IEC 62361-100

Edition 1.0 2016-06

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Power systems management and associated information exchange – Interoperability in the long term – Part 100: CIM profiles to XML schema mapping

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 62361-100:2016-06(en-fr)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

IEC 62361-100

Edition 1.0 2016-06

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Power systems management and associated information exchange – Interoperability in the long term – Part 100: CIM profiles to XML schema mapping

Gestion des systèmes de puissance et échanges d'informations associés – Interopérabilité à long terme – Partie 100: Mapping des profils CIM avec le schéma XML

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 33.200

ISBN 978-2-8322-3454-9

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

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

International Standard IEC 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)

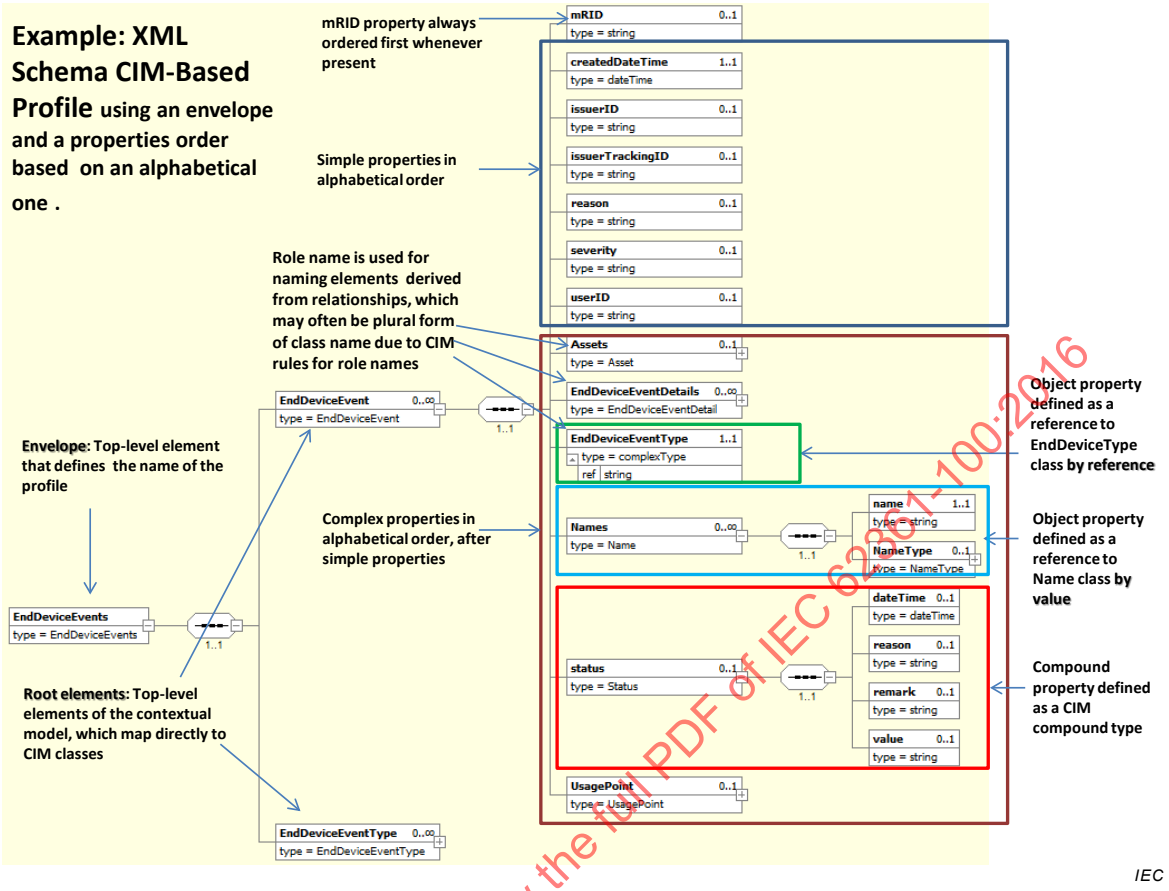


Figure 1 – Example XML Schema CIM-based profile

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:

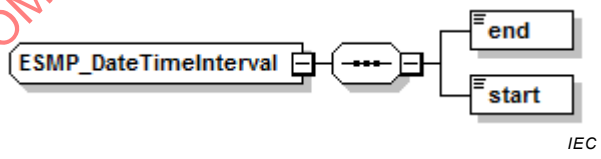


Figure 2 – Example of alphabetical order

- 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:

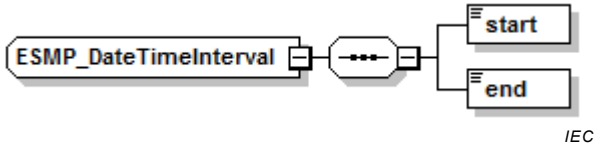


Figure 3 – Example of business order

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:

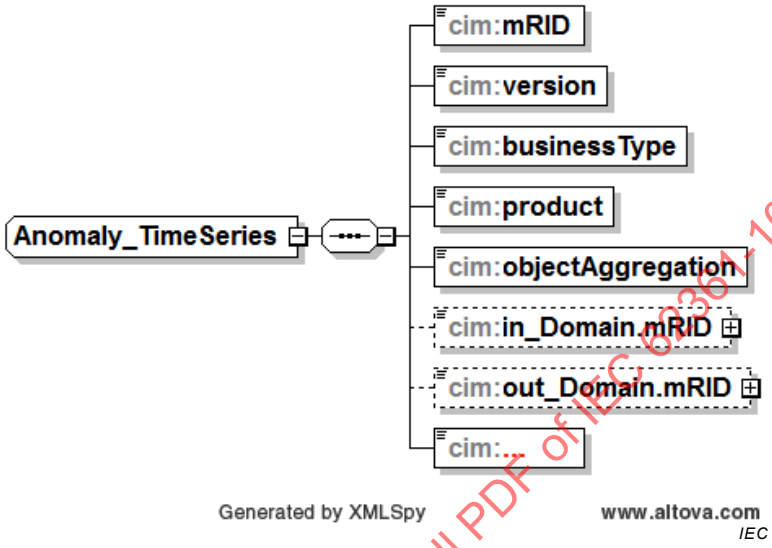


Figure 4 – Example business context order of a schema

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":

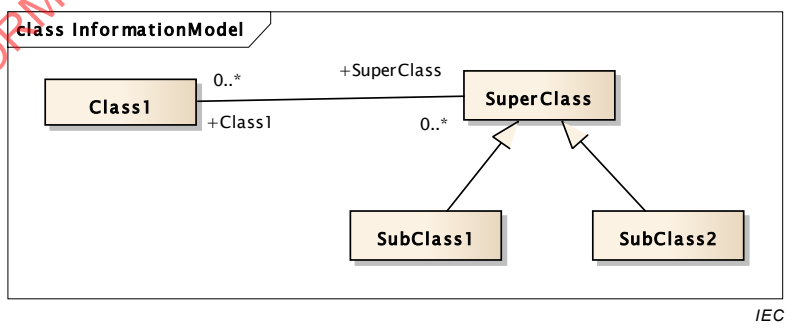


Figure C.1 – Example of end role name matching super class name

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:

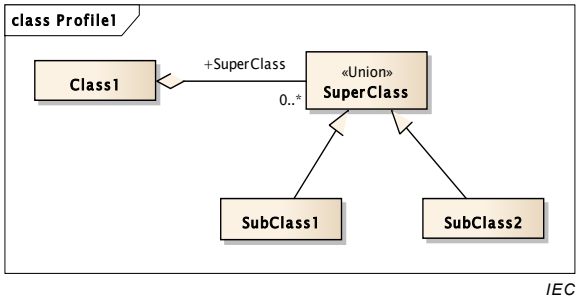


Figure C.2 – Contextual model end role name matching super class name

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":

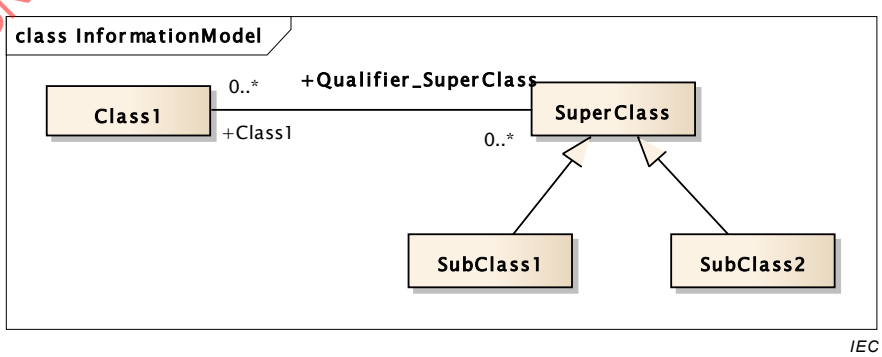


Figure C.3 – Example of end role name with a qualifier

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:

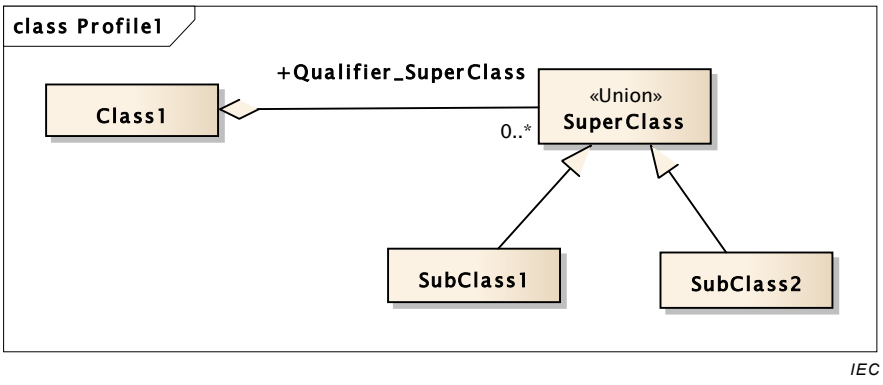


Figure C.4 – Contextual model end role name with a qualifier

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":

