compound-name
      minOccurs = min
      sawsdl:modelReference = prop-ref>
      Content: annotation
</xs:element>
```

Example of a **compound property** from Figure 1 is the element "status" under "EndDeviceEvent".

The `prop-name` is the mapped name of the property.

The `compound-name` is the mapped name of the referent **compound class**.

The value `min` is 0 if the property is optional in the contextual model or 1 if mandatory. (In the latter case the `minOccurs` attribute can be omitted, according to XML schema recommendation.)

The `prop-ref` is the URIRef designating the property definition (CIM property) of which this contextual model property is a subset.

5.12 Object properties

5.12.1 Mapping rules overview

Each **object property** in the contextual model is mapped (as a local element definition) to a `typed-elem`, a `ref-elem`, or a `union-elem`.

5.12.2 Typed object properties mapping

If the referent of the object of the property is a **structured class** then the `typed-elem` form applies:

```
<xs:element
      name = prop-name
      type = class-name
      minOccurs = min
      maxOcurs = max
      sawsdl:modelReference = prop-ref>
      Content: annotation
</xs:element>
```

Example of a **typed object property** in Figure 1 is the element "Names" under "EndDeviceEvent". It is easy to recognise, because it results in nesting, i.e., further definition of sub-elements: "name" and "NameType".

The `prop-name` is the mapped name of the property (in the example: "Names"). The `class-name` is the mapped name of the referent class (in the example: "Name").

The value `min` is the minimum cardinality of the property as defined in the contextual model. The value `max` is the maximum cardinality of the property as defined in the contextual model. (If `max` or `min` equals 1, the corresponding `maxOccurs` or `minOccurs` attribute can be omitted, according to XML schema specification.)

The `prop-ref` is the URIRef designating the property definition (CIM property) of which this contextual model property is a subset.

5.12.3 By reference object properties mapping

If the object property is defined as "**by-reference**", then it is mapped to a `ref-elem` as follows:

```
<xs:element
      name = prop-name
      minOccurs = min
      maxOccurs = max
      sawsdl:modelReference = prop-ref>
      Content: (annotation, reference)
</xs:element>
```

Example for a **by reference object property** in Figure 1 is the element "EndDeviceEventType" under "EndDeviceEvent". It is easy to distinguish from a **typed object property**, because it results in an element with just a `ref` xml attribute and without further definition of sub-elements.

The `prop-name`, `min`, `max`, and `prop-ref` are as defined above.

The `reference` is defined as follows:

```
<xs:complexType
      sawsdl:modelReference = class-ref>
      <xs:attribute
            name = ref
            type = xs:string
            use: required>
            content: (annotation?)
      </xs:attribute>
      <xs:attribute
            name = referenceType
            type = xs:string
            default = mRID
            use = optional/>
</xs:complexType>
```

The `class-ref` is a URIRef designating the class (CIM class) of which the contextual model referent is a subset.

In an XML instance, the `ref` attribute defined here takes a string representing the identifier used to identify the property's referent: it could be the “mRID” or the “Names.name” of the instance referent class.

The referent may or may not be represented elsewhere in the instance. When the referent is not represented in the instance, and thus is external to the instance, it may be defined in the contextual model as an abstract class with no concrete sub classes.

The `referenceType` attribute could be used to define which kind of identifier is used as the content of the `ref` attribute. For example, if “mRID” property is used as an identifier, the value of `referenceType` will be “mRID”.

5.12.4 Union object properties mapping

If the **object property** is marked as a **union** and its referent is a **super class**, or if the referent of the object of the property is an abstract **super class** marked as a **union** `(see annex C for examples)`, then the `union-elem` form applies:

```
<xs:choice
      minOccurs = min
      maxOcurs = max
      sawsdl:modelReference = prop-ref>
      Content: (annotation, ((branch-elem*)|(branch-ref*)))
</xs:choice>
```

The value `min` is the minimum cardinality of the property as defined in the contextual model. The value `max` is the maximum cardinality taking into account the cardinality defined in the CIM and any restrictions defined in the contextual model. (If `max` or `min` equals 1, the corresponding `maxOccurs` or `minOccurs` attribute can be omitted, according to XML schema specification.)

The `prop-ref` is the URIRef designating the property definition (CIM property) of which this contextual model property is a subset.

The content is a choice of element definitions representing properties whose referent are the **subclasses** of the referent **union superclass**. If the **union property** is **by-reference**, the `branch-ref` form applies; otherwise, the `branch-elem` form applies.

The `branch-elem` form is as follows:

```
<xs:element
      name = subproperty-name
      type = subclass-name
      sawsdl:modelReference = prop-ref>
      Content: annotation
</xs:element>
```

The `subproperty-name` will be the result of the changing name rule (see 5.17 and see examples in Annex C) and could be one of the following:

- The subclass name,
- The subclass name prefixed by the same qualifier than the one used in the object property name itself, qualifier followed by an underscore,
- The subclass name prefixed by the object property name followed by an underscore.
- The `subclass-name` is the mapped name of the **subclass**.

The `prop-ref` is designating the property definition (CIM property) of which this **union property** is a subset.

The `branch-ref` form is:

```
<xs:element
      name = subproperty-name
      sawsdl:modelReference = prop-ref>
      Content: (annotation, reference)

</xs:element>
```

The `subproperty-name` and `prop-ref` are as defined above. The `reference` is defined in the previous clause.

Either by-Ref or renaming may be used to support the mapping; however, we recommend that each organization decides which one will be used before implementation.

5.13 Exclusive property group mapping

Each **exclusive property** is mapped, to a local element definition, `exclusive-elem`, as follows:

```
<xs:choice>
      Content: (annotation, ((xor-elem*)|(xor-ref*)))
</xs:choice>
```

The content is a choice of element definitions representing the different properties part of the choice. If the property is **by-reference**, the `xor-ref` form applies; otherwise, the `xor-elem` form applies.

The `xor-elem` form is:

```
<xs:element
      name = prop-name
      type = class-name
      minOccurs = min
      maxOccurs = max
      sawsdl:modelReference = prop-ref>
      Content: annotation
</xs:element>
```

The `prop-name` is the mapped name of the contextual model property. The `class-name` is the mapped name of the referent of the property.

The value `min` is the minimum cardinality of the property as defined in the contextual model. The value `max` is the maximum cardinality taking into account the cardinality in defined in the CIM and any restrictions defined in the contextual model. (If `max` or `min` equals 1, the corresponding `maxOccurs` or `minOccurs` attribute can be omitted, according to XML schema specification).

The `prop-ref` is the URIRef designating the property definition (CIM property) of which this contextual model property is a subset.

The `xor-ref` form is:

```
<xs:element
      name = prop-name
      minOccurs = min
      maxOccurs = max
      sawsdl:modelReference = prop-ref>
      Content: (annotation, reference)
</xs:element>
```

The `prop-name`, `min`, `max` and `prop-ref` are as defined as above. The `reference` is as defined in 5.12.3 “By Reference Object Properties”.

NOTE The use of this feature is left to the decision of organizations that create the profile.

5.14 Documentation and categorized documentation

5.14.1 General mapping

Each **Documentation** or **Categorized Documentation** item attached to a contextual model element or its related CIM element is mapped to an `annotation` element on the corresponding XML schema definition:

```
<xs:annotation>
      Content: (document-elem | catdocument-elem)*
</xs:annotation>
```

5.14.2 Documentation mapping

Each **documentation** item is mapped to a `document-elem` as follows:

```
<xs:documentation>
      Content: paragraph
</xs:documentation>
```

`paragraph` specifies a paragraph of text of the prose definition of the contextual model element.

It is recommended that the documentation from the CIM UML be split into paragraphs and inserted first. Documentation from the contextual model, if any, may follow.

5.14.3 Categorized documentation mapping

Each **categorizedDocumentation** item is mapped to a `catdocument-elem` as follows:

```
<xs:documentation
      xml:lang = language
      p:category: xs:string
      p:subCategory: xs:string
      p:notes: xs:string>
      content: paragraph
</xs:documentation>
```

"`lang`" attribute specifies in which human language the documentation content is written.

"`category`" and "`subCategory`" attributes are used to express the documentation classification.

"`notes`" attribute mapped the notes attached to the documentation classification.

5.14.4 Stereotype mapping

Each **stereotype** will map to a `catdocument-elem` as follows:

```
<xs:documentation
      p:category: stereotype
      p:notes: xs:string
      content: stereotype-name
</xs:documentation>
```

The category is "*stereotype*".

The `stereotype-name` is the name of the contextual model **stereotype**.

5.15 Names

The names of classes, properties and values appearing in the contextual model are mapped to XML Schema such that:

- All mapped names except enumeration values must conform with *Namespaces in XML 1.0* NCName syntax.
- Enumeration values must be mapped to values that conform with *XML1.0* attribute values syntax.

The mapped name is obtained by transliterating the contextual model name, character by character.

5.16 Mapping order

5.16.1 General mapping order basis

There shall be a mapping order:

- Either an alphabetical order, when the mapping order is not defined in the contextual model.

  EXAMPLE:

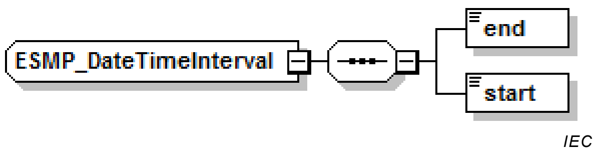

Figure 2 – Example of alphabetical order

- Or a business context order (e.g. to have "start" attribute before "end" attribute in a time interval). In such a case, this business context order shall be defined prior to the mapping (for example, this is done at contextual model level in the 62325-451 series). In that case this order is used for the mapping. This standard does not specify how this order is defined.

  EXAMPLE:

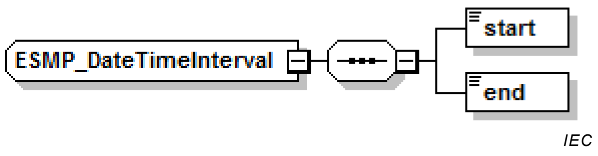

Figure 3 – Example of business order

5.16.2 Alphabetical based mapping order

5.16.2.1 General mapping order

All the mapping is done so that all elements appear in the schema in a defined order:

- `Root` elements are ordered according to an alphabetical order;
- `Elements` representing class's properties are ordered according to the following subclauses depending if there are or not super classes in the contextual model.

5.16.2.2 Mapping order when there is no super class

- When there is no super class, elements representing properties are ordered as follows:
  - mRID simple property first,
  - then other simple properties in alphabetical order,
  - then object and compound properties in alphabetical order.

NOTE **Union object properties** are mapped to an `xs:choice` element. This choice element place in the alphabetical ordered object and compound properties list is given by the name of the **union object property**. Example: in contextual model, “MktOrganization” could have a “RegisteredResource” **union object property** that will be mapped to an `xs:choice`. So the `xs:choice` place in the order is the one that “RegisteredResource” property would have had if “RegisteredResource” have been map to an element instead to an `xs:choice`.

5.16.2.3 Mapping order when there is a super class

When a super class exist, elements representing super class properties are ordered as the previous subclause. The element representing subclasses properties are ordered as follows:

- First elements representing super class properties in the alphabetical order as defined in 5.16.2.2,
- Then elements representing native properties of the subclass in the same order as defined in 5.16.2.2.

5.16.2.4 Mapping order examples

In CIM, “MktOrganisation” is inheriting from “Organisation” that inherits from “IdentifiedObject”. In contextual model, we could have two cases:

- “MktOrganisation” **structured class** stands alone and could have, as native **simple properties**, properties whose CIM counterparts (attributes) are in “IdentifiedObject”, for example “mRID” and “name”.
- “MktOrganisation” **structured class** inherits, for example, from an “IdentifiedObject” **super structured class** that has “mRID” and “name” **simple properties**.

If contextual model “MktOrganisation” stands alone and has “mRID”, “name”, “creditFlag” and “lastModified” **simple properties**, the mapping order will be as follows:

- mRID
- creditFlag
- lastModified
- name

If contextual model “MktOrganisation” inherits from “IdentifiedObject”. “IdentifiedObject” has “mRID” and “name” **simple properties** and “MktOrganisation” has “creditFlag” and “lastModified” **simple properties**. The mapping order will be as follows:

- mRID
- name
- creditFlag

- lastModified

5.16.3 Business context based mapping order

When a business context defined a specific order for the properties, this order is used to do the mapping order.

Figure 4 provides an example:

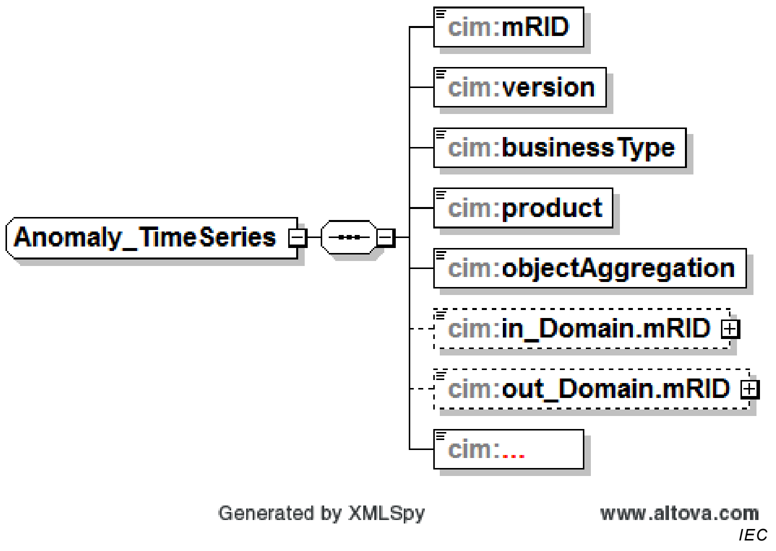

Figure 4 – Example business context order of a schema

5.17 Changing name rules

The rule to change contextual model **object property** names to schema mapped names could be used to avoid duplication of element name in a sequence or a choice schema design. This could occur when the contextual model **object property**:

- is marked as a “**union**” and its referent is a contextual model **super class**,
- or is a subset of a CIM association with a CIM super class that is used in the contextual model with a referent that is a subset of subclasses of this CIM super class.

So, in both cases, the contextual model **object property** name matches a CIM association end role name whose referent is a CIM super class.

Contextual model **object property** name could be of three kinds:

1) same as the one of the **superclass** name,
2) same as the **superclass** name prefixed by a qualifier with or without an underscore,
3) or not the same as the **superclass** name.

The changing name rule would be as follows for a **union object property**:

- If the names are the same, the mapped name will be the contextual model **subclass** name,
- If the names differ just with a qualifier, the mapped name will be the contextual model **subclass** name prefixed by the same qualifier and the underscore,
- If the names are different, the mapped name will be the contextual model **subclass** name prefixed by the **object property** name and an underscore.

The changing name rule would be as follows in the second case:

- If the object property name matches the CIM super class name, the mapped name will be the contextual model **subclass** name,
- If the object property name matches the CIM super class name plus a qualifier (and an underscore), the mapped name will be the contextual model **subclass** name prefixed by the same qualifier and the underscore,
- If the names are different, the mapped name will be the contextual model **subclass** name prefixed by the **object property** name and an underscore.

Annex A
(normative)

Use of dedicated XML schemas for datatypes, enumerations and codelists

A.1 Context:

Dedicated schemas (for datatypes, enumerations and or codelists) could be specified in order to:

- Have modular schemas;
- Allow flexibility in the maintenance and use of enumerations and codelists. This is particularly useful when the enumerations or codelists are maintained by other organisations.

Those dedicated schemas could be specified by a mapping from the contextual model or specified outside and then imported or included in the main schema specified by this standard.

There are two ways to use dedicated schemas for datatypes, and/or enumerations and/or codelists:

- Using one namespace for all the elements: this is done through dedicated schemas inclusion (`xs:include`);
- Using different namespaces for the elements: this is done through dedicated schemas import (`xs:import`).

A.2 Modular schema design and mapping:

When modular schema design is used, the single profile is mapped to a single XML Schema with the following form:

```
<xs:schema
      (xmlns:prefix = namespace-uri)*
      targetNamespace = namespace-uri
      elementFormDefault = qualified
      attributeFormDefault = unqualified
      version = version-string >
      content: ((include | import)*, annotation, ((envelope-elem, envelope-
type) | (root-elem, root-type)), (complex-type | simple-type)*)
</xs:schema>
```

The "`xmlns`" attributes specify all the namespaces described in 5.2.1 plus the namespaces of the imported schemas. `prefix` is the namespace prefix and `namespace-uri` is the namespace URI.

Each `import` is defined as follows:

```
<xs:import
      namespace = namespace-uri
      schemaLocation = any-uri
      content: annotation
</xs:import>
```

The `namespace-uri` is the URI of the dedicated imported schema namespace.

Each `include` is defined as follows:

```
<xs:include
      schemaLocation = location-uri
      content: annotation
</xs:include>
```

The `location-uri` is the URI where is located the dedicated included schema namespace.

When defining dedicated schemas for datatypes, enumerations and/or codelists, the general mapping is defined as follows:

```
<xs:schema
      (xmlns:prefix = namespace-uri)*
      targetNamespace = namespace-uri
      elementFormDefault = qualified
      attributeFormDefault = unqualified
      version = version-string >
      content:(include | import)*, annotation, (complex-type | simple-
type)*
</xs:schema>
```

The meaning of the attributes is the same as above.

Dedicated schema could itself import or include other dedicated schemas.

A.3 Artefact mapping

A.3.1 General rule

The main change in the mapping is that the `type-name` of an element type is a "Qname" and not an "NCName" if the referring type is in a dedicated imported schema.

```
<xs:element
      name = prop-name
      type = type-name ➔ is a QName
      minOccurs = min
      sawsdl:modelReference = prop-ref>
      Content: annotation
</xs:element>
```

EXAMPLE: In CIM, “EndDeviceInfo” “ratedVoltage” attribute has CIMDatatype “Voltage” for type. In contextual model, “ratedVoltage” **simple property** will have “Voltage” **data type** type. Thus, if there is a dedicated datatype schema that have a namespace whose prefix is "dt", and if this dedicated schema is imported, then the `type-name` will be “`dt:Voltage`” (QName) and not "`Voltage`" (NCName).

A.3.2 Datatype, enumeration or codelist mapping:

If datatypes, enumerations or codelists schemas are part of an imported or included schema in the main schema, then there are no mapping of datatypes, enumerations or codelists in the main schema: Subclauses 5.7, 5.8 and 5.9 are not used for the main schema. The corresponding complexTypes or simpleTypes are part of the dedicated schemas.

A.3.3 Simple property mapping

Each **simple property** is mapped (as a local element definition) to a `simple-elem` as follows (see 5.10):

```
<xs:element
      name = prop-name
      type = type-name
      minOccurs = min
      sawsdl:modelReference = prop-ref>
      Content: annotation
</xs:element>
```

If the type-name of the element.type is referring to a type (complexType or simpleType) of an imported dedicated schema, then the type-name must be a QName as follows:

- `prefix`: the prefix of the imported schema namespace
- `colon` (":")
- `mapped name`

For example, if the imported schemas have the following namespaces:

- For datatypes: xmlns:dt="http://datatypes"
- For enumerations: xmlns:enum="http://enumerations"
- For codeLists: xmlns:cl="http://codelists"

Then

- If the referent is a **simple type** then the `type-name` is its mapped name prefixed by the prefix of the simple type schema namespace followed by a colon. Example: if in contextual model, "issuerID" type was a simple type name “Char24_String” (for example a String of 24 characters), the mapped `type-name` will be “`dt:Char24_String`”.
- If the referent is a **data type** then the `type-name` is its mapped name prefixed by the prefix of the simple type schema namespace followed by a colon. Example: in CIM, “EndDeviceInfo” “ratedVoltage” attribute has CIMDatatype “Voltage” for type. In contextual model, “ratedVoltage” **simple property** will have “Voltage” **data type** type. Thus, the mapped `type-name` will be “`dt:Voltage`”.
- If the referent is an **enumeration class** then the `type-name` is its mapped name prefixed by the prefix of the simple type schema namespace followed by a colon. This designates the corresponding type definition described in 5.8. Example: in CIM, “ComFunction” “technology” attribute has “ComTechnologyKind” enumeration for type. In contextual model, “technology” **simple property** will have “ComTechnologyKind” **enumeration** type. Thus, the mapped `type-name` will be “`en:ComTechnologyKind`”.
- If the referent is a **codelist class** then the `type-name` is its mapped name prefixed by the prefix of the simple type schema namespace followed by a colon. This designates the corresponding type definition described in 5.9.

Annex B
(informative)

Contextual model representations

The mapping definitions apply to a contextual model which must exist in some form before an XML Schema can be constructed. The representation of a contextual model and the rules for constructing it are outside the scope of this specification and the following information is not normative.

Two contextual model representations are presently in use: *Web Ontology Language* (OWL) and *Unified Modeling Language* (UML). The contextual model concepts used in this specification could be represented in UML and OWL as defined in Table B.1.

Table B.1 – Contextual model representation

Concept	UML Definition	OWL Definition
Structured class	Class	owl:Class
Root class	Class with “Root” qualifier	owl:Class
Union class	Class with a stereotype to be defined (“Union” for example)	owl:Class with owl:unionOf
Compound class	Class with “Compound” stereotype	owl:Class with compound annotation
Object property	Association end	owl:ObjectProperty with zero or more owl:Restriction
By-reference object property	Association end whose end type is designated by its reference	owl:ObjectProperty with zero or more owl:Restriction and byReference annotation
Union property	Association end with a stereotype to be defined (“Union” for example)	
Compound property	Class attribute whose type is a compound class	
Exclusive property group	Associations with an XOR constraint	
Simple property	Class attribute	owl:DatatypeProperty with zero or more owl:Restriction
BasicType	Datatype with “Primitive” stereotype	XML schema datatype
Simple type	Datatype with “CIMDatatype” stereotype that has only a value attribute and constraints expressing restrictions on the type of the value attribute	rdfs:Datatype
Datatype	Datatype with “CIMDatatype” stereotype and optional constraints	rdfs:Datatype with quantity annotation
Enumeration class	Class with “enumeration” stereotype	owl:Class with owl:oneOf
CodeList	Class with a stereotype to be defined (“CodeList” for example)	owl:Class
Subclass/Superclass relationship	Generalized By (inherits)	owl:SubClassOf
Documentation	Notes	rdfs:comment
Categorized documentation	TaggedValue, stereotype, constraint	rdfs:comment

Annex C
(informative)

Changing name rules examples

C.1 Changing name rule context

The rule to change contextual model **object property** names to schema mapped names could be used to avoid duplication of element name in a sequence or a choice schema design. This could occur:

- when the referent of a contextual model **object property** is a contextual model **super class** marked as a "**union**",
- or when a contextual model class has several **object properties** that have the same name. This occurs when these **object properties** are subsets of the same CIM association with a CIM super class and are used in the contextual model with referents that are subset of subclasses of this CIM super class.

So, in both cases, the contextual model **object property** name matches a CIM association end role name whose referent is a CIM super class.

The examples are using UML and XML to illustrate the problem and the changing name rules.

C.2 Changing name rules when using "union" super class

C.2.1 General

The different cases are described in C.2.2 to C.2.4.

C.2.2 Changing name rule when CIM association end role name and CIM super class name are the same

Changing name rule could occur when in the information model there is a class that has an association with a super class and the association end role name is the same than the super class name. In Figure C.1, the association end role name is "SuperClass" and SuperClass name is also "SuperClass":

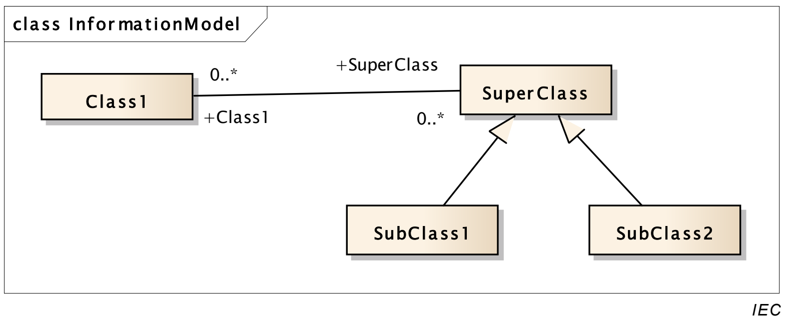

Figure C.1 – Example of end role name matching super class name

In this case, at contextual model level, the **superclass** could be used and marked as a "**Union**" (meaning that subclasses are selected), see Figure C.2:

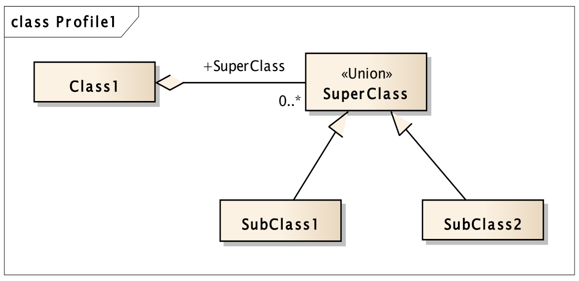

Figure C.2 – Contextual model end role name matching super class name

When mapping to XSD according to IEC 62361-100, the result will be:

```
<xs:element name="Class">
    <xs:complexType>
      <xs:choice minOccurs="0">
              <xs:element name="SuperClass" type="SubClass1"/>
              <xs:element name="SuperClass" type="SubClass2"/>
      </xs:choice>
    </xs:complexType>
</xs:element>
```

As, in a choice, XML elements could not have the same name but different types, we have to apply the changing name rule, which in this case says that the element name will be the **subclass** name:

```
<xs:element name="Class">
   <xs:complexType>
      <xs:choice minOccurs="0">
              <xs:element name="SubClass1" type="SubClass1"/>
              <xs:element name="SubClass2" type="SubClass2"/>
      </xs:choice>
   </xs:complexType>
</xs:element>
```

C.2.3 Changing name rule when CIM association end role name is the CIM super class name prefixed by a qualifier followed by an underscore

Changing name rule could occur when in the information model there is a class that has an association with a super class and the association end role name is the same than the super class name prefixed by a qualifier followed by an underscore. In Figure C.3, SuperClass name is "SuperClass" and association end role name is "Qualifier_SuperClass":

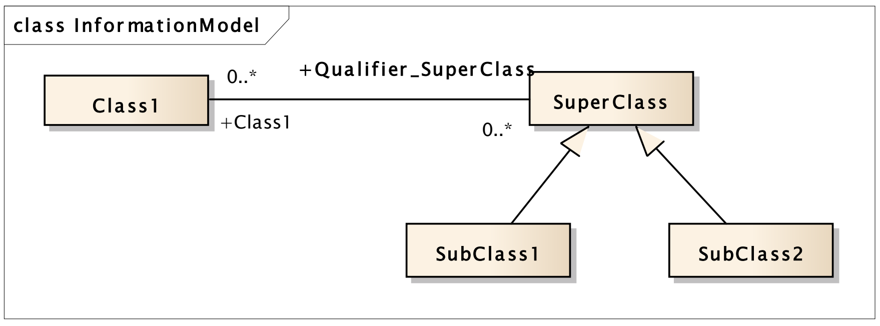

Figure C.3 – Example of end role name with a qualifier

In this case, at contextual model level, the **superclass** could be used and marked as a **Union** (meaning that **subclasses** are selected), see Figure C.4:

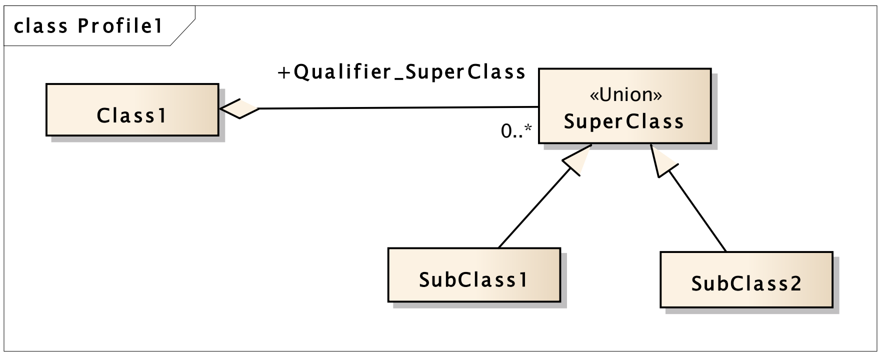

Figure C.4 – Contextual model end role name with a qualifier

When mapping to XSD according to IEC 62361-100, the result will be:

```
<xs:element name="Class">
    <xs:complexType>
      <xs:choice minOccurs="0">
              <xs:element name="Qualifier_SuperClass" type="SubClass1"/>
              <xs:element name="Qualifier_SuperClass" type="SubClass2"/>
      </xs:choice>
    </xs:complexType>
</xs:element>
```

As, in a choice, XML elements could not have the same name but different types, we have to apply the changing name rule, which in this case says that the element name will be the **subclass** name prefixed by the qualifier followed by an underscore:

```
<xs:element name="Class">
   <xs:complexType>
      <xs:choice minOccurs="0">
              <xs:element name="Qualifier_SubClass1" type="SubClass1"/>
              <xs:element name="Qualifier_SubClass2" type="SubClass2"/>
      </xs:choice>
   </xs:complexType>
</xs:element>
```

C.2.4 Changing name rule when CIM association end role name and the CIM super class name are completely different

Changing name rule could occur when in the information model there is a class that has an association with a super class and the association end role name is different than the super class name. In Figure C.5, SuperClass name is "SuperClass" and association end role name is "EndName":

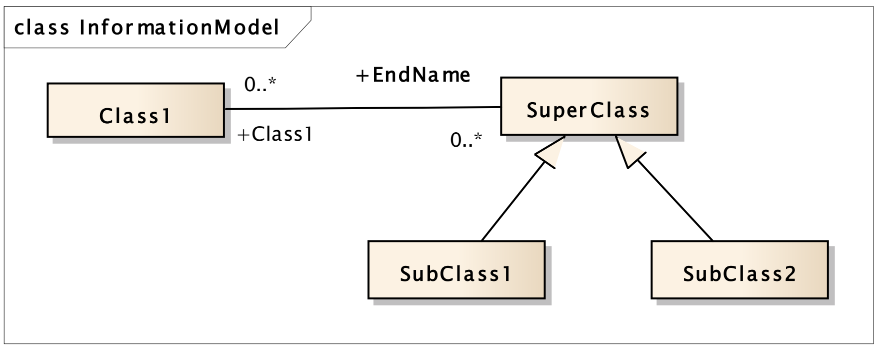

Figure C.5 – End role name and super class name different

In this case, at contextual model level, the **superclass** could be used and marked as a **Union** (meaning that subclasses are selected), see Figure C.6:

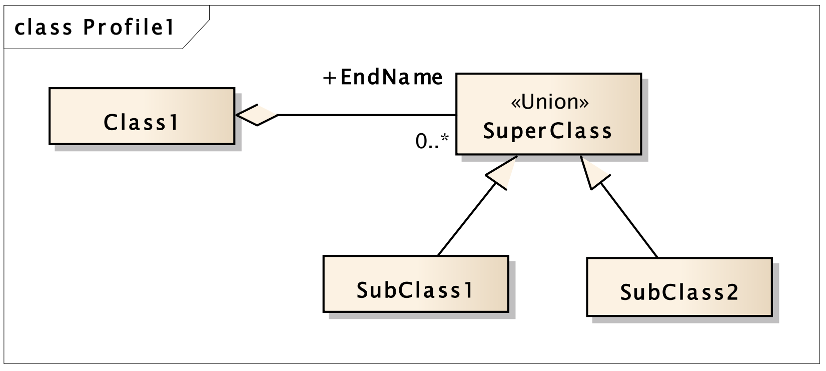

Figure C.6 – Contextual model with end role name different from super class name

```
<xs:element name="Class">
    <xs:complexType>
       <xs:choice minOccurs="0">
              <xs:element name="EndName" type="SubClass1"/>
              <xs:element name="EndName" type="SubClass2"/>
       </xs:choice>
    </xs:complexType>
</xs:element>
```

As, in a choice, XML elements could not have the same name but different types, we have to apply the changing name rule, which in this case says that the element name will be the **subclass** name prefixed by the association end role name followed by an underscore:

```
<xs:element name="Class">
    <xs:complexType>
      <xs:choice minOccurs="0">
              <xs:element name="EndName_SubClass1" type="SubClass1"/>
              <xs:element name="EndName_SubClass2" type="SubClass2"/>
      </xs:choice>
    </xs:complexType>
</xs:element>
```

C.3 Changing name rules when using complex properties with the same name

C.3.1 General

The different cases are described in the C.3.2 to C.3.4.

C.3.2 Changing name rule when CIM association end role name and CIM super class name are the same

Changing name rule could occur when in the information model there is a class that has an association with a super class and the association end role name is the same than the super class name. In Figure C.7, the association end role name is "SuperClass" and SuperClass name is also "SuperClass":

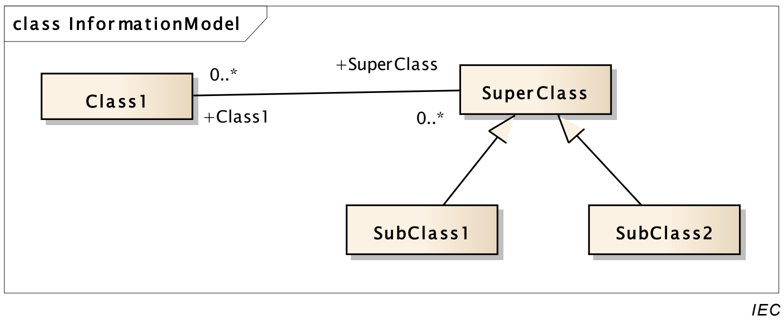

Figure C.7 – Example of end role name matching super class name

In this case, at contextual model level, if the inheritance is not used, contextual model class could be associated with the classes that are subsets of subclasses of the CIM super class, see Figure C.8:

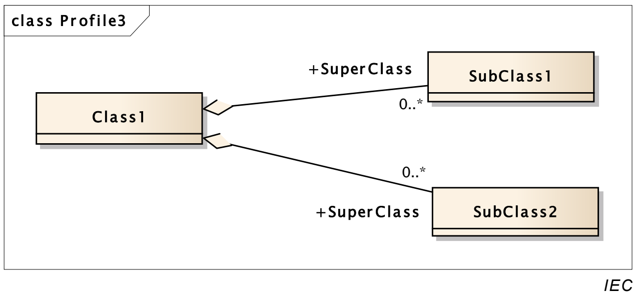

Figure C.8 – Contextual model end role name matching super class name

When mapping to XSD according to IEC 62361-100, the result will be:

```
<xs:element name="Class">
    <xs:complexType>
      <xs:sequence>
            <xs:element name="SuperClass" type="SubClass1" minOccurs="0"/>
            <xs:element name="SuperClass" type="SubClass2" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
```

As, in a sequence, XML elements could not have the same name but different types, we have to apply the changing name rule, which in this case says that the element name will be the **subclass** name:

```
<xs:element name="Class">
   <xs:complexType>
      <xs:sequence>
            <xs:element name="SubClass1" type="SubClass1" minOccurs="0"/>
            <xs:element name="SubClass2" type="SubClass2" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

C.3.3 Changing name rule when CIM association end role name is the CIM super class name prefixed by a qualifier followed by an underscore

Changing name rule could occur when in the information model there is a class that has an association with a super class and the association end role name is the same than the super class name prefixed by a qualifier followed by an underscore. In Figure C.9, SuperClass name is "SuperClass" and association end role name is "Qualifier_SuperClass":

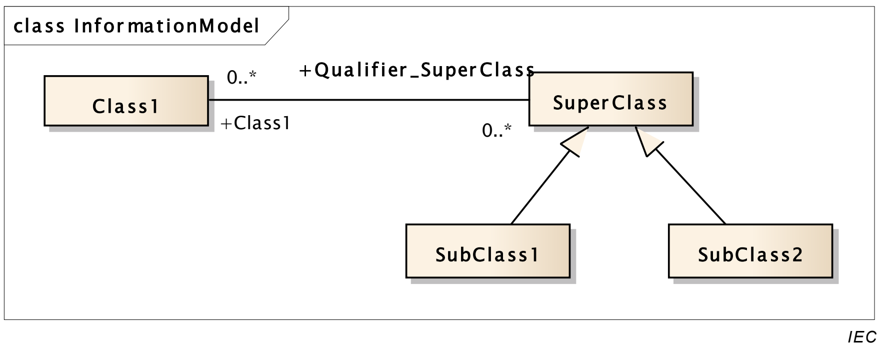

Figure C.9 – Example of end role name with a qualifier

In this case, at contextual model level, if the inheritance is not used, contextual model class could be associated with the classes that are subsets of subclasses of the CIM super class, see Figure C.10:

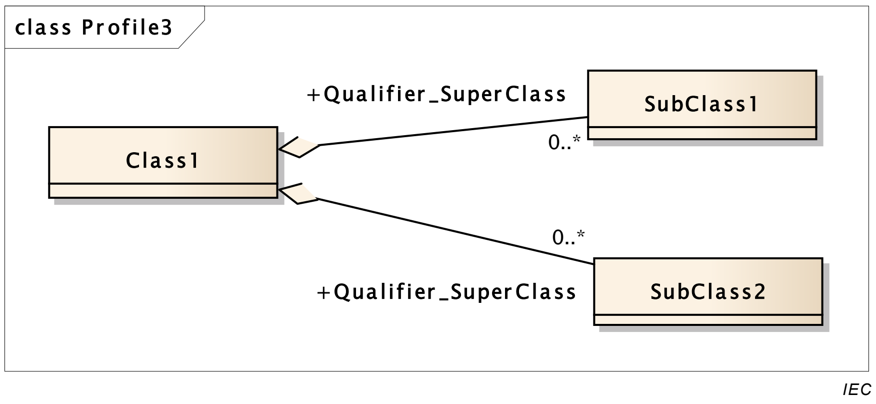

Figure C.10 – Contextual model end role name with a qualifier

When mapping to XSD according to IEC 62361-100, the result will be:

```
<xs:element name="Class">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Qualifier_SuperClass"  type="SubClass1" minOccurs="0"/>
   <xs:element name="Qualifier_SuperClass"  type="SubClass2" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

As, in a sequence, XML elements could not have the same name but different types, we have to apply the changing name rule, which in this case says that the element name will be the **subclass** name prefixed by the qualifier followed by an underscore:

```
<xs:element name="Class">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Qualifier_SubClass1"  type="SubClass1" minOccurs="0"/>
   <xs:element name="Qualifier_SubClass2"  type="SubClass2" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

C.3.4 Changing name rule when CIM association end role name and the CIM super class name are completely different

