IEC/TS 62656-2

Edition 1.0 2013-09

TECHNICAL SPECIFICATION

SPÉCIFICATION TECHNIQUE

colour inside

Standardized product ontology register and transfer by spreadsheets – Part 2: Application guide for use with the IEC common data dictionary (CDD)

Enregistrement d'ontologie de produits normalisés et transfert par tableurs – Partie 2: Guide d'application pour l'utilisation avec le Dictionnaire de données communes de la CEI (le CEI CDD)

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

About the IEC

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

About IEC publications

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

Useful links:

IEC publications search - www.iec.ch/searchpub

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It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

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

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary (IEV) on-line.

Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications CEI

Le contenu technique des publications de la CEI est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Liens utiles:

Recherche de publications CEI - www.iec.ch/searchpub

La recherche avancée vous permet de trouver des publications CEI en utilisant différents critères (numéro de référence, texte, comité d'études,…).
Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

Just Published CEI - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications de la CEI. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org

Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (VEI) en ligne.

Service Clients - webstore.iec.ch/csc

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

IEC/TS 62656-2

Edition 1.0 2013-09

TECHNICAL SPECIFICATION

SPÉCIFICATION TECHNIQUE

colour inside

Standardized product ontology register and transfer by spreadsheets – Part 2: Application guide for use with the IEC common data dictionary (CDD)

Enregistrement d'ontologie de produits normalisés et transfert par tableurs – Partie 2: Guide d'application pour l'utilisation avec le Dictionnaire de données communes de la CEI (le CEI CDD)

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX XB

ICS 01.040.01; 01.110

ISBN 978-2-8322-1114-4

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

STANDARDIZED PRODUCT ONTOLOGY REGISTER AND TRANSFER BY SPREADSHEETS –

Part 2: Application guide for use with the IEC common data dictionary (CDD)

FOREWORD

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

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

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

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

IEC 62656-2, which is a technical specification, has been prepared by subcommittee 3D, Product properties and classes and their identification, of IEC technical committee 3: Information structures, documentation and graphical symbols.

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

Enquiry draft	Report on voting
3D/202/DTS	3D/213/RVC

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

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

A list of all the parts of the IEC 62656 series under the general title *Standardized product ontology register and transfer by spreadsheets* can be found on the IEC website.

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

- transformed into an International Standard,
- 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 62656 series entitled *Standardized product ontology register and transfer by spreadsheets* defines the means and methods for registering and exchanging product ontology(ies) expressed in spreadsheet forms.

IEC 62656 consists of the following parts:

– Part 1: Logical structure for data parcels[1];

– Part 2: Application guide for use with the IEC common data dictionary (IEC CDD);

– Part 3: Interface for common information model[2].

[1] To be published.

[2] To be published.

STANDARDIZED PRODUCT ONTOLOGY REGISTER AND TRANSFER BY SPREADSHEETS –

Part 2: Application guide for use with the IEC common data dictionary (CDD)

1 Scope

This part of IEC 62656 provides an application guide for the data parcels specified in IEC 62656-1 and used for the definition of a domain data dictionary that may be imported from and exported to the IEC common data dictionary, or IEC CDD for short, maintained as the IEC 61360-4 database [1][3]. This part of IEC 62656 provides instructions for the interpretation and use of the technical specification defined in IEC 62656-1 within a software application, to avoid misuse of the data constructs available in IEC 62656-1.

This application guide contains the following items:

- principal information for implementing data parcels for data dictionaries from/to the IEC CDD,
- typical examples of how to implement typical features on data parcels,
- extension of conformance classes for implementation of parcel-based systems to import/export data parcels from/to the IEC CDD.

The following items are outside the scope of this part of IEC 62656:

- procedures for building IEC 61360 compliant domain data dictionaries,
- semantics of a standard data dictionary itself,
- theoretical explanation of the logical structure of data parcels, which is considered in IEC 62656-1,
- interface for the common information model (IEC 61970-301 [2]), which is considered in IEC 62656-3 [3].

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 61360-1, *Standard data element types with associated classification scheme for electric items – Part 1: Definitions – Principles and methods*

IEC 61360-2, *Standard data element types with associated classification scheme for electric components – Part2: EXPRESS dictionary schema*

IEC 61987-10:2009, *Industrial-process measurement and control – Data structures and elements in process equipment catalogues – Part 10: List of properties (LOPs) for industrial-process measurement and control for electronic data exchange – Fundamentals*

---

3 Numbers in square brackets refer to the Bibliography.

IEC 62656-1:—[4], *Standardized product ontology register and transfer by spreadsheets – Part 1: Logical structure for data parcels*

IEC 62720, *Identification of units of measurement for computer-based processing*

ISO 13584-42, *Industrial automation systems and integration – Parts library – Part 42: Description methodology: Methodology or structuring parts families*

ISO/IEC Guide 77-2:2008, *Guide for specification of product properties and classes – Part 2: Technical principles and guidance*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62656-1:—, as well as the following apply.

3.1
cardinality
minimum and maximum number of occurrences of elements within a collection

3.2
classification tree
inheritance tree
is-a tree
acyclic graph of classes and relations as nodes and edges, where each node represents a concept and each edge represents a specialization, or so called “is-a” relationship between the two concepts connected by the edge

3.3
composition tree
has-a tree
acyclic graph of classes and relations as nodes and edges, in which each node represents a concept and each edge represents a part-whole relationship, or so called “has-a” relationship between the two concepts connected by the edge

Note 1 to entry: A composition tree allows a node to contain several sub-nodes and is also called an aggregation.

3.4
condition property
property whose value affects a value decision of another property

3.5
ontological element
artifact instantiated by a meta class, that serves as metadata and is used to clarify the semantics of a property or class, or to add information to it

Note 1 to entry: In general, the notion of ontological element subsumes properties within; however, it often refers to the artifacts other than the properties, such as enumerations, data types, documents, units of measurement, terms, relations, etc.

Note 2 to entry: In this part of IEC 62656, all occurrences of “meta class” are replaced by “parcel sheet” for ease of understanding.

3.6
polymorphism
pattern that allows substitution of a single concept in the same context by a different concept

[4] To be published.

[SOURCE: IEC 61987-10:2009, 3.1.21, modified — "more specific (specialized)" and the notes to entry have been deleted.]

4 Overview

4.1 General

This part of IEC 62656 is an application guide for parcel users such as:

– domain experts who implement data parcels for their domain data dictionary, for registration in the IEC CDD online database by parcelling tools,
– users who download a (piece of) data dictionary from the IEC CDD online database,
– users who edit or exchange a (piece of) data dictionary,
– application vendors who develop a parcelling tool, such as an editor, viewer or equivalent.

A typical use scenario of the IEC 62656 series for the IEC CDD is depicted in Figure 1.

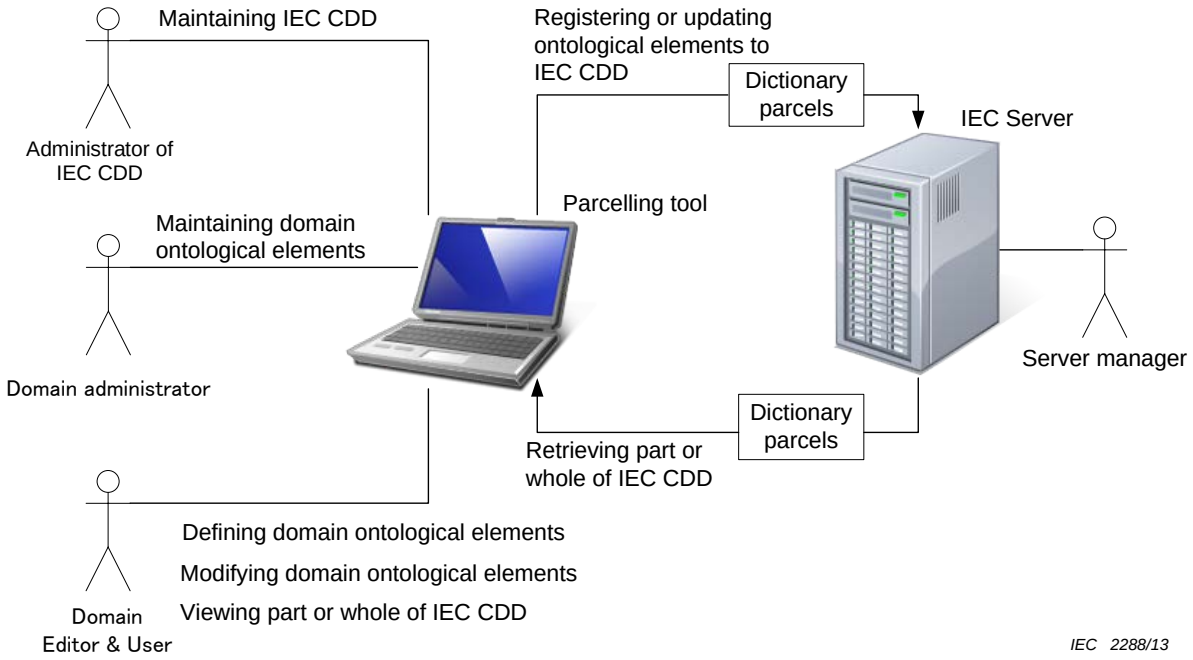


Figure 1 – Typical use scenario

For ease of reading of this part of IEC 62656, “parcel” and “attribute” are used instead of “meta-class” and “meta-property”, respectively. For example, “class meta-class” is reworded as “class parcel”.

4.2 Data dictionary

ISO/IEC Guide 77-2 recommends that each data dictionary should conform to the ISO 13584-42/IEC 61360-2 common dictionary model. In the ISO 13584-42/IEC 61360-2 common dictionary model, each data dictionary is represented by a set of classes and their associated characteristic properties. The IEC CDD is a data dictionary maintained as a database that defines an ontology of products and services, including components, materials, systems, and concepts, that are essential in electro-technical domains.

For a data dictionary, classes and properties are fundamental elements. A class is a concept embodied as a data structure for representing a real world object, such as a product, material, etc., while a property is a concept embodied as another data structure for characterizing a

class or classes. A class has a set of characteristic properties (though in extreme cases it only has one property) and shall be clearly distinguishable from other classes by the member properties in the set. In other words, if two classes have exactly the same sets of properties, those two classes are not distinguishable in a machine sensible manner. However, such a style of class modelling is not recommended in IEC 61360-1.

If there are classes which conceptually share some characteristics in common, a generalized class of those classes may be defined. Such a generalized class is called a “superclass” of those grouped classes, and each of the grouped classes is called a “subclass” with respect to the superclass. Such a relationship between a superclass and a subclass is called more familiarly an “is-a” relationship. Note that in accordance with the ISO 13584-42/IEC 61360-2 common dictionary model, each class may have one class as its superclass and each class may have multiple subclasses. For a number of classes, if there is no apparent superclass, a virtual class called a “universal class” will be assumed to exist, acting as a single common superclass. As a result, the entirety of relationships among the classes forms a tree (to be exact, an acyclic graph) structure.

If there are characteristics which are shared among several classes, such characteristics are modelled as “properties” and they shall be defined at a general class of the classes, and then inherited into its subclasses.

Figure 2 gives a simple example of a data dictionary. In this figure, the concepts of “Gasoline-powered vehicle” and “Electric vehicle” are two different kinds of vehicles, so their superclass “Vehicle” may be defined with common properties, i.e. those named “product name”, “manufacturer“ and “tyre” are defined at this level in the class tree. Likewise, “Vehicle” and “Computer” are different kinds of product concepts, so their superclass “Product” is defined with common properties, i.e. those named “product name” and “manufacturer” are defined at this level. As a consequence, the data dictionary containing those classes comprises a tree structure, as illustrated in Figure 2.

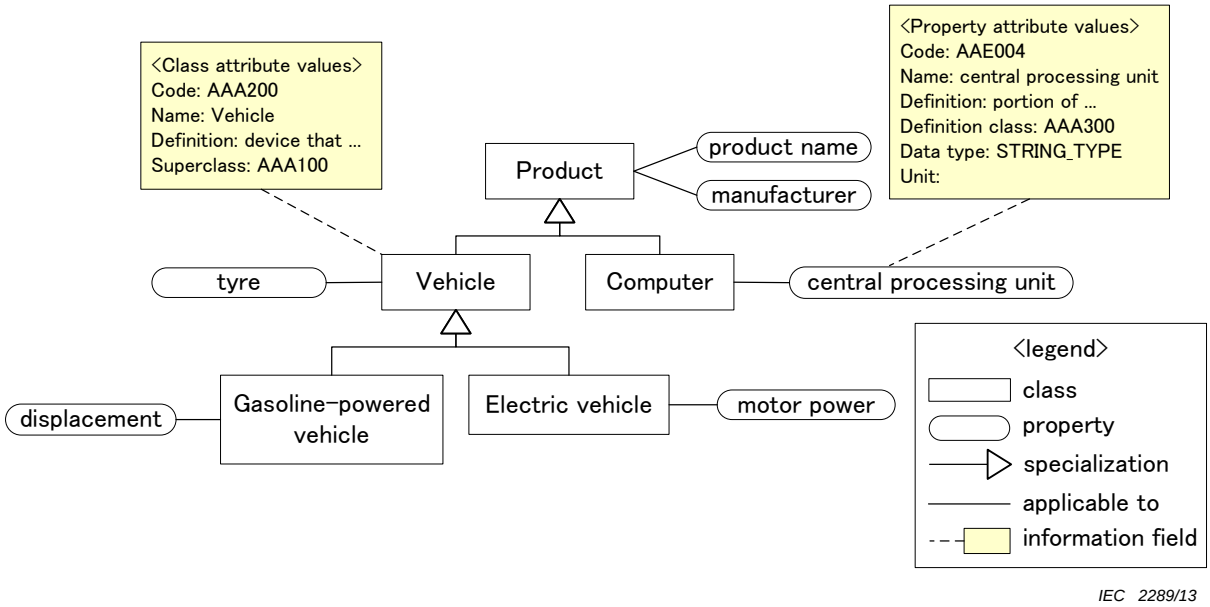


Figure 2 – Data dictionary

In accordance with the ISO 13584-42/IEC 61360-2 common dictionary model, each entity has its own unique identifier, containing structural information, to make the entity distinguishable from others. For example, a class has information fields such as name, definition, superclass, respectively, while a property has information fields such as name, definition, definition class, data type, and unit. The boxes linked by the dashed lines in Figure 2 are examples of information fields contained in a class and a property.

4.3 Data parcel

IEC 62656-1, sometimes referred to as the “parcel standard”, defines a set of containers for product ontology information, (i.e. a tree of product families and their characteristics), by sorting the information into a few homogeneous data collections, such as a list of classes, a list of properties, and a list of enumerations. Every such collection is called a “data parcel”. To be precise, a data parcel may be used not only for defining a data dictionary, but also for representing a library or catalogue of products with specific values of individual products. However, for the readers of this document, the primary interest lies in the use of data parcels to represent a product ontology. In this context, the readers are expected to see that each of the data parcels carries a homogeneous collection of product ontology. A typical form for implementation of such a data collection is a sheet within a spreadsheet, often used in day to day engineering. Thus in the rest of this Technical Specification, a data parcel will be rephrased as a “parcel sheet” to visually represent the likely form of implementation.

A class parcel (i.e. class meta-class) is for designing and instantiating classes. The class parcel has attributes (i.e. meta-properties) for describing the characteristics of a class, such as its class code, preferred name, definition and superclass. Likewise, a property parcel (i.e. property meta-class) is for designing properties. The property parcel has attributes for describing property information such as property code, preferred name, definition, definition class, data type and unit. In order to implement a data dictionary within homogeneous data collections in parcels, a few spreadsheets should be prepared, each implementing only one category of the overall parcel. For example, in the case depicted within Figure 2, a sheet for class parcel and another for property parcel are required (as shown in Figure 3).

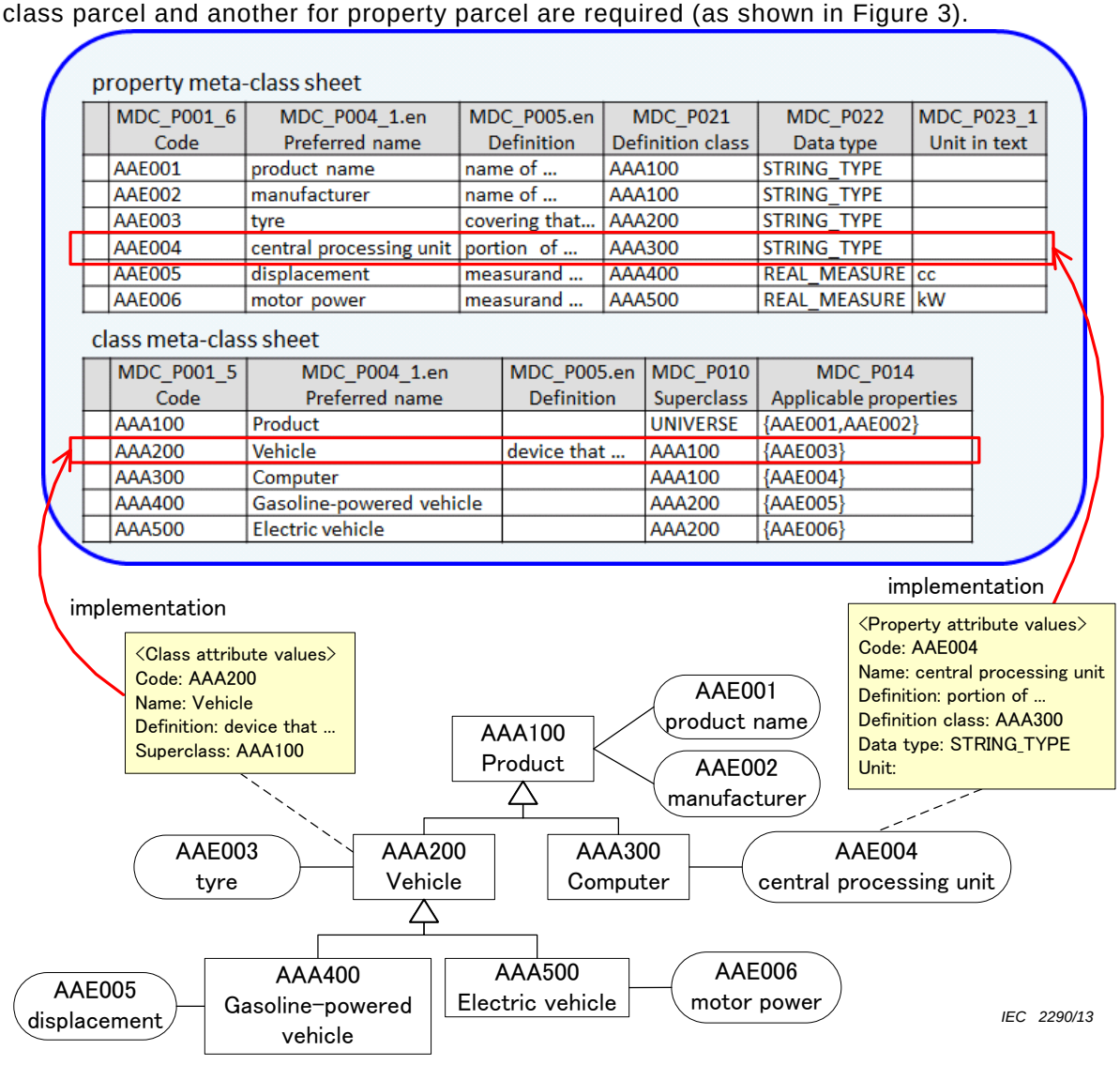


Figure 3 – Spreadsheet implementation

Figure 4 shows the basic structure of a parcel sheet. Each parcel sheet consists of its header section and data section.

The header section further consists of the class header section and schema header section. In the class header section, the information contained in the parcel sheet is described, e.g. the identifier of the data parcel and the default values which may be applied to any attribute of the parcel in the header section (see 5.2). In the schema header section, which contains metadata for describing values in the data section, each attribute of the parcel is specified in each cell column.

In the data section, each ontological element is described in each row. In each cell, the value of the attribute, which is specified in its corresponding column, is described for defining such an ontological element.

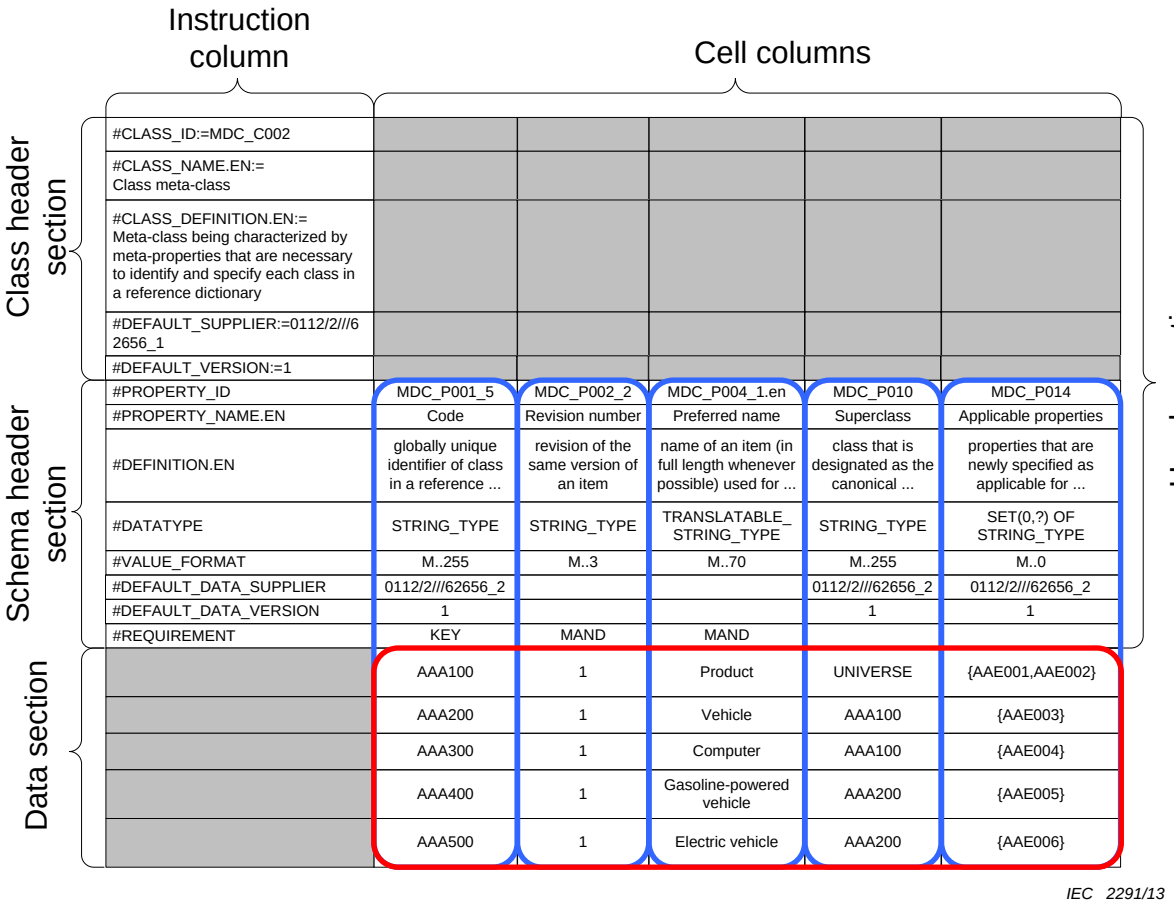


Figure 4 – Parcel sheet

4.4 Blank parcel sheets

Blank parcel sheets for editing a data dictionary from scratch will be obtained from:

– IEC CDD website at the following URL: <http://std.iec.ch/iec61360>

or can be generated by:

– a parcelling tool.

In Annex E, a list of tools that conform to this Technical Specification is given.

Four sheets comprising dictionary, class, property and supplier sheets, are mandatory for implementing a data dictionary. The other sheets are optional and they are prepared only

when they are required in the process of completing the information described in the four mandatory sheets.

5 Common cases for defining ontological elements

5.1 Semantics

In the IEC 62656 series and ISO 13584/IEC 61360 series, principal concepts of products are represented by classes, and their characteristics are modelled by properties. Some attributes of the classes and properties require the use of other parcels, such as data type, enumeration and term parcels. Thus, in many cases, determining the semantics of each ontological element modelled by the corresponding parcel will be the first step for describing the data dictionary by the data parcels. The aforementioned parcels have a basic set of attributes for describing the names and meanings of their ontological elements.

For describing names, there are 3 kinds of attributes defined in IEC 62656-1, i.e. MDC_P004_1 (Preferred name), MDC_P004_2 (Synonymous name), MDC_P004_3 (Short name). Likewise, for describing the meaning of an ontological element, or for clarifying the meaning, there are 3 kinds of attributes, i.e. MDC_P005 (Definition), MDC_P007_1 (Note) and MDC_P007_2 (Remark).

The above attributes except MDC_P004_2 are defined as TRANSLATABLE_STRING_TYPE for localization of the content which is described in a language specified as the source language. These attributes may comprise multiple columns which are identified by a specified language code (which may be combined with a country code to show a language variant, if needed). For example, if there are names in two languages, English and French, appearing in a parcel sheet, then columns identified by “MDC_P004_1.en” and “MDC_P004_1.fr” shall be prepared for describing preferred names in the sheet.

By contrast, the attribute MDC_P004_2 is defined as SET(0,?) OF LIST(2,2) OF STRING_TYPE. Therefore, only one column is provided and a synonymous name in any language shall be described as a value of this attribute. In each field of this attribute, a set containing the combination of a name and a language code (and with country code, if needed) in order is expected as a valid value. For example, if “battery” in English and its French translation “batterie” are given as the synonymous names of an ontological element, then the value will be described as “{(battery,en),(batterie,fr)}” or “{(batterie,fr),(battery,en)}”.

In each parcel sheet, there is an optional instruction #SOURCE_LANGUAGE which specifies the language in which the original semantic content in the parcel is prepared. For example, if there is a description “#SOURCE_LANGUAGE:=en” in the instruction column of a parcel, English content shall be the source and the content in the other languages shall be considered as a translation from the English content in the sheet. If no language is specified in the field, or the instruction cannot be found in the parcel sheet, by default, English shall be assumed as the source language for the sheet.

NOTE The description of values of the above attributes follows IEC 61360-6 [4].

Figure 5 gives an example of a class parcel sheet which only comprises the attributes describing the semantics of ontological elements. There are three classes described in the data section, i.e. a capacitor and its subclasses, which can be actually found in the IEC CDD.

#CLASS_ID:= MDC_C002						
#CLASS_NAME.en:= Class meta-class						
#PROPERTY_ID	MDC_ P004_1.en	MDC_P004_2	MDC_ P004_3.en	MDC_P005.en	MDC_P007_1.en	MDC_ P007_2.en
#PROPERTY_ NAME.en	Preferred name	Synonymous name	Short name	Definition	Note	Remark
#DEFAULT_ DATA_SUPPLIER						
#DEFAULT_ DATA_VERSION						
	Capacitor	{(capacitor,en)}		system of two conductors (plates) separated over the extent of its surfaces by a thin insulating medium (dielectric), its intended characteristic being capacitance	Since capacitance is a function of temperature, it may still vary with temperature.	
	Fixed capacitor	{(fixed,en)}		capacitor that has no designed provision for changing its capacitance value	Since capacitance is a function of temperature, it may still vary with temperature.	
	Variable capacitor	{(variable,en)}		capacitor designed so that its main property can be varied by mechanically changing the spatial relationship of their parts		

IEC 2292/13

Figure 5 – Semantic definitions of ontological elements

5.2 Assigning an identifier

Each ontological element shall be identified in conformance with the international concept identifier (ICID), which is defined as the primary identification scheme in IEC 62656-1. Such an identifier is not only used for distinguishing ontological elements from each other, but also for identifying a relationship between the ontological elements.

Each parcel sheet for a data dictionary contains its own attribute to describe identifiers of its ontological elements. This kind of attribute is identified as MDC_P001_X (Code), where X is a positive integer. For example, MDC_P001_5 is provided for a class parcel, and MDC_P001_6 is provided for a property parcel.

Each identifier comprises a concatenation of RAI (registration authority identifier), DI (data identifier) and VI (version identifier) in this order. Firstly, the RAI indicates the responsible supplier of a piece of data in conformity with ISO/IEC 6523 [5]. For example, “0112/2///61360_4” is the default RAI for the IEC CDD; next, the DI indicates the code assigned to each ontological element which shall be allocated uniquely within the ontological elements administered by the same RAI. In the IEC CDD, it is requested that the DI consists of six letters; the first three letters are roman-alphabetic characters and the last three letters are whole numbers (format AAANNN). Finally, the VI indicates the version number of each ontological element which shall be incremented with each new version. A new version of an ontological element shall be backward compatible with any former version of the same concept.

The capital letters “I” and “O” should be avoided in identifiers because it is difficult to distinguish such letters from numeric characters “1” and “0”, respectively, on some computer systems which can cause problems due to misreading identifiers.

IEC 62656-1 provides a means of shorthand notation for identifiers in the header section and the data section. Because almost all items in a data dictionary at each meta-layer may have a common RAI and VI, this part of IEC 62656 recommends that all applications use this mechanism for:

– simplifying the maintenance of a data dictionary,

– assigning temporary RAI and VI in designing a skeleton of a data dictionary,

– avoiding the repetition of inputting the same value of RAI and VI,
– reducing data size,
– giving better readability.

In this document, the RAI and VI of each attribute in the header section of each parcel are omitted for clarity. In a style fully conformant with IEC 62656-1, in order to omit such RAI and VI in a parcel sheet in the description, they shall be explicitly declared in the instructions noted “#DEFAULT_SUPPLIER” and “#DEFAULT_VERSION” in the class header section of the parcel sheet, but in this document, those instructions are also omitted from each figure, which is used to explain parcel sheets.

Furthermore, in this document, the RAI and VI of each attribute value are omitted for ease of reading. In a style fully conformant with IEC 62656-1, in order to omit such RAI and VI in an attribute value, they shall be explicitly declared in the instruction #DEFAULT_DATA_SUPPLIER” and “#DEFAULT_DATA_VERSION” for the attribute, respectively, but in this document, those instruction lines are also omitted in each figure, which is used to explain parcel sheets.

Figure 6 gives an example of the class parcel sheet which contains the attribute MDC_P001_5 for identification of ontological elements (i.e. the sheet is the extension of the sheet depicted in Figure 5). In the column for the attribute MDC_P001_5, the identifier of each class is described. In this example, the default RAI “0112/2///61360_4” and the default VI “003” are given, respectively. In accordance with the shorthand notation, the default RAI will be applied to the identifiers of the second ontological element (i.e. “fixed capacitor”) and the third ontological element (i.e. “variable capacitor”). Likewise, the default VI will be applied to the third ontological element.

#CLASS_ID:= MDC_C002							
#CLASS_NAME.en:= Class meta-class							
#PROPERTY_ID	MDC_P001_5	MDC_ P004_1.en	MDC_P004_2	MDC_ P004_3.en	MDC_P005.en	MDC_P007_1.en	MDC_ P007_2.en
#PROPERTY_ NAME.en	Code	Preferred name	Synonymous name	Short name	Definition	Note	Remark
#DEFAULT_ DATA_SUPPLIER	0112/2///61360_4						
#DEFAULT_ DATA_VERSION	003						
	0112/2///61360_4# AAA020##002	Capacitor	{(capacitor,en)}		system of two conductors (plates) separated over the extent of its surfaces by a thin insulating medium (dielectric), its intended characteristic being capacitance	Since capacitance is a function of temperature, it may still vary with temperature.	
	AAA021##004	Fixed capacitor	{(fixed,en)}		capacitor that has no designed provision for changing its capacitance value	Since capacitance is a function of temperature, it may still vary with temperature.	
	AAA031	Variable capacitor	{(variable,en)}		capacitor designed so that its main property can be varied by mechanically changing the spatial relationship of their parts		

IEC 2293/13

Figure 6 – Identification of ontological elements

5.3 Assigning a definition class

In accordance with IEC 62656-1, each ontological element, except for the class parcel shall have a definition class which defines its application domain. Such information shall be described as a value of the mandatory attribute MDC_P021 (Definition class) which is contained in each parcel sheet.

Properties, data types and documents shall be further applicable to classes, in or under which those ontological elements are actually used. Attributes for describing such information are provided in a class parcel sheet, those identifiers are MDC_P014 (Applicable properties), MDC_P015 (Applicable types) and MDC_P094 (Applicable documents). Further information is obtained in 6.2.

5.4 Attributes to be considered

Annexes E and F of IEC 62656-1:— define the requirement of the attributes in each parcel. In many cases, attributes which are one of KEY, NOT NULL or MANDATORY are essential or important to define ontological elements. Therefore, such attributes should be displayed.

The predefined instruction “#REQUIREMENT” indicates the requisiteness of each attribute. If there is such an instruction available in a parcel sheet, it is recommended that it should be explicitly displayed in an implementation.

6 Specifying structures for data dictionaries

6.1 General

There are two fundamental types of structures for data dictionaries, that is, classification tree and composition tree. Sometimes, the former is called an “is-a” tree or inheritance tree, and the latter is called a “has-a” tree in other literature. Both structures shall be in accordance with the principles of the ISO 13584-42/IEC 61360-2 common dictionary model.

6.2 Classification tree

Each classification tree comprises a parent-child relationship between classes where a child class (called a “subclass”) inherits all the properties of its parent class (called a “superclass”). Each class shall have just one class as its superclass and may have one or more classes as subclass(es).

Each domain data dictionary is assumed to have one root class, for logical conformity. According to the ISO 13584-42/IEC 61360-2 common dictionary model, an abstract universal class named UNIVERSE is specified as the root to collect all domain data dictionaries. The UNIVERSE class shall be a class having no properties.

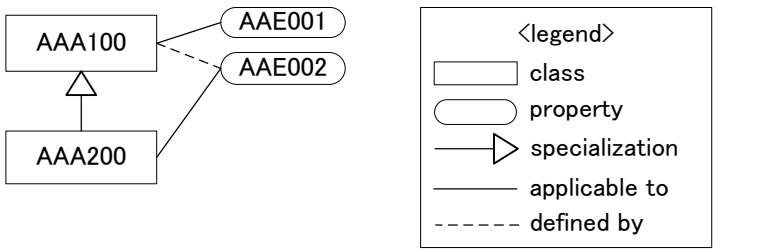
To avoid replication of the same property, data type and document in several branches, each of such ontological elements which may be commonly used by several classes should be defined within their common general class.

Enumerations, units of measurement and terms will become applicable to their definition class, while the applicability of each of the properties, data types and documents shall be explicitly declared in their definition class or a subclass corresponding to the definition class. Once such an ontological element becomes applicable to a class, this ontological element will be applicable in all the subclasses of the class.

The attribute MDC_P010 (Superclass) of class parcel specifies the identifier of a superclass of a class.

The attributes MDC_P014 (Applicable properties), MDC_P015 (Applicable types) and MDC_P094 (Applicable documents) of a class parcel specify a list of the identifiers of properties, data types and documents, respectively, which are applicable to a class. Since inherited properties, data types and documents from upper classes (known properties, data types and documents) are to be previously described as applicable in a relevant upper class, at the point where they first become applicable, attributes for describing inherited properties, data types and documents, i.e. “known applicable XYZ” shall be omitted to avoid possible inconsistency in a data parcel used for data exchange with eternal systems or tools. As any change in an upper class, with regard to an inherited property, data type or document, is to be propagated to all its subclasses, it is possible for inconsistencies to occur due to the limited and different validation capabilities of each system and tool. Such attributes may be shown within a tool as derived attributes for convenience, which shall be marked with DER for their requirement (#REQUIREMENT), but shall not be used in data exchange with external partners.

Figure 7 shows an example of a simple classification tree which contains the two classes AAA100 and AAA200 and the two properties AAE001 and AAE002. The class AAA200 is defined as a subtype of the class AAA100. In this figure, the properties AAE001 and AAE002 are defined in the class AAA100, but only the property AAE001 is applicable to the class AAA100. The property AAE002 becomes applicable to the subclass AAA200 by inheritance and as a consequence, while the class AAA100 has one applicable property, the class AAA200 has two applicable properties.



IEC 2294/13

Figure 7 – Example of a simple classification tree

Figure 8 shows conjunctive parcels to describe the classification tree shown in Figure 7. In the class parcel sheet, the two classes are defined and the parent-child relationship between the classes is described in the attribute MDC_P010. In the property sheet, the two properties are defined. The property AAE001 is not listed in the attribute MDC_P014 for the class AAA200 because the property AAE001 is a known applicable property, which is already applicable to its superclass AAA100.

#CLASS_ID:= **MDC_C002**				
#CLASS_NAME.en:= **Class meta-class**				
#PROPERTY_ID	MDC_P001_5	MDC_P010	MDC_P011	MDC_P014
#PROPERTY_ NAME.en	Code	Superclass	Class type	Applicable properties
	AAA100	**UNIVERSE**	ITEM_CLASS	**{AAE001}**
	AAA200	**AAA100**	ITEM_CLASS	**{AAE002}**

#CLASS_ID:= **MDC_C003**					
#CLASS_NAME.en:= **Property meta-class**					
#PROPERTY_ID	MDC_P001_6	MDC_P020	MDC_P021	MDC_P022	MDC_P023_1
#PROPERTY_ NAME.en	Code	Property data element type	Definition class	Data type	Unit in text
	AAE001	NON_DEPENDENT_ P_DET	**AAA100**	STRING	
	AAE002	NON_DEPENDENT_ P_DET	**AAA100**	REAL_MEASURE	m

IEC 2295/13

Figure 8 – Parcel implementation for simple classification trees

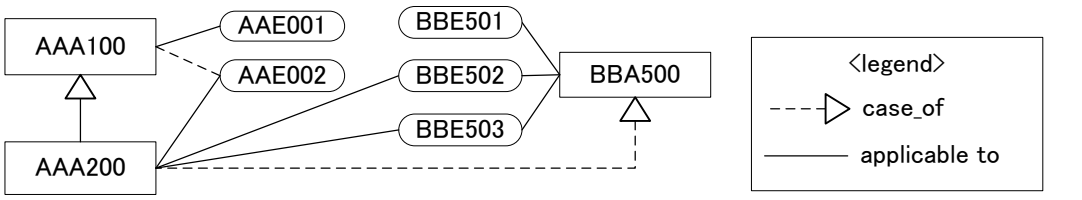
6.3 Reuse of properties, data types and documents in other branches

To enable reuse of existing properties, data types and documents defined in another branch in some classification tree, a mechanism named case_of is provided (see IEC 61360-2). The case_of mechanism enables a class to specify other class(es) located within another branch

in the same or in another data dictionary, and then to import some of the applicable properties, applicable data types and/or applicable documents from the specified class(es). Note that imported properties, data types and documents become immediately applicable to the importing class and are inherited by its subclasses.

The attributes MDC_P090 (Imported properties), MDC_P091 (Imported types) and MDC_P093 (Imported documents) specify a list of the identifiers of imported properties, data types and documents, respectively. In this case, a set of identifiers of classes from which those properties, data types and documents are imported shall be described in the attribute MDC_P013 (Is case of).

Figure 9 shows an example of the reuse of previously existing properties by importing those properties of the respective existing class. The class AAA100 is a class of the data dictionary shown in Figure 7. The class BBA500 is a class within another data dictionary which consists of one class and three properties BBE501 through BBE503. In this example, the class AAA200 is a special case of the class BBA500, that is, the class AAA200 imports certain properties from the class BBA500. As a result of the case_of relationship, the class AAA200 imports the two properties BBE502 and BBE503 from the class BBA500. Note that those properties are a subset of the applicable properties of the class BBA500.



IEC 2296/13

Figure 9 – Example of import mechanism

Figure 10 shows an updated sheet for the class parcel shown in Figure 8, to describe the data dictionary shown in Figure 9. For the class AAA200, the class BBA500 is specified in the attribute MDC_P013 (Is case of), for defining the case_of relationship between those classes. Also, the two imported properties BBE502 and BBE503 are listed in the attribute MDC_P090 (Imported properties) for the class AAA200.

#CLASS_ID:= **MDC_C002**						
#CLASS_NAME.en:= **Class meta-class**						
#PROPERTY_ID	MDC_P001_5	MDC_P010	MDC_P011	MDC_P013	MDC_P014	MDC_P090
#PROPERTY_ NAME.en	Code	Superclass	Class type	Is case of	Applicable properties	Imported properties
	AAA100	UNIVERSE	ITEM_CLASS		{AAE001}	
	AAA200	AAA100	ITEM_CLASS	**{BBA500}**	{AAE002}	**{BBE502,BBE503}**

IEC 2297/13

Figure 10 – Parcel implementation for case of relationships

6.4 Composition tree

Each composition tree comprises part-whole relationships between classes. This type of structure is often present in cases where it is necessary to represent products with several parts or materials, or to attach information for product life-cycle management.

According to IEC 62656-1, a part-whole relationship between classes shall be described by a property defined as either CLASS_REFERENCE_TYPE or CLASS_INSTANCE_TYPE. Both of the data types specify an identifier of a class (a part-class) to be a part of another class (a whole-class), but there is a difference as follows:

- CLASS_REFERENCE_TYPE is used in the case where the instances of the part class exist in a part-class (referenced by the CLASS_REFERENCE_TYPE) independent of the whole-class. For example, instances of the product “tyre” can be referenced by several instances of the product “vehicle”. For implementing the relationships between instances of “vehicle” and “tyre”, CLASS_REFERENCE_TYPE shall be used.
- CLASS_INSTANCE_TYPE is used in the case where the instances of the part class are embedded in the whole-class (i.e. where the CLASS_INSTANCE_TYPE resides). In this case, if the whole-class is destroyed, those instances of the part-class shall be destroyed.

From a slightly different viewpoint, in the case of CLASS_INSTANCE_TYPE, the referenced class is used just as a builder of a composite property, which is sometimes called an “object-type property” in other literatures.

Figure 11 shows an example of a composition relationship between two branches. In Figure 11, the class BBA500 has the class AAA200 as its part. To define the part-whole relationship between the class AAA200 (a part-class) and the class BBA500 (a whole-class), the class BBA500 has the CLASS_REFERENCE_TYPE property BBE504, which specifies the class AAA200.

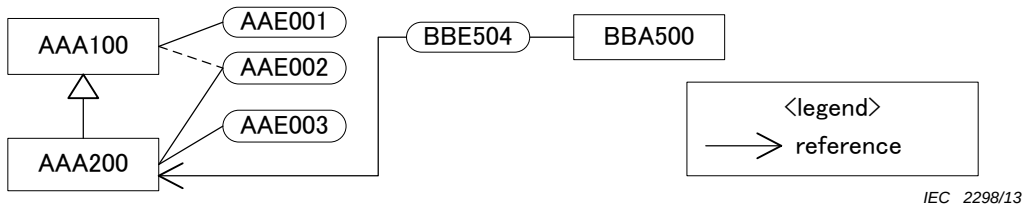


Figure 11 – Composition relationship between two branches

Figure 12 gives a composition view of the class BBA500 derived from the classification trees in Figure 11. The composition relationship between the two classes BBA500 and AAA200 is depicted as a filled diamond and a solid line.

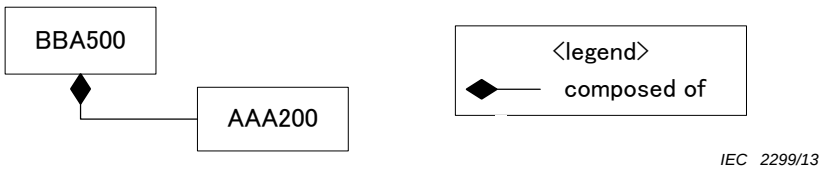


Figure 12 – Example of a composition tree

Figure 13 shows sheets of conjunctive parcels for implementing the composition tree depicted in Figure 11. The property BBE504 is listed as an applicable property to the class BBA500, to describe a part-whole relationship between the class AAA200 (a part-class) and the class BBA500 (a whole-class).

#CLASS_ID:= **MDC_C002**				
#CLASS_NAME.en:= **Class meta-class**				
#PROPERTY_ID	MDC_P001_5	MDC_P010	MDC_P011	MDC_P014
#PROPERTY_ NAME.en	Code	Superclass	Class type	Applicable properties
	BBA500	UNIVERSE	ITEM_CLASS	**{BBE504}**

#CLASS_ID:= **MDC_C003**				
#CLASS_NAME.en:= **Property meta-class**				
#PROPERTY_ID	MDC_P001_6	MDC_P020	MDC_P021	MDC_P022
#PROPERTY_ NAME.en	Code	Property data element type	Definition class	Data type
	BBE504	NON_DEPENDENT_ P_DET	BBA500	**CLASS_REFERENCE (AAA200)**

IEC 2300/13

Figure 13 – Parcel implementation for composition trees

7 Defining ontological elements by optional parcels

7.1 Defining enumerations

If it is necessary to assign a possible value list to a property, such a value list shall be defined as an enumeration in an enumeration parcel sheet and then it shall be assigned to properties in a property parcel sheet.

The attribute MDC_P044 (Enumeration code list) of the enumeration parcel is for formulating a list of values. Each value in the value list shall conform to the same data type or its sub type, e.g. if a value list is for properties which are defined with a real type, the value list shall only contain real values.

In the case that it is required to define meanings of the values in the value list, each value shall be defined as a term in a term parcel sheet and shall then be assigned to enumerations in the enumeration parcel sheet.

The attribute MDC_P025_1 (Preferred letter symbol in text) of the term parcel is used for describing a value which may be assigned as an actual value for properties.

The attribute MDC_P043 (Enumerated list of terms) of the enumeration parcel specifies a list of the identifiers of terms which may be applied to properties as their value candidates. If identifiers of terms are specified, values which are described in the attribute MDC_P044 shall be a list of values of the attribute MDC_P025_1 of the terms.

If values for both the attributes MDC_P043 and MDC_P044 are specified, then the number of elements of MDC_P043 shall be the same as the number of elements of MDC_P044.

NOTE ISO 13584-42 does not provide a way to define a meaning of a value. Thus, terms will be ignored for ISO 13584-42 compliant systems.

Figure 14 shows an example of a data dictionary which contains enumerations. There are five properties, four enumerations and seven terms. Each property is defined as a subtype of ENUM_TYPE which specifies an enumeration.

The enumerations AFA001 through AFA003 specify terms which are defined in the term parcel sheet. The AFA001 specifies the two terms AQA106 and AQA107. The enumeration AFA002 specifies the three terms AQA104 through AQA106, in which string values are specified. The enumeration AFA003 specifies the three terms AQA101 through AQA103, in which real values are specified. The enumeration AFA004 just specifies a list of four integer values, so no term is specified for each value. The term AQA106 is specified by the two enumerations AFA001 and AFA002, because the meaning of the term can be shared by those enumerations. In contrast, the two terms AQA104 and AQA107 have the same value “green”, but they are defined separately, because they indicate different kinds of “green”. The enumeration AFA003 is shared by the properties AAE202 and AAE203.

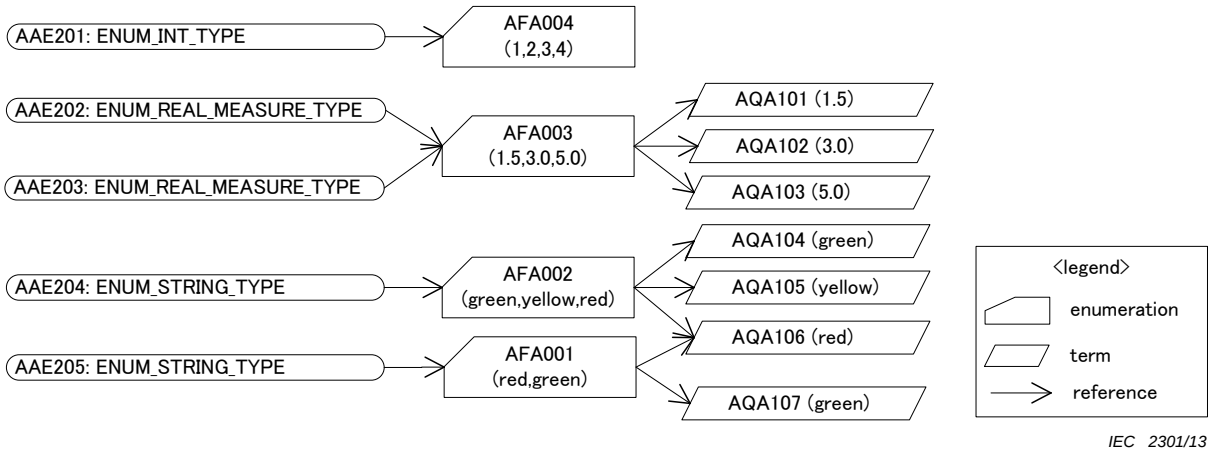


Figure 14 – Example of a use case of enumeration

Figure 15 shows sheets of conjunctive parcels to describe the data dictionary shown in Figure 14.

In the property parcel sheet, which is depicted as the first sheet, five properties are described. The column of the attribute MDC_P022 (Data type) specifies the data type of each property as a subtype of ENUM_TYPE which specifies the identifier of an enumeration and its possible value list.

In the enumeration parcel sheet, which is depicted as the second sheet, four enumerations are described. The attribute MDC_P043 (Enumerated list of terms) specifies terms which are used as value candidates for properties, while the attribute MDC_P044 (Enumeration code list) specifies actual values for properties. The enumeration AFA004 has no value for the attribute MDC_P043, so only value candidates are specified.

In the term parcel sheet, which is depicted as the third sheet, seven terms are described. Each value that can be selected as a candidate value for properties is described in the attribute MDC_P025_1 (Preferred letter symbol in text).

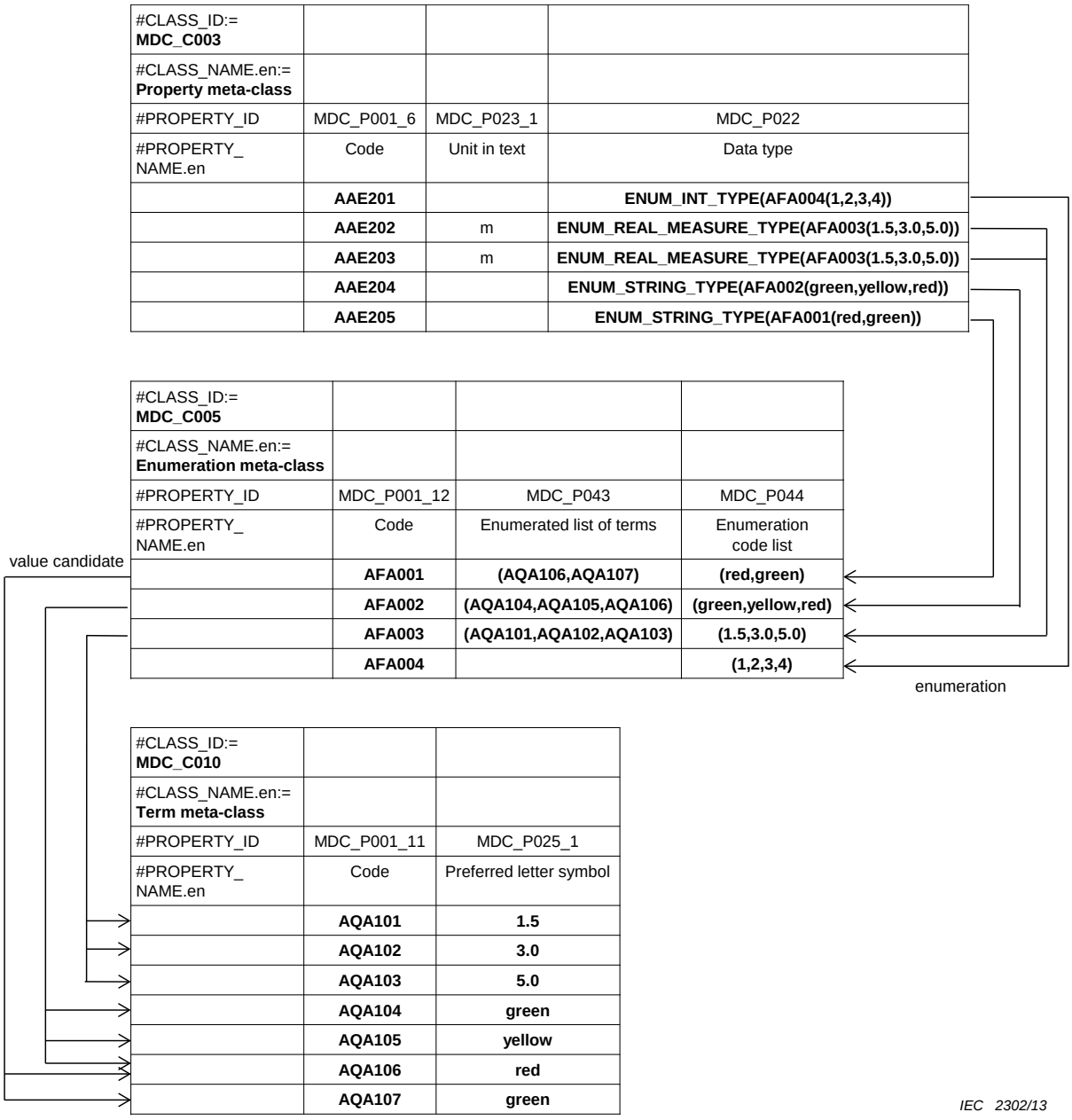


Figure 15 – Parcel implementation for enumerations

7.2 Defining named data types

In IEC 62656-1, there are many data types, most of which are based on the data types defined in IEC 61360-2, but some of them are extended