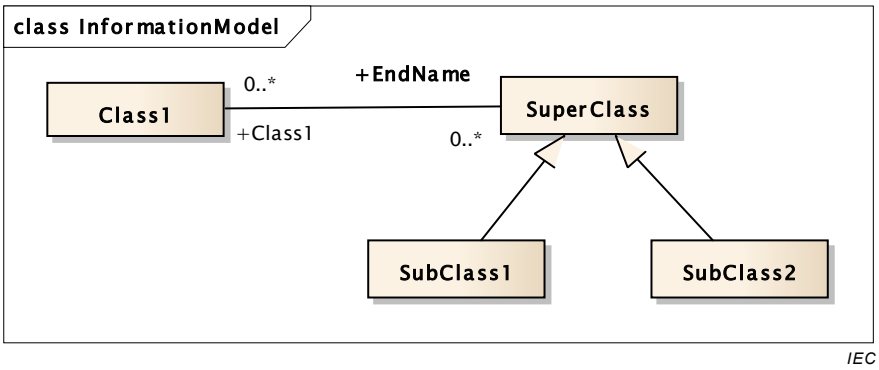


Figure C.5 – End role name and super class name different

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:

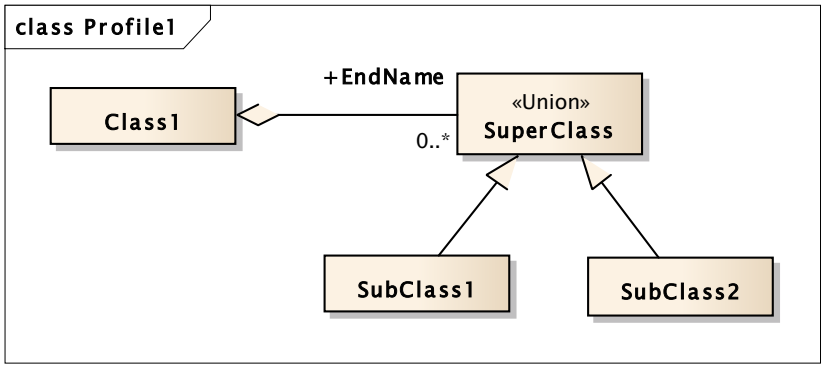


Figure C.6 – Contextual model with end role name different from super class name

```
<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":

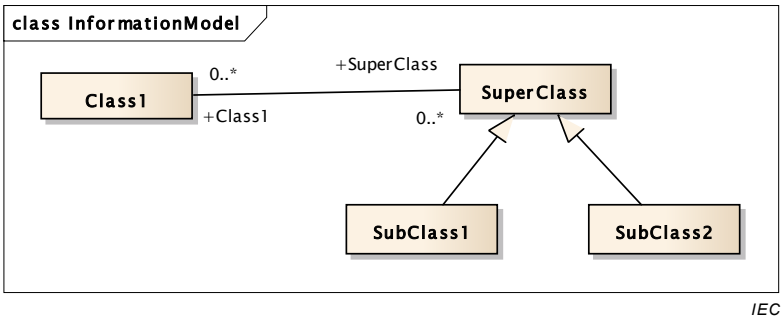


Figure C.7 – Example of end role name matching super class name

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:

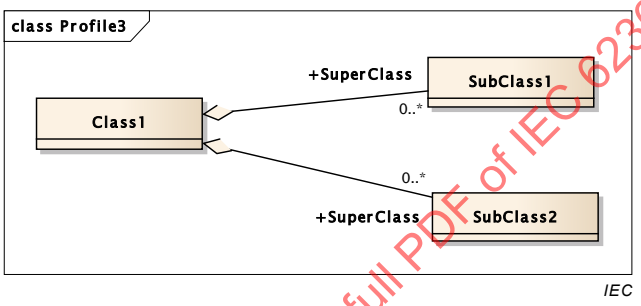


Figure C.8 – Contextual model end role name matching super class name

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":

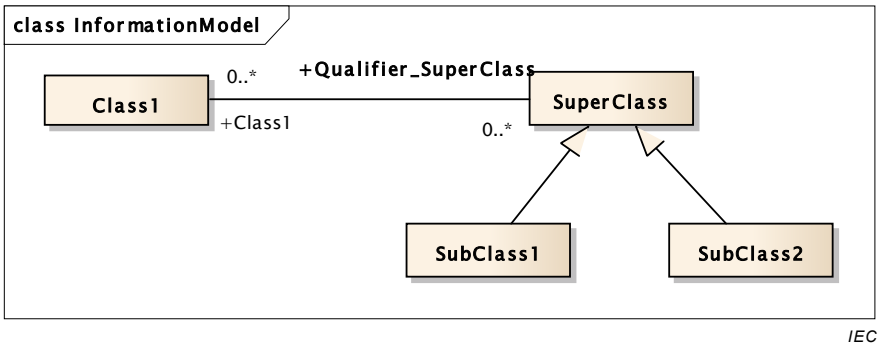


Figure C.9 – Example of end role name with a qualifier

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:

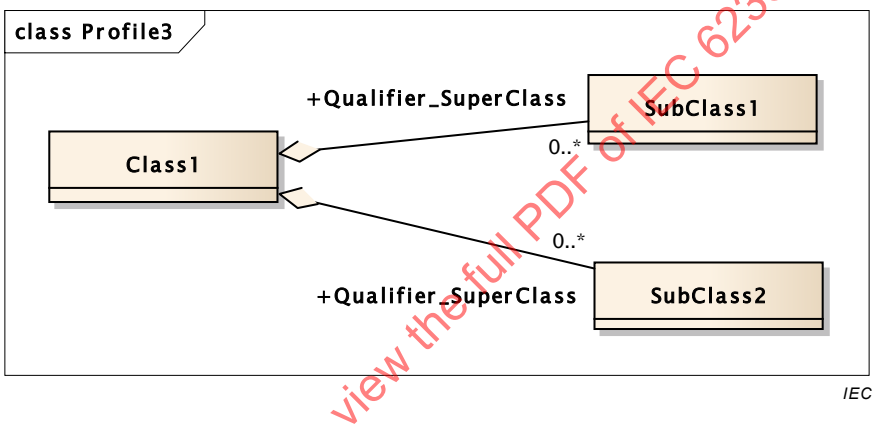


Figure C.10 – Contextual model end role name with a qualifier

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":

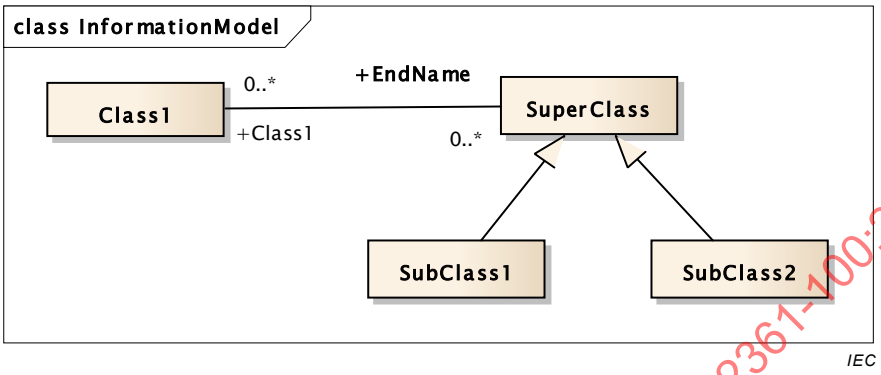


Figure C.11 – End role name and super class name different

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:

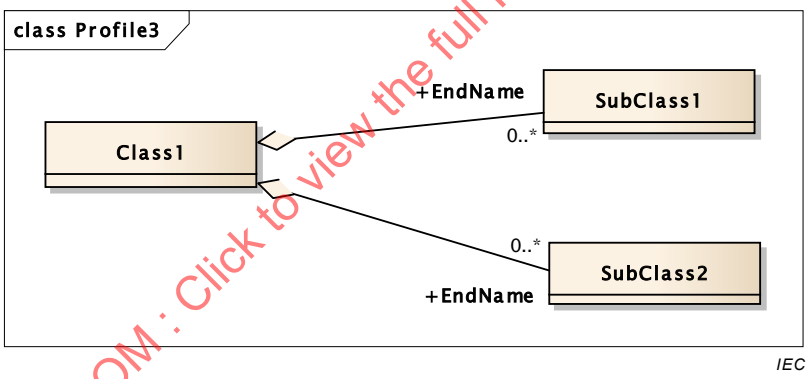


Figure C.12 – Contextual model with end role name different from super class name

```
<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

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

GESTION DES SYSTÈMES DE PUISSANCE ET ÉCHANGES D'INFORMATIONS ASSOCIÉS – INTEROPÉRABILITÉ À LONG TERME –

Partie 100: Mapping des profils CIM avec le schéma XML

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Norme internationale IEC 62361-100 a été établie par le comité d'études 57 de l'IEC: Gestion des systèmes de puissance et échanges d'informations associés.

Le présent document est la première édition de la norme.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
57/1704/FDIS	57/1735/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/IEC, Partie 2.

Dans le présent document, les caractères d'imprimerie suivants sont employés:

- les mots imprimés en **Arial Black** s'appliquent aux termes définis comme des artéfacts de modèles contextuels en 4.4.2;
- les mots imprimés en `Courier New` s'appliquent aux termes utilisés dans le cadre de la représentation du schéma XML (définie en 4.6) ou dans des exemples XML;
- les mots placés "entre guillemets" s'appliquent aux termes utilisés comme des jetons dans les articles normatifs définis comme des artéfacts du CIM.

Une liste de toutes les parties de la série IEC 62361, publiées sous le titre général: *Gestion des systèmes de puissance et échanges d'informations associés – Interopérabilité à long terme*, peut être consultée sur le site web de l'IEC.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

IMPORTANT – Le logo *"colour inside"* qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

La série IEC 62361 définit les normes, qui traitent de domaines d'intérêt affectant plusieurs normes et apportent de la cohérence aux mises en œuvre.

La présente partie de l'IEC 62361 décrit une mise en correspondance (mapping ou mappage) entre les profils CIM et les schémas XML W3C (World Wide Web Consortium), puis définit les règles auxquelles les charges utiles des messages XML CIM doivent se conformer.

La présente partie de l'IEC 62361 a pour objectif principal de faciliter l'échange d'informations sous la forme de documents XML dont la sémantique est définie par le CIM de l'IEC et dont la syntaxe est définie par un schéma XML W3C. Cela facilitera l'intégration de toutes les applications qui utilisent les charges utiles de messages du schéma XML développées par les groupes de travail et mises en œuvre de façon indépendante par les différents fournisseurs dans leurs systèmes.

Le modèle d'information commun (CIM, *Common Information Model*) spécifie la base de la sémantique des échanges de charges utiles de messages définies par l'IEC. Les spécifications de profil, qui sont contenues dans d'autres parties des normes de la série IEC 62361, spécifient le contenu des charges utiles de messages échangées. Le format/la syntaxe de ces charges utiles sont spécifiés dans la présente partie de l'IEC 62361.

GESTION DES SYSTÈMES DE PUISSANCE ET ÉCHANGES D'INFORMATIONS ASSOCIÉS – INTEROPÉRABILITÉ À LONG TERME –

Partie 100: Mapping des profils CIM avec le schéma XML

1 Domaine d'application

La présente partie de l'IEC 62361 décrit le mapping entre les profils CIM et les schémas XML W3C.

Ce mapping a pour objectif de faciliter l'échange d'informations sous la forme de documents XML dont la sémantique est définie par le CIM de l'IEC et dont la syntaxe est définie par un schéma XML W3C.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

IEC 61968-11, *Intégration d'applications pour les services électriques – Interfaces système pour la gestion de distribution – Partie 11: Extensions du modèle d'information commun (CIM) pour la distribution*

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

IEC 61970-301, *Interface de programmation d'application pour système de gestion d'énergie (EMS-API) – Partie 301: Base de modèle d'information commun (CIM)*

IEC 62325-301, *Cadre pour les communications pour le marché de l'énergie – Partie 301: Extensions du modèle d'information commun (CIM) pour les marchés*

IEC 62325-450:2013, *Cadre pour les communications pour le marché de l'énergie – Partie 450: Règles de modélisation de profils et de contextes*

XML Schema Part 1: Structures Second Edition W3C Recommendation 28 October 2004 (disponible en anglais seulement)

IETF RFC 3986 Uniform Resource Identifier (URI): Generic Syntax January 2005 (disponible en anglais seulement)

Semantic Annotations for WSDL and XML Schema W3C Recommendation 28 August 2007 (disponible en anglais seulement)

3 Termes et définitions

Pour les besoins du présent document, les termes et définitions de l'IEC TS 61970-2, ainsi que les suivants s'appliquent.

NOTE Se référer à l'IEC 60050, Vocabulaire Electrotechnique International, pour les définitions générales du glossaire.

3.1
artéfact
élément d'un modèle qui représente les objets d'un domaine donné et leurs caractéristiques

3.2
modèle canonique
modèle abstrait représentant tous les principaux objets dans un domaine donné (énergie, électricité…) à l'aide d'artéfacts

3.3
modèle d'information commun
CIM
modèle canonique (modèle abstrait) représentant tous les principaux objets dans une entreprise de service public de distribution d'électricité généralement nécessaires pour modéliser les aspects opérationnels d'un service public

Note 1 à l'article: Le terme CIM est défini dans les séries IEC 61968, IEC 61970 et IEC 62325.

Note 2 à l'article: L'abréviation "CIM" est dérivée du terme anglais développé correspondant "Common Information Model".

3.4
modèle contextuel
sous-ensemble restreint d'artéfacts du CIM

3.5
langage de balisage extensible
XML
langage de marquage qui définit un ensemble de règles pour le codage des documents dans un format lisible à la fois par l'homme et la machine

Note 1 à l'article: Ce terme est défini dans la spécification XML élaborée par le World Wide Web Consortium (W3C).

Note 2 à l'article: L'abréviation "XML" est dérivée du terme anglais développé correspondant "eXtensible Markup Language".

3.6
profil
sous-ensemble nommé de façon unique de classes CIM, d'associations et d'attributs nécessaires pour réaliser un type spécifique d'interface

Note 1 à l'article: Un profil, tel qu'utilisé dans le présent document, est défini dans l'IEC 62325-450:2013 et l'IEC 62361-101[1].

3.7
format de description de ressource
RDF
famille de spécifications du W3C conçue à l'origine comme un modèle de données de métadonnées

Note 1 à l'article: Ce terme a fini par être utilisé comme une méthode générale pour la description conceptuelle ou la modélisation d'informations qui sont mises en œuvre dans des ressources web, en utilisant une diversité de formats de syntaxe.

Note 2 à l'article: L'abréviation "RDF" est dérivée du terme anglais développé correspondant "Resource Description Format".

[1] A l'étude.

3.8
annotation sémantique pour le WSDL et le schéma XML
SAWSDL
ensemble d'attributs d'extension pour le langage WSDL (Web Services Description Language) et le langage de définition du schéma XML

Note 1 à l'article: L'abréviation "SAWSDL" est dérivée du terme anglais développé correspondant "Semantic Annotation for WSDL and XML Schema".

3.9
langage de modélisation unifié
UML
langage descriptif formel et complet avec des techniques de présentation de diagrammes utilisées pour représenter des systèmes logiciels, depuis l'analyse des exigences, en passant par la conception et la mise en œuvre, jusqu'à la documentation

Note 1 à l'article: L'UML est une norme définie par l'OMG (Object Management Group). L'UML est utilisé pour décrire le CIM.

Note 2 à l'article: L'abréviation "UML" est dérivée du terme anglais développé correspondant "Unified Modelling Language".

3.10
identificateur de ressource uniforme
URI
chaîne de caractères utilisée pour identifier un nom ou une ressource, permettant l'interaction avec des représentations de la ressource sur un réseau (en général, la toile mondiale (World Wide Web)) en utilisant des protocoles spécifiques

Note 1 à l'article: Des plans spécifiant une syntaxe concrète et des protocoles associés définissent chaque URI.

Note 2 à l'article: L'abréviation "URI" est dérivée du terme anglais développé correspondant "Uniform Resource Indicator".

3.11
schéma XML
famille de spécifications du W3C utilisée pour définir la structure, le contenu et la sémantique des fichiers en langage de balisage extensible (XML)

Note 1 à l'article: Les schémas XML se trouvent généralement dans des fichiers ayant une extension "xsd". Les fichiers XSD sont utilisés pour définir des messages entre applications.

3.12
langage d'ontologies Web
OWL
famille de langages de représentation des connaissances pour des ontologies de création, caractérisés par la sémantique formelle et des sérialisations basées sur RDF/XML pour le web sémantique

Note 1 à l'article: L'OWL est avalisé par le W3C.

Note 2 à l'article: L'abréviation "OWL" est dérivée du terme anglais développé correspondant "Ontology Web Language".

4 Contexte du système

4.1 Processus de profilage

Le but du processus de profilage est de définir un modèle syntaxique qui régira les données d'instance échangées dans un contexte métier donné et dont la sémantique est définie par un modèle canonique (comme le CIM). Le processus de profilage consiste en un processus simple en deux étapes:

- Définition d'un modèle contextuel qui est un sous-ensemble du modèle canonique (un sous-ensemble qui pourrait inclure des restrictions). Dans le présent document, il n'est pris aucune hypothèse concernant les règles utilisées pour réaliser cette procédure, mais les artéfacts de modèles contextuels sont décrits dans le Tableau 1.
- Génération d'un modèle syntaxique sous la forme d'un schéma XML avec un mapping défini des artéfacts de modèles contextuels: cet aspect relève du domaine d'application de la présente norme IEC 62361-100.

L'IEC 62361-100 définit la manière dont un artéfact de modèle contextuel est mappé avec les artéfacts du schéma XML (comme les types `element`, `simple` et `complex`). Elle ne définit pas le mapping entre les artéfacts du modèle canonique et les artéfacts du schéma XML, mais tient compte du fait qu'il existe une relation entre ceux deux artefacts.

4.2 CIM

Le CIM est un modèle canonique représentant tous les principaux objets dans une entreprise de service public de distribution d'électricité généralement nécessaires pour modéliser les aspects opérationnels d'un service public. Ce modèle inclut les classes et attributs publics de ces objets, ainsi que les relations entre eux. Les classes, attributs, relations et types d'attributs comme "Primitive", "enumeration", "CIMdatatype" et "Compound" constituent les artéfacts principaux du CIM.

Le CIM est défini par les normes IEC 61968-11, IEC 61970-301 et IEC 62325-301.

Le CIM peut être enrichi avec des extensions spécifiques à un projet ou à une application. Dans ce cas, les références au CIM dans ce paragraphe peuvent être lues comme le CIM avec extensions.

4.3 Modèle contextuel

Le concept du modèle contextuel est emprunté à la méthode de modélisation Centre des Nations Unies pour la facilitation du commerce et les transactions électroniques (CEFACT/ONU) et peut être utilisé dans l'élaboration des normes du CIM. Le modèle contextuel peut être choisi parmi de nombreux formats (dont l'OWL) ou être un paquetage de sous-ensemble de l'UML.

La présente spécification ne prend pour hypothèse aucun langage spécifique de modélisation contextuelle. Toutefois, les artéfacts définis dans le Tableau 1 sont utilisés dans le présent document lorsqu'il est fait référence au modèle contextuel et l'hypothèse retenue est qu'ils peuvent être exprimés dans n'importe quel langage utilisé.

Les spécifications de mapping (voir Article 5) s'appliquent à ces artéfacts de modèles contextuels qui pourraient être représentés dans plusieurs langages. Deux représentations possibles sont données dans les annexes.

4.4 Artéfacts de modèles contextuels

4.4.1 Artéfacts de modèles contextuels et sous-ensemble du CIM

Dans le Tableau 1, les artéfacts contextuels sont définis par rapport à des artéfacts du CIM. Ici, le terme sous-ensemble est utilisé dans le sens d'un artéfact contextuel qui est un sous-ensemble d'un artéfact du CIM. Le terme sous-ensemble signifie qu'un artéfact contextuel pourrait avoir les mêmes caractéristiques que sa correspondance CIM ou un sous-ensemble de ces caractéristiques. Exemples:

- La classe "IdentifiedObject" possède quatre attributs ("mRID", "aliasName", "name" et "description") et une association "Names", c'est-à-dire ses caractéristiques. La **classe structurée** "IdentifiedObject" dans le modèle contextuel pourrait avoir les mêmes caractéristiques que sa correspondance CIM ou juste certaines de celles-ci: par

conséquent, l'artéfact contextuel "IdentifiedObject" est défini comme étant un sous-ensemble de l'artéfact CIM "IdentifiedObject".

- L'attribut "name" de la classe "IdentifiedObject" du CIM a deux caractéristiques: une cardinalité qui est facultative et un type qui est une chaîne. Dans le modèle contextuel, la **propriété simple** "name" de la **classe structurée** "IdentifiedObject" pourrait avoir les mêmes caractéristiques que sa correspondance CIM ou certaines plus restreintes de celles-ci: par exemple, la cardinalité de "name" pourrait être restreinte à être mandatory (c'est-à-dire obligatoire) et/ou une longueur de chaîne pourrait être définie. Par conséquent, l'artéfact contextuel "name" est défini comme un sous-ensemble de l'artéfact "name" du CIM.
- Le nom de rôle d'extrémité d'association "Names" de "IdentifiedObject" du CIM a une seule caractéristique: une cardinalité 0 à plusieurs. Dans le modèle contextuel, la **propriété d'objet** "Names" de la **classe structurée** "IdentifiedObject" pourrait avoir la même caractéristique que sa correspondance CIM ou une caractéristique plus restreinte: par exemple, la cardinalité de "Names" pourrait être restreinte à 1 ou "1 à plusieurs". Par conséquent, l'artéfact contextuel "Names" est défini comme un sous-ensemble de l'artéfact CIM "name".

4.4.2 Définition des artéfacts de modèles contextuels

Les artéfacts de modèles contextuels sont répertoriés et définis dans le Tableau 1.

Tableau 1 – Artéfacts de modèles contextuels

Artéfact de modèle contextuel	Définition
Classe structurée (structured class)	Sous-ensemble d'une classe CIM non stéréotypée par un élément "enumeration", "Primitive", "CIMDatatype" ou "Compound". Une classe structurée peut avoir zéro à plusieurs propriétés d'objet, propriétés compound et propriétés simples. Toute sous-classe d'une classe structurée est également une classe structurée.
Superclasse (superclass)	Par rapport à une classe structurée donnée, classe structurée plus générale dont l'étendue est un sur-ensemble de la classe structurée donnée.
Sous-classe (subclass)	Par rapport à une classe structurée donnée, classe sous-ensemble de la classe structurée donnée.
Classe Racine (root class)	Classe structurée pouvant avoir des instances autonomes qui ne sont le référent d'aucune propriété d'objet. Un modèle contextuel peut attribuer les limites de cardinalité à une classe racine limitant le nombre d'instances autonomes pouvant se produire.
Classe Union (union class)	Sous-ensemble d'une superclasse CIM non stéréotypée définie comme une union de (certaines de) ses sous-classes. Chaque membre de l'union est défini comme une classe structurée. Chacune de ces classes est une sous-classe d'une classe CIM unique donnée. Une instance d'union est une instance de l'une de ses classes structurées constituantes. Exemple: dans le CIM, "RegisteredResource" est une superclasse de "RegisteredLoad", "RegisteredTie" et "RegisteredGenerator". Dans le modèle contextuel, "RegisteredResource" pourrait être une superclasse de certaines de ces sous-classes. Lorsqu'elle a été définie comme une union, "RegisteredResource" définissait l'ensemble des sous-classes ("RegisteredLoad", "RegisteredTie"…) qui allaient être utilisées comme les classes référentes de la **propriété d'objet** "RegisteredResource" qui en l'occurrence sera une **propriété d'objet** union (voir ci-dessous). Note: Cette caractéristique est utilisée pour obtenir tous les éléments représentant des instances de sous-classes dans un ordre aléatoire.

Artéfact de modèle contextuel	Définition
Classe Compound (compound class)	Sous-ensemble d'une classe CIM définie comme "Compound" avec des restrictions supplémentaires. Une instance d'une classe compound est une valeur structurée. Elle a une ou plusieurs propriétés, mais elle n'a aucune identité distincte de la combinaison de ses valeurs de propriété.
Type de base (basic type)	Classe CIM définie comme "Primitive" (elle comprend "Integer", "Decimal", "Boolean", "Duration", "DateTime", "Date", "Time", "Float", "String"). Un sous-ensemble de la classe CIM "Float" définie comme "Primitive" et marquée comme "simple precision" ou "double precision". Un sous-ensemble de la classe CIM "String" définie comme "primitive" et marquée comme "normalized", "token", "NMTOKEN", "Name", "NCName" et "anyURI". Un type de base peut être utilisé directement dans un modèle contextuel sans autre définition. Par hypothèse, la plage de valeurs de chaque type de base correspond à celle donnée dans la recommandation *XML Schema Part 2: Datatypes* (`integer`, `decimal`, `boolean`, `duration`, `datetime`, `date`, `time`, `float`, `double`, `string`, `normalizedString`, `token`, `MNTOKEN`, `Name`, `NCName` et `anyURI` respectivement).
Type simple (simple type)	Sous-ensemble d'une classe CIM définie comme "primitive" avec des restrictions supplémentaires. Comme définie ci-dessus, la plage de valeurs de cette classe CIM correspond par hypothèse à l'un des types de données (datatypes) de la recommandation *XML Schema Part 2: Datatypes* définis ci-dessus. Les restrictions supplémentaires limitent cette plage de valeurs en définissant une ou plusieurs facettes pour ce type de données (p. ex.: "TwentyFourChar_String" est une chaîne dont la longueur maximale est de 24 caractères). Une instance de type simple n'a pas de propriétés simples, ni de propriétés d'objet et n'a aucune identité distincte de sa valeur.
Type de données (data type)	Sous-ensemble d'une classe CIM définie comme "CIMDatatype" avec des restrictions supplémentaires. Un **type de données** est une classe dont les instances portent une valeur et d'autres propriétés qui donnent une signification à cette valeur. La valeur du type de données et les autres propriétés du type de données pourraient être limitées par des contraintes supplémentaires. Une instance d'une classe de **type de données** est une valeur structurée. Elle a une ou plusieurs propriétés, mais elle n'a aucune identité distincte de la combinaison de ses valeurs de propriété.
Classe enumeration (enumeration class)	Sous-ensemble d'une classe CIM "enumeration".
Classe CodeList (CodeList class)	Sous-ensemble d'une classe CIM "enumeration" et marqué comme "CodeList". Chaque instance de l'énumération est associée à un "code" dont le type fait partie des "**types de base**".
Propriété simple (simple property)	Sous-ensemble d'un attribut de classe CIM avec des restrictions supplémentaires. L'objet d'une **propriété simple** est un **type simple**, un **type de base**, un **type de données** ou une classe d'**énumération**.
Propriété compound (compound property)	Sous-ensemble d'un attribut de classe CIM dont le référent est une classe définie comme "Compound".

Artéfact de modèle contextuel	Définition
Propriété d'objet (object property)	Sous-ensemble d'une association CIM avec des restrictions supplémentaires et une direction spécifique allant de la classe de référence vers la classe référente. Le référent d'une **propriété objet** est une instance d'une **classe structurée**. Les restrictions peuvent limiter les classes de référence/classes référentes ou mettre des limites sur la cardinalité de la propriété objet.
Propriété d'objet par référence (by-reference object property)	Sous-ensemble d'une association CIM, comme par propriété d'objet, défini comme **par référence**. Le référent d'une **propriété d'objet par référence** est soit une instance d'une classe structurée soit une instance externe. L'hypothèse retenue est qu'il existe une instance externe, mais elle n'est pas décrite dans le présent message. D'un point de vue pragmatique, une **propriété d'objet par référence** est mise en œuvre en citant l'identificateur du référent (p. ex.: "mRID").
Propriété d'objet union (union object property)	Propriété d'objet définie comme **union** dont la classe référente est une superclasse ou propriété d'objet dont la classe référente est une classe union. Dans le CIM, "ResourceCapacity" a une association avec "RegisteredResource", superclasse de "RegisteredLoad", "RegisteredTie" et "RegisteredGenerator". L'association a deux noms de rôles d'extrémité: "ResourceCapacity" et "RegisteredResource". Dans le modèle contextuel, "ResourceCapacity" pourrait avoir la **propriété d'objet** "RegisteredResource" dont la classe référente est "RegisteredResource". Si cette propriété d'objet est marquée comme **union** ou si la classe référente "RegisteredResource" est marquée comme **union**, la **propriété d'objet** "RegisteredResource" est une **propriété d'objet union**. Note: Cette caractéristique est utilisée pour obtenir tous les éléments représentant des instances de sous-classes dans un ordre aléatoire.
Groupe de propriété exclusive (exclusive property group)	Restriction sur une **classe structurée** par rapport à un groupe de propriétés tel qu'une seule des propriétés peut apparaître dans une instance donnée de la classe.
Propriété BasedOn (BasedOn property)	relation entre un artéfact de modèle contextuel (classe **structurée**, **propriété**, type **simple** ou **type de données)** et son artéfact CIM correspondant ("class", "attribute", "association", "Primitive", "enumeration", "CIMDatatype", "Compound").
Documentation	Description courante accompagnant une définition dans le CIM ou le modèle contextuel
Documentation catégorisée (categorized documentation)	Description courante accompagnant une définition dans le CIM ou le modèle contextuel, avec quelques propriétés de classification qui indiquent la catégorie et l'objectif de la description
Stéréotype (stereotype)	Identificateur associé à une classe ou une propriété du modèle contextuel qui qualifie son usage ou sa sémantique, mais pas le mapping de son schéma XML La signification de chaque stéréotype doit être fournie dans la documentation accompagnant le modèle contextuel.

4.5 Mapping du modèle contextuel avec le schéma XML

4.5.1 Généralités

Le mapping détermine:

- un schéma XML autonome unique pour un modèle contextuel donné;

- la syntaxe des documents d'instance XML devant être échangés;
- la relation entre les définitions dans le schéma XML et les définitions dans le modèle contextuel;
- la relation entre les éléments dans les documents XML échangés et les définitions du CIM.

Le mapping s'applique, au moment de la conception, pour mapper un profil CIM avec un schéma XML W3C.

Dans ce mapping, un profil définit la sémantique d'un seul type de charge utile de message qui sera codée en XML.

En conséquence:

- La syntaxe ou la sémantique des en-têtes éventuels qui peuvent être ajoutés aux documents d'instance lorsqu'ils sont échangés n'est pas spécifiée.
- Le modèle contextuel et son schéma XML mappé sont tous les deux des artéfacts de conception et ne sont pas nécessairement exigés au moment où les documents d'instance XML sont échangés. Mais le schéma XML correspondant doit être convenu et partagé avant l'échange.
- La méthode utilisée pour échanger les documents d'instance XML n'est pas spécifiée.

4.5.2 Traçabilité

Dans ce mapping, les relations entre les éléments dans les documents XML échangés et les définitions du modèle canonique dont le modèle contextuel est un sous-ensemble sont exprimées. Pour garder trace de ces relations, le mapping utilise une annotation sémantique conforme à la "Semantic Annotations for WSDL and XML Schema W3C Recommendation" (Recommandation du W3C relative aux annotations sémantiques pour le WSDL et le schéma XML). Le mapping avec ces annotations sémantiques est réalisé à l'aide de la propriété "**BasedOn**" de l'artéfact de modèle conceptuel.

4.6 Représentation du schéma XML

Les mappings du schéma XML sont présentés à l'aide de la notation "Représentation synthétique XML" utilisée dans la spécification du W3C: "XML Schema Part 1: Structures Second Edition". L'exemple suivant est extrait du Paragraphe 1.3 de cette spécification du W3C:

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

La notation consiste en une structuration d'une construction de schéma XML avec les conventions suivantes:

- Les attributs obligatoires sont représentés en gras, par exemple **`count`**
- Les attributs facultatifs sont représentés en caractère normal, par exemple `size`
- Les valeurs d'attribut littérales sont représentées en italique, par exemple *`medium`*
- Les valeurs d'attribut alternatives figurent entre parenthèses et sont séparées par des barres verticales, par exemple (*`large`* | *`medium`* | *`small`*), et s'il y a une valeur par défaut, elle est indiquée après un deux-points, par exemple: `medium`
- Le contenu de l'élément du schéma est introduit par `Content:`

- La grammaire du contenu est mise entre parenthèses avec une virgule de séparation pour la concaténation ou une barre verticale pour les valeurs alternatives, par exemple `(annotation, (all | any*))`
- Les opérateurs Kleenex; **?**, **+** et ***** sont respectivement utilisés le plus souvent pour une, au moins une et un nombre quelconque de répétitions
- Les simples mots en caractère normal se rapportent à des définitions ailleurs. (Contrairement à la recommandation du schéma XML, ces définitions ne sont pas liées par hyperlien dans le présent document.), par exemple `annotation`

4.7 Espaces de noms (namespaces)

Les définitions de l'espace de noms XML ne sont pas explicitement représentées dans les définitions de mapping. Le choix des préfixes d'espace de noms ne relève pas du domaine d'application de la spécification de mapping. Cependant, pour les besoins du présent document et des exemples, l'hypothèse retenue est que:

- L'espace de noms par défaut est le même que l'espace de noms cible du schéma, désigné comme étant `namespace-uri` ci-dessous
- Le préfixe `xs:` représente l'espace de noms normalisé du schéma XML, `http://www.w3.org/2001/XMLSchema`
- Le préfixe `sawsdl:` représente les annotations sémantiques pour l'espace de noms WSDL `http://www.w3.org/ns/sawsdl`
- Le préfixe `p:` représente l'espace de noms d'extension `http://iec.ch/TC57/<year>/<Profile>`

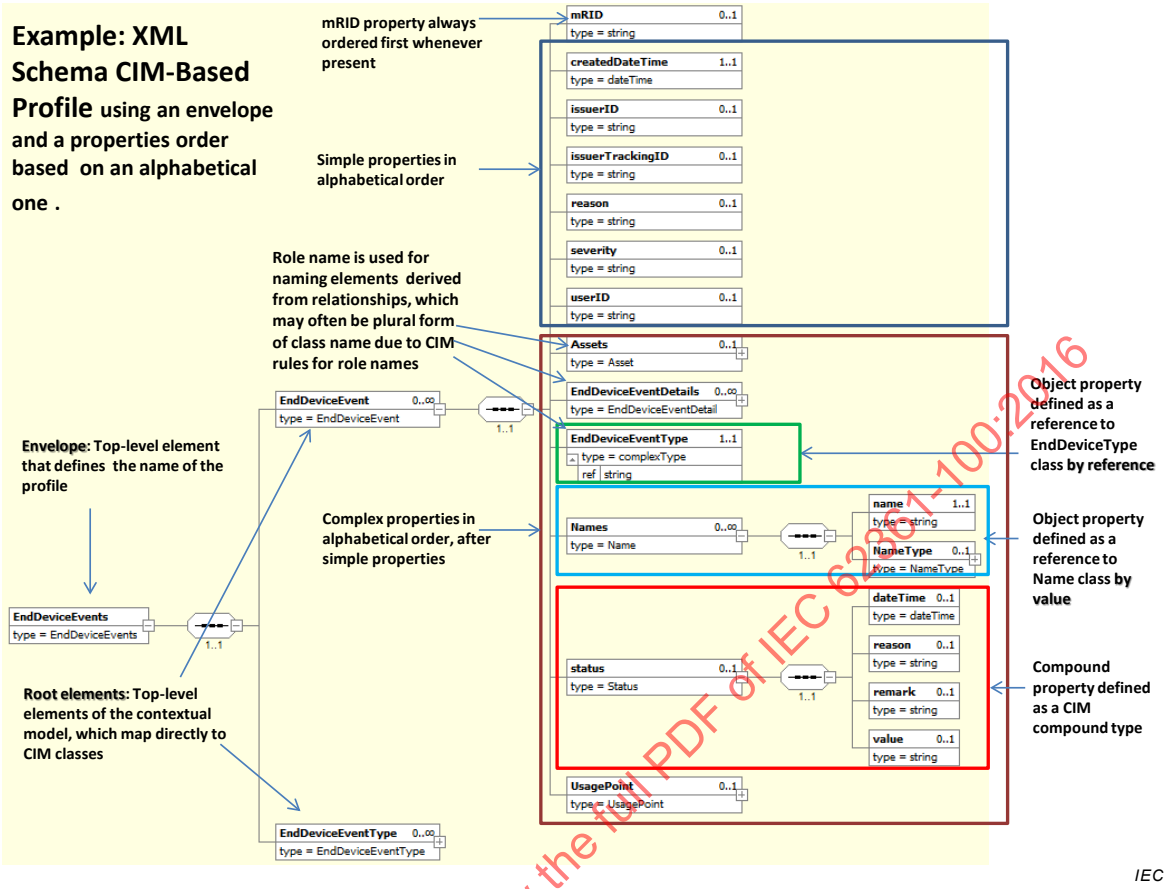
5 Spécification de mapping

5.1 Généralités

Les Paragraphes 5.2 à 5.17 définissent le mapping d'un modèle contextuel générique avec un schéma XML.

5.1.1 Exemple

La Figure 1 sert d'exemple pour décrire certaines constructions introduites dans les Paragraphes 5.2 à 5.17. Elle représente un type de mapping utilisant une enveloppe et un ordre alphabétique pour les propriétés. D'autres conceptions de schémas sont possibles comme celles n'utilisant pas une enveloppe (voir 5.2.3.3) ou n'utilisant pas un ordre alphabétique (voir 5.16.1).



IEC

Anglais	Français
Example: XML Schema CIM-based profile using an envelope and a properties order based on an alphabetical one.	Exemple: profil basé sur le CIM du schéma XML utilisant une enveloppe et des propriétés classées dans l'ordre alphabétique
Envelope: Top-level element that defines the name of the profile	Envelope (Enveloppe): élément de premier niveau qui définit le nom du profil
Root elements: Top-level elements of the contextual model, which map directly to CIM classes	Eléments root: éléments de premier niveau du modèle contextuel qui sont directement mappés avec les classes du CIM
mRID property always ordered first whenever present	Propriété mRID toujours classée en première position lorsqu'elle est présente
Simple properties in alphabetical order	Propriétés simples dans l'ordre alphabétique
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	Le nom de rôle est utilisé pour dénommer les éléments dérivés de relations qui peuvent parfois être une forme plurielle du nom de la classe en raison des règles CIM relatives aux noms de rôles
Complex properties in alphabetical order, after simple properties	Propriétés complexes dans l'ordre alphabétique, après les propriétés simples
Object property defined as a reference to EndDeviceType class by reference	Propriété d'objet définie comme une référence à la classe EndDeviceType par référence
Object property defined as a reference to Name class by value	Propriété d'objet définie comme une référence à la classe Name par valeur
Compound property defined as a CIM compound type	Propriété compound définie comme un type compound (compound type) du CIM

Figure 1 – Exemple de profil basé sur le CIM du schéma XML

5.1.2 Nom mappé

Le processus de mapping définit la manière dont un artéfact de modèle contextuel est mappé avec un artéfact XML. Tous les artéfacts ont un nom. L'une des étapes du mapping consiste à définir le nom de l'artéfact XSD par rapport au nom de l'artéfact de modèle contextuel: dans le paragraphe qui suit, il est défini comme le nom mappé de l'artéfact de modèle contextuel. Habituellement, le nom mappé correspond au nom de l'artéfact de modèle contextuel, excepté lorsque la règle de changement de nom décrite en 5.17 s'applique.

5.2 Mapping de profil

5.2.1 Généralités

Un profil unique est mappé avec un schéma XML unique sous la forme suivante:

```
<?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>
```

Les attributs `xmlns` spécifient les espaces de noms qui sont utilisés dans le schéma:

- Les espaces de noms définis en 4.7 sont ceux qui sont toujours utilisés dans le schéma.
- Une bonne pratique consiste à inclure l'espace de noms du modèle canonique CIM qui est utilisé pour la définition du modèle contextuel. Note: Dans le cas d'un modèle canonique étendu CIM, les extensions sont définies dans d'autres espaces de noms qui pourraient être inclus également.

5.2.2 Espace de noms et version

Le `namespace-uri` identifie le profil. L'espace de noms de chaque profil de norme IEC est alloué par l'IEC dans le domaine `iec.ch`.

L'attribut `version` peut être utilisé par un implémenteur ou au cours de la définition d'un profil de projet de norme.

La forme de l'attribut `version` dans ces applications ne relève pas du domaine d'application de la présente spécification.

5.2.3 Elément de premier niveau du schéma

5.2.3.1 Généralités

Il convient que l'élément de premier niveau du schéma soit un élément `Envelope` ou `Root`.

5.2.3.2 Elément Envelope (enveloppe)

Lorsqu'il est utilisé, l'élément `Envelope` est un élément qui caractérise le profil (il est le nom du profil). Il est utilisé en tant qu'élément de premier niveau du schéma lorsque le nom du profil est utilisé pour définir la charge utile. Il n'est pas défini dans le modèle contextuel et ne doit donc pas avoir le nom utilisé par une classe du modèle contextuel. A la Figure 1, il s'agit de l'élément `EndDeviceEvents`.

L'élément `envelope-elem` est la définition d'élément de premier niveau du schéma, comme suit:

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

L'élément `envelope-name` est choisi de manière à ce que la combinaison de `namespace-uri` et envelope-name soit unique parmi tous les schémas XML publiés.

L'élément `envelope-type` est une définition de type `global` de l'élément `envelope` comme suit:

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

Le contenu consiste en une définition d'élément `root-elem` locale pour chaque classe racine du profil dans un ordre alphanumérique par nom d'élément ou dans un ordre défini par le contexte métier.

Le modèle contextuel pourrait définir plusieurs classes racines. A la Figure 1, par exemple, ce sont les éléments **EndDeviceEvent** et **EndDeviceEventType**.

5.2.3.3 Elément Root (racine)

La classe **racine** est l'élément de haut niveau du modèle contextuel (il pourrait y avoir plusieurs classes racines dans un modèle contextuel). Généralement, elle est utilisée comme élément de premier niveau du schéma, selon un contexte métier, lorsque la charge utile n'est pas définie par le nom du profil.

L'élément `root-elem` est défini comme suit:

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

L'élément `root-name` est le nom mappé de la classe racine dans le modèle contextuel.

La valeur `min` est la cardinalité minimale de la classe définie dans le modèle contextuel. La valeur `max` est la cardinalité maximale définie dans le modèle contextuel. (Si `max` ou `min` est égal à 1, l'attribut `maxOccurs` ou `minOccurs` correspondant peut être omis conformément à la Spécification du schéma XML.)

5.2.4 Types

Les éléments `complex-type` et `simple-type` sont les types XML "complexTypes" and "simpleTypes" définis dans les articles qui suivent.

5.2.5 Annotation sémantique

Afin d'assurer la traçabilité, pour chacun des éléments `complexType` et `simpleType` du schéma XML définis dans les articles suivants, un attribut `modelReference` d'annotation sémantique est défini comme un URIRef absolu défini comme suit:

- l'URI de l'espace de noms du modèle d'information,
- le caractère #,
- et le nom de l'artéfact CIM correspondant (classe, attribut, association, datatypes, etc.).

Le nom de l'artéfact CIM correspondant est donné par la relation **BasedOn**. La définition d'attribut est définie comme suit:

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

où:

- `class-ref` est un URIRef désignant la classe structurée ou compound (classe CIM) dont cette classe du modèle contextuel est un sous-ensemble. A la Figure 1, `Name` est un élément `complexType` mappé à partir de la **classe structurée "Name"** contextuelle qui est elle-même un sous-ensemble de la classe CIM "Name". Ainsi, `class-ref` pour l'élément `complexType` "`Name`" est par exemple: "`http://iec.ch/TC57/2013/Cim-schema-cimXX#Name`".
- `prop-ref` est un URIRef désignant la définition de propriété (attribut ou association CIM) dont cette propriété du modèle contextuel est un sous-ensemble. A la Figure 1, `name` est un `element` mappé à partir de la **propriété simple "name"** contextuelle qui est elle-même un sous-ensemble de l'attribut CIM "name". Ainsi, `prop-ref` pour l'`element` "`name`" est par exemple: `http://iec.ch/TC57/2013/Cim-schema-cimXX#Name.name`. A la Figure 1, `Names` est un `element` mappé à partir de la **propriété d'objet "Names"** contextuelle qui est elle-même un sous-ensemble d'un nom de rôle d'extrémité CIM "Names". Ainsi `prop-ref` pour l'`element` "`Names`" est par exemple: "`http://iec.ch/TC57/2013/Cim-schema-cimXX#IdentifiedObject.Names`".
- `type-ref` est un URIRef désignant la classe ("Primitive" CIM ou classe "CIMDatatype") dont ce type est un sous-ensemble. A la Figure 1, `xs:dateTime` est le type d'`element` "`createdDateTime`", `xs:dateTime` est mappé à partir du **type de base "DateTime"** contextuel qui correspond à la Primitive CIM "DateTime". Ainsi, `type-ref` pour l'élément `xs:dateTime` est par exemple: "`http://iec.ch/TC57/<year>/Cim-schema-cimXX#DateTime`".
- `enum-ref` est un URIRef désignant l'énumération ("enumeration" CIM) dont cette **énumération** du modèle contextuel est un sous-ensemble. Exemple: `UsagePointConnectedKind` est un élément `simpleType` mappé à partir de la **classe d'énumération "UsagePointConnectedKind"** contextuelle qui est un sous-ensemble de l'énumération CIM "UsagePointConnectedKind". Ainsi, `enum-ref` pour l'élément `UsagePointConnectedKind` est par exemple: "`http://iec.ch/TC57/<year>/Cim-schema-cimXX#UsagePointConnectedKind`".
- `enum-value-ref` est un URIRef désignant la valeur "enumeratedLiteral" de l'"enumeration" CIM dont cette **énumération** du modèle contextuel est un sous-ensemble. Exemple: `connected` est l'une des valeurs d'énumération `simpleType` "`UsagePointConnectedKind`", mappée à partir de la valeur d'énumération **"connected"** contextuelle de la **classe d'énumération** UsagePointConnectedKind contextuelle, qui est un sous-ensemble des littéraux énumérés CIM "UsagePointConnectedKind". Ainsi, `enum-value-ref` pour la valeur d'énumération "connected" est par exemple: "`http://iec.ch/TC57/<year>/Cim-schema-cimXX#UsagePointConnectedKind.connected`".

- `codelist-ref` est l'URIRef désignant l'"énumération" dont cette classe du modèle contextuel est un sous-ensemble.

L'URIRef est conforme à l'IETF RFC 3986.

5.3 Classes structurées

Chaque **classe structurée** dans le modèle contextuel, y compris la **classe racine**, est mappée avec une définition `complex-type` globale comme suit:

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

L'élément `class-name` est le nom mappé de la classe du modèle contextuel.

L'élément `class-ref` est un URIRef désignant la classe (classe CIM) dont cette classe du modèle contextuel est un sous-ensemble.

Le contenu `annotation` apporte la documentation issue du modèle contextuel et du CIM. Cela est décrit au paragraphe Annotation.

Le contenu restant dépend du fait que la classe du modèle contextuel est ou non une sous-classe qui a une superclasse du modèle contextuel. Si la classe contextuelle est une **sous-classe**, la forme `derived-class` (classe dérivée) s'applique comme suit:

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

L'élément `super-name` est le nom mappé de la **superclasse** de la classe définie dans le modèle contextuel.

La forme whole-class s'applique comme suit:

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

Le contenu consiste en une définition d'élément `local` pour chaque propriété définie dans le modèle contextuel comme un membre de la classe.

Les définitions d'éléments XML apparaissent dans l'ordre spécifié en 5.16. Cependant, si l'un des éléments XML est le résultat du mapping de la **propriété simple mRID** du modèle contextuel mRID, cet élément XML apparaît le premier.

La forme de chaque définition d'élément XML dépend du type de la propriété et est décrite dans les articles suivants.

5.4 Classes compound

Chaque **classe compound** est mappée avec une définition `complex-type` globale comme suit:

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

L'élément `class-name` est le nom mappé de la classe compound du modèle contextuel.

L'élément `class-ref` est un URIRef désignant la classe (classe CIM) dont cette classe compound du modèle contextuel est un sous-ensemble.

Le contenu `annotation` apporte la documentation issue du modèle contextuel et du CIM. Cela est décrit au paragraphe Annotation.

La forme `whole-compound` s'applique comme suit:

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

Le contenu consiste en une définition d'élément `local` pour chaque propriété définie dans le modèle contextuel comme un membre de la classe compound.

Les définitions d'éléments XML apparaissent dans l'ordre spécifié en 5.16.

La forme de chaque définition d'élément XML dépend du type de la propriété et est décrite dans les articles suivants.

5.5 Types de base

Les **types de base** répertoriés au Tableau 2 ont des mappings fixes avec les types de données de la recommandation XML Schema 1.1 Part 2: Datatypes du W3C. Aucune définition n'est créée pour ces types dans le schéma XML.

Tableau 2 – Types de base

Type de base	Type mappé
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 Types simples

5.6.1 Règles de mapping

Chaque **type simple** défini dans le modèle contextuel est mappé avec une définition `simpleType` globale comme suit:

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

L'élément `type-name` est le nom mappé du **type simple** du modèle contextuel.

L'élément `type-ref` est un URIRef désignant la classe (classe CIM) dont ce type est un sous-ensemble.

L'élément `base-type` s'applique comme suit:

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

L'élément `xs:type` est le nom qualifié de l'un des types de données de la recommandation XML Schema Part 2: Datatypes spécifiés dans le modèle contextuel (voir 5.5).

Chaque élément `facet` est une restriction d'une facette du schéma XML correspondant à une restriction de facette donnée dans le modèle contextuel pour ce **type simple**.

La définition `facet` a la forme suivante:

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

L'élément `facet-name` est l'un des noms de facettes admis définis dans la recommandation *XML Schema Part 2: Datatypes* et décrits dans le Tableau 3.

EXEMPLE:

```
<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>
```

Cet exemple représente la facette `minInclusive` qui prend un entier non négatif comme `facet-value`.

5.6.2 Facettes possibles

Les facettes possibles sont répertoriées pour chaque **type de base** dans le Tableau 3.

Tableau 3 – Facettes

Type de base	Facette
Boolean	*Pas de facette*
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

Ces facettes sont définies dans la recommandation *XML 1.1 Schema Part 2: Datatypes* du W3C.

5.7 Mapping des types de données

Chaque élément **type de données** défini dans le modèle contextuel est mappé avec une définition `complexType` globale comme suit:

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

L'élément `datatype-name` est le nom mappé du nom de **type de données**.

L'élément `datatype-ref` est un URIRef désignant la classe "CIMDatatype" (classe CIM) dont ce type de données du modèle contextuel est un sous-ensemble.

Exemple: dans le CIM, "ActivePower" est un "CIMDatatype". Dans le modèle contextuel, un artéfact correspondant est le **type de données ActivePower** qui est mappé avec un élément complexType dont le `datatype-name` sera "ActivePower" et `datatype-ref` sera "http://iec.ch/TC57/2013/CIM-schema-cimXX#ActivePower".

L'élément `simple-content` s'applique comme suit:

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

L'élément "`base-type`" dépend du référent de la propriété de valeur du type de données comme suit:

- Si le référent est l'un des **types de base** répertoriés en 5.5, l'élément `base-type` est le nom de **type de données** fourni dans la recommandation XML Schema Part 2.
- Si le référent est une classe d'**énumération**, alors le `base-type` est son nom mappé. Cela désigne la définition de type correspondante décrite en 5.8.
- Si le référent est l'un des **types de base** répertoriés en 5.5 avec des restrictions supplémentaires, l'élément `base-type` est un type simple global dont le nom mappé est le nom de **type de données** accolé au terme "-base". La définition du type simple est construite conformément à la description en 5.6, excepté que les éléments `annotation` et `type-ref` sont omis.

Exemple: dans le CIM, "ActivePower" est un "CIMDatatype" qui a un attribut "value" dont l'espace de valeurs est défini par un type qui est la "Primitive" Float du CIM. Dans le modèle contextuel, l'artéfact correspondant est le **type de données** "ActivePower" qui a une propriété "value" et un certain nombre d'autres propriétés ("unit" et "multiplier"). Le mapping de la propriété "value" dépend de son type. Le type de la propriété "value" pourrait être de trois types, conduisant à trois types de mappings:

- L'espace de valeurs ou le type de la propriété "value" est le **type de base** "SingleFloat", l'élément `base-type` de simpleContent sera "xs:float".
- L'espace de valeurs ou le type de la propriété "value" est restreint à une **énumération** nommée, "ActivePowerValueKind", par exemple, l'élément `base-type` de simpleContent sera "ActivePowerValueKind".
- L'espace de valeurs ou le type de la propriété "value" est restreint à un "SingleFloat" qui est positif (facette "minInclusive" = "0"). L'élément `base-type` de simpleContent sera "ActivePower-base". Ce base-type est un élément `simpleType` dont le nom est "ActivePower-base". L'élément `simpleType` est une restriction de `xs:float` avec une restriction "minInclusive" dont la valeur est "0" (voir l'article relatif au mapping des éléments simpleType et l'exemple ci-dessous).

La définition `attribute-elem` (en tant que définition d'attribut locale) s'applique comme suit:

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

L'élément `prop-name` est le nom mappé de la propriété de type de données (autre que celle de la "value"). Exemple: "unit", "multiplier"…

La forme de l'élément `type-name` dépend du référent de la propriété de **type de données** comme suit:

- Si le référent est l'un des **types de base** répertoriés en 5.5, l'élément `type-name` est le nom de type de données fourni dans la recommandation XML Schema Part 2.
- Si le référent est une classe d'**énumération**, alors l'élément `type-name` est son nom mappé. Cela désigne la définition de type correspondante décrite en 5.8.

La chaîne `default` est la valeur de la valeur par défaut établie pour la propriété de type de données du modèle contextuel.

La chaîne `fixed` est la valeur fixe établie pour la propriété de **type de données** du modèle contextuel.

Les deux attributs `fixed` et `default` sont exclusifs s'ils sont utilisés.

La valeur `use` est `optional` si la propriété **type de données** est facultative, `required` si elle obligatoire ou `prohibited` si elle n'est pas admise dans le modèle contextuel. Dans le premier cas, l'élément `optional` pourrait être omis.

Exemple: dans le modèle contextuel, le **type de données** "ActivePower" a une propriété value dont l'espace de valeurs est un **type de base** "SingleFloat" restreint à une valeur positive. Le **type de données** "ActivePower" a aussi deux autres propriétés, à savoir "unit" et "multiplier". La propriété "unit" a une valeur fixe "W" et la propriété "multiplier" a une valeur par défaut "none". Le mapping du résultat sera comme suit:

```
<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 Mapping des classes enumeration

Chaque **classe d'énumération** définie dans le modèle contextuel est mappée avec une définition `enum-type` globale comme suit:

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

L'élément `enum-name` est le nom mappé de la classe du modèle contextuel.

Par conséquent, l'élément `enum-ref` est un URIRef désignant l'énumération (énumération CIM) dont cette énumération du modèle contextuel est un sous-ensemble.

Les `enum-values` sont comme suit:

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

Chaque élément `enum-value` représente une valeur de l'**énumération** définie dans le modèle contextuel comme suit:

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

L'élément `enum-value` est le nom mappé de la valeur enumeration.

L'élément `enum-value-ref` est un URIRef désignant la valeur "enumeration" de l'énumération dont cette **énumération** du modèle contextuel est un sous-ensemble.

5.9 Mapping des classes CodeList

Chaque **classe CodeList** définie dans le modèle contextuel est mappée avec une définition `codelist-type` globale comme suit:

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

L'élément `codelist-name` est le nom mappé de la classe du modèle contextuel.

L'élément `codelist-ref` est un URIRef désignant la valeur "enumeration" ou "CodeList" dont cette classe du modèle contextuel est un sous-ensemble.

L'élément `codelist-value` est défini comme suit:

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

L'élément `codelist-type` est le nom mappé du type de code.

Chaque élément `codelist-value` représente une valeur du code définie dans le modèle contextuel comme suit:

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

L'élément `codelist-value` est le nom mappé de la valeur du code.

L'élément `enum-value-ref` est un URIRef désignant la valeur enumeration correspondante dans la classe "CodeList" dont cette **classe CodeList** est un sous-ensemble.

L'élément `codelist-value annotation` est défini comme suit (voir 5.14.3):

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

L'élément `catdocument-elem` est défini comme suit:

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

`p:notes` est la description associée à la valeur de l'énumération.

`content` est le nom mappé de la valeur associée de l'énumération.

5.10 Mapping des propriétés simples

Chaque **propriété simple** dans le modèle contextuel est mappée (en tant que définition d'élément locale) avec un élément `simple-elem` comme suit:

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

A la Figure 1, des exemples de **propriétés simples** sont les éléments "createdDateTime" (dont le type est un **type de base** "Datetime", mappé ici avec `xs:datetime`) et "reason" (dont le type est un **type de base** "String", mappé ici avec `xs:string`).

L'élément `prop-name` est le nom mappé de la propriété.

La valeur `min` pour l'attribut `minOccurs` est 0 si la propriété est optional (facultative) dans le modèle contextuel ou 1 si elle est mandatory (obligatoire). (Dans le second cas, l'attribut `minOccurs` peut être omis, conformément à la Spécification de schéma XML.)

L'élément `prop-ref` est un URIRef désignant la définition de propriété (propriété CIM) dont cette propriété du modèle contextuel est un sous-ensemble – Exemple à la Figure 1: "createdDateTime" est un élément de l'élément "EndDeviceEvent". Dans le modèle contextuel, "createdDateTime" est une **propriété simple** de la classe "EndDeviceEvent". Cette **propriété simple** est reliée à l'attribut correspondant du CIM. Dans le CIM, l'attribut correspondant correspond à l'élément "createdDateTime" de la classe "ActivityRecord" et cet attribut est hérité par "EndDeviceEvent". Ainsi, l'élément `prop-ref` est l'URIRef: http://iec.ch/TC57/2013/Cim-schemacimXX#ActivityRecord.createdDatetime.

La forme de l'attribut `type` dépend du référent de la propriété simple comme suit:

- Si le référent est l'un des **types de base** répertoriés en 5.5, il n'y a pas de définition simpleType dans le schéma XML et l'élément `type-name` est le nom de type de données fourni dans la recommandation XML Schema Part 2. Exemple: dans le modèle contextuel, le type "issuerID" est un **type de base** "String". Ainsi, l'élément `type-name` mappé est `xs:string`.
- Si le référent est un **type simple**, alors l'élément `type-name` est son nom mappé. Cela désigne la définition de type correspondante décrite en 5.6. Exemple: si dans le modèle contextuel, le type "issuerID" était un nom de type simple "Char24_String" (p. ex.: String de 24 caractères), l'élément `type-name` mappé sera "`Char24_String`".
- Si le référent est un **type de données**, alors l'élément `type-name` est son nom mappé. Cela désigne la définition de type correspondante décrite en 5.7. Exemple: dans le CIM, l'attribut "ratedVoltage" de "EndDeviceInfo" a le CIMDatatype "Voltage" pour type. Dans le modèle contextuel, la **propriété simple** "ratedVoltage" aura le **type de données** "Voltage" pour type. Ainsi, l'élément `type-name` mappé sera "`Voltage`".
- Si le référent est une **classe d'énumération**, alors l'élément `type-name` est son nom mappé. Cela désigne la définition de type correspondante décrite en 5.8. Exemple: dans le CIM, l'attribut "technology" de "ComFunction" a l'énumération "ComTechnologyKind" pour type. Dans le modèle contextuel, la **propriété simple** "technology" aura le type **énumération** "ComTechnologyKind". Ainsi, l'élément `type-name` mappé sera "`ComTechnologyKind`".
- Si le référent est une **classe codelist**, alors l'élément `type-name` est son nom mappé. Cela désigne la définition de type correspondante décrite en 5.9.

5.11 Mapping des propriétés compound

Chaque **propriété compound** dans le modèle contextuel a pour référent une **classe compound** et est mappée (en tant que définition d'élément locale) avec un élément `compound-elem` comme suit:

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

A la Figure 1, l'élément "status" sous "EndDeviceEvent" représente un exemple de **propriété compound**.

L'élément `prop-name` est le nom mappé de la propriété.

L'élément `compound-name` est le nom mappé de la **classe compound** référente.

La valeur `min` est 0 si la propriété est optional (facultative) dans le modèle contextuel ou 1 si elle est mandatory (obligatoire). (Dans le second cas, l'attribut `minOccurs` peut être omis, conformément à la Recommandation relative au schéma XML.)

L'élément `prop-ref` est un URIRef désignant la définition de propriété (propriété CIM) dont cette propriété du modèle contextuel est un sous-ensemble.

5.12 Propriétés d'objet

5.12.1 Vue d'ensemble des règles de mapping

Chaque **propriété d'objet** dans le modèle contextuel est mappée (en tant que définition d'élément locale) avec un élément `typed-elem`, un élément `ref-elem` ou un élément `union-elem`.

5.12.2 Mapping des propriétés d'objet typées

Si le référent de l'objet de la propriété est une **classe structurée**, la forme `typed-elem` s'applique:

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

A la Figure 1, l'élément "Names" sous "EndDeviceEvent" représente un exemple de **propriété d'objet typée**. Elle est facile à reconnaître, car elle conduit à une imbrication, à savoir une autre définition de sous-éléments: "name" et "NameType".

L'élément `prop-name` est le nom mappé de la propriété (dans l'exemple: "Names"). L'élément `class-name` est le nom mappé de la classe référente (dans l'exemple: "Name").

La valeur `min` est la cardinalité minimale de la propriété définie dans le modèle contextuel. La valeur `max` est la cardinalité maximale de la propriété définie dans le modèle contextuel. (Si `max` ou `min` est égal à 1, l'attribut `maxOccurs` ou `minOccurs` correspondant peut être omis conformément à la Spécification du schéma XML.)

L'élément `prop-ref` est un URIRef désignant la définition de propriété (propriété CIM) dont cette propriété du modèle contextuel est un sous-ensemble.

5.12.3 Mapping des propriétés d'objet par référence

Si la propriété objet est définie comme "**par référence**", alors elle est mappée avec un élément `ref-elem` comme suit:

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

A la Figure 1, l'élément "EndDeviceEventType" sous "EndDeviceEvent" représente un exemple de **propriété d'objet par référence**. Elle est facile à distinguer d'une **propriété d'objet typée**, car elle conduit à un élément avec juste un attribut xml `ref` et sans autre définition de sous-éléments.

Les éléments `prop-name`, `min`, `max` et `prop-ref` sont définis ci-dessus.

L'élément `reference` est défini comme suit:

```
<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>
```

L'élément `class-ref` est un URIRef désignant la classe (classe CIM) dont le référent du modèle contextuel est un sous-ensemble.

Dans une instance XML, l'attribut `ref` défini ici prend une chaîne représentant l'identificateur utilisé pour identifier le référent de la propriété: il pourrait être le "mRID" ou le "Names.name" de la classe référente de l'instance.

Le référent peut ou peut ne pas être représenté ailleurs dans l'instance. Lorsque le référent n'est pas représenté dans l'instance et il est donc externe à l'instance, il peut être défini dans le modèle contextuel comme une classe abstraite sans sous-classes concrètes.

L'attribut `referenceType` pourrait être utilisé pour définir le type d'identificateur utilisé comme contenu de l'attribut `ref`. Par exemple, si la propriété "mRID" est utilisée comme identificateur, la valeur de `referenceType` sera "mRID".

5.12.4 Mapping des propriétés d'objet union

Si la **propriété d'objet** est marquée comme **union** et son référent est une **superclasse** ou si le référent de l'objet de la propriété est une **superclasse** abstraite marquée comme **union** `(voir Annexe C pour des exemples)`, la forme `union-elem` s'applique:

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

La valeur `min` est la cardinalité minimale de la propriété définie dans le modèle contextuel. La valeur `max` est la cardinalité maximale, en prenant en considération la cardinalité définie dans le CIM et les restrictions définies dans le modèle contextuel. (Si `max` ou `min` est égal à 1, l'attribut `maxOccurs` ou `minOccurs` correspondant peut être omis conformément à la Spécification du schéma XML.)

L'élément `prop-ref` est un URIRef désignant la définition de propriété (propriété CIM) dont cette propriété du modèle contextuel est un sous-ensemble.

Le contenu est un choix de définitions d'éléments représentant des propriétés dont les référents sont les **sous-classes** de la **superclasse union** référente. Si la **propriété union** est **par référence**, la forme `branch-ref` s'applique, autrement, la forme `branch-elem` s'applique.

La forme `branch-elem` s'applique comme suit:

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

L'élément `subproperty-name` sera le résultat de la règle de changement de nom (voir 5.17 et les exemples de l'Annexe C) et pourrait être l'une des possibilités suivantes:

- le nom de la sous-classe;
- le nom de la sous-classe préfixé par le même qualificatif que celui utilisé dans le nom de la propriété objet lui-même, le qualificatif étant suivi d'un trait de soulignement (_);
- le nom de la sous-classe préfixé par le nom de propriété objet suivi d'un trait de soulignement (_);
- le `subclass-name` étant le nom mappé de la **sous-classe**.

L'élément `prop-ref` est l'URIRef désignant la définition de propriété (propriété CIM) dont cette **propriété union** est un sous-ensemble.

La forme `branch-ref` est:

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

Les éléments `subproperty-name` et `prop-ref` sont définis ci-dessus. L'élément `reference` est défini à l'article précédent.

by-Ref ou le changement de nom peut être utilisé pour prendre en charge le mapping; cependant, il est recommandé que chaque organisation décide de celui qui sera utilisé avant la mise en œuvre.

5.13 Mapping de groupes de propriétés exclusives

Chaque **propriété exclusive** est mappée avec une définition d'élément `exclusive-elem` locale comme suit:

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

Le contenu est un choix de définitions d'éléments représentant la partie des différentes propriétés du choix. Si la propriété est **par référence**, la forme `xor-ref` s'applique; autrement, c'est la forme `xor-elem` qui s'applique.

La forme `xor-elem` est:

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

L'élément `prop-name` est le nom mappé de la propriété du modèle contextuel. L'élément `class-name` est le nom mappé du référent de la propriété.

La valeur `min` est la cardinalité minimale de la propriété définie dans le modèle contextuel. La valeur `max` est la cardinalité maximale, en prenant en considération la cardinalité définie dans le CIM et les restrictions définies dans le modèle contextuel. (Si `max` ou `min` est égal à 1, l'attribut `maxOccurs` ou `minOccurs` correspondant peut être omis conformément à la Spécification du schéma XML.)

L'élément `prop-ref` est un URIRef désignant la définition de propriété (propriété CIM) dont cette propriété du modèle contextuel est un sous-ensemble.

La forme `xor-ref` est:

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

Les éléments `prop-name`, `min`, `max` et `prop-ref` sont définis ci-dessus. L'élément `reference` est défini en 5.12.3 "Propriétés d'objet par référence".

NOTE L'utilisation de cette caractéristique est laissée à la discrétion des organismes qui créent le profil.

5.14 Documentation et documentation catégorisée

5.14.1 Mapping général

Chaque élément **Documentation** ou **Categorized Documentation** (documentation catégorisée) attaché à un élément du modèle contextuel ou à son élément associé du CIM est mappé avec un élément `annotation` dans la définition du schéma XML correspondant:

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

5.14.2 Mapping de la documentation

Chaque élément **documentation** est mappé avec un `document-elem` comme suit:

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

L'élément `paragraph` spécifie un alinéa du texte d'une définition courante de l'élément du modèle contextuel.

Il est recommandé de diviser la documentation de l'UML CIM en alinéas et de l'insérer en premier. La documentation issue du modèle contextuel, s'il y en a une, peut suivre.

5.14.3 Mapping de la documentation catégorisée

Chaque élément **categorizedDocumentation** est mappé avec un élément `catdocument-elem` comme suit:

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

L'attribut "lang" spécifie dans quelle langue naturelle le contenu de la documentation est écrit.

Les attributs "category" et "subCategory" sont utilisés pour exprimer la classification de la documentation.

L'attribut "notes" est mappé avec les notes attachées à la classification de la documentation.

5.14.4 Mapping du stéréotype

Chaque **stéréotype** sera mappé avec un catdocument-elem comme suit:

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

La catégorie est "*stereotype*".

L'élément stereotype_name est le nom du **stéréotype** du modèle contextuel.

5.15 Noms

Les noms des classes, propriétés et valeurs qui apparaissent dans le modèle contextuel sont mappés avec le schéma XML de telle manière que:

- Tous les noms mappés, à l'exception des valeurs d'énumération, doivent être conformes aux *espaces de noms* dans la syntaxe XML 1.0 NCName.
- Les valeurs d'énumération doivent être mappées avec les valeurs qui sont conformes avec la syntaxe des valeurs d'attribut *XML 1.0*.

Le nom mappé est obtenu par la translittération du nom du modèle contextuel, caractère par caractère.

5.16 Ordre de mapping

5.16.1 Principes de l'ordre de mapping général

Il doit y avoir un ordre de mapping:

- soit dans l'ordre alphabétique lorsque l'ordre de mapping n'est pas défini dans le modèle contextuel.