Changing name rule could occur when in the information model there is a class that has an association with a super class and the association end role name is different than the super class name. In Figure C.11, SuperClass name is "SuperClass" and association end role name is "EndName":

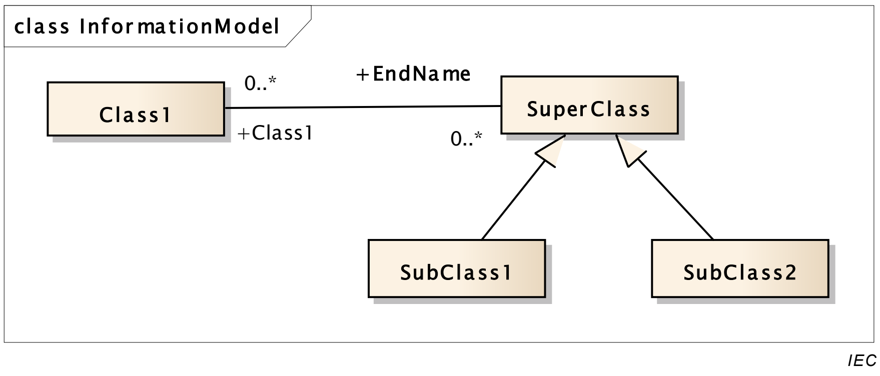

Figure C.11 – End role name and super class name different

In this case, at contextual model level, if the inheritance is not used, contextual model class could be associated with the classes that are subsets of subclasses of the CIM super class, see Figure C.12:

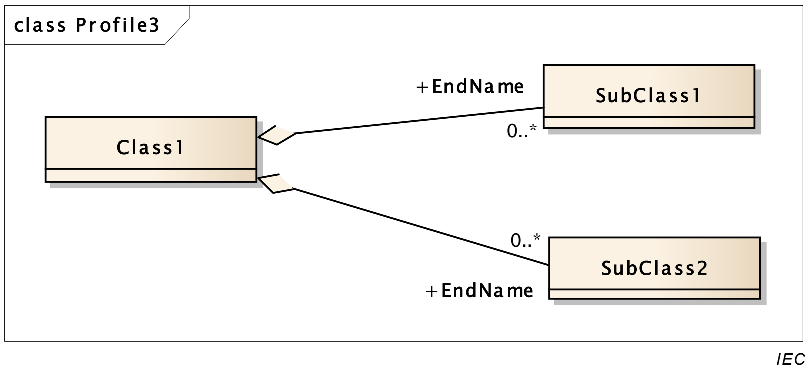

Figure C.12 – Contextual model with end role name different from super class name

```
<xs:element name="Class">
    <xs:complexType>
       <xs:sequence>
              <xs:element name="EndName" type="SubClass1" minOccurs="0"/>
              <xs:element name="EndName" type="SubClass2" minOccurs="0"/>
       </xs:sequence>
    </xs:complexType>
</xs:element>
```

As, in a sequence, XML elements could not have the same name but different types, we have to apply the changing name rule, which in this case says that the element name will be the **subclass** name prefixed by the association end role name followed by an underscore:

```
<xs:element name="Class">
   <xs:complexType>
      <xs:sequence>
       <xs:element name="EndName_SubClass1" type="SubClass1" minOccurs="0"/>
       <xs:element name="EndName_SubClass2" type="SubClass2" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
```

```
</xs:element>
```

Bibliography

IEC 60050 series, *International Electrotechnical Vocabulary*

IEC 61968 series, *Application integration at electric utilities – System interfaces for distribution management*

IEC 62325 series, *Framework for energy market communications*

CIM Users Group – The community of CIM users. http://cimug.ucaiug.org

Extensible Markup Language (XML) 1.0 (Fifth Edition) W3C Recommendation 5 February 2008

W3C: Extensible Markup Language (XML) 1.0 http://www.w3.org/XML/

Namespaces in XML 1.0 (Third Edition), W3C Recommendation 8 December 2009

XML Schema Part 2: Datatypes Second Edition W3C Recommendation 28 October 2004

OWL Web Ontology Language Reference W3C Recommendation 10 February 2004

Unified Modelling Language (UML) Specification V2.2, Object Management Group

UN/CEFACT XML Naming and Design Rules Version 3.0, 19 December 2009

UN/CEFACT Core Component Technical Specification 3.0, 29 September 2009

This page deliberately left blank

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Copyright in BSI publications

All the content in BSI publications, including British Standards, is the property of and copyrighted by BSI or some person or entity that owns copyright in the information used (such as the international standardization bodies) and has formally licensed such information to BSI for commercial publication and use.

Save for the provisions below, you may not transfer, share or disseminate any portion of the standard to any other person. You may not adapt, distribute, commercially exploit, or publicly display the standard or any portion thereof in any manner whatsoever without BSI's prior written consent.

Storing and using standards

Standards purchased in soft copy format:

- A British Standard purchased in soft copy format is licensed to a sole named user for personal or internal company use only.
- The standard may be stored on more than 1 device provided that it is accessible by the sole named user only and that only 1 copy is accessed at any one time.
- A single paper copy may be printed for personal or internal company use only.

Standards purchased in hard copy format:

- A British Standard purchased in hard copy format is for personal or internal company use only.
- It may not be further reproduced – in any format – to create an additional copy. This includes scanning of the document.

If you need more than 1 copy of the document, or if you wish to share the document on an internal network, you can save money by choosing a subscription product (see 'Subscriptions').

Reproducing extracts

For permission to reproduce content from BSI publications contact the BSI Copyright & Licensing team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email subscriptions@bsigroup.com.

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Useful Contacts

Customer Services
Tel: +44 345 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 345 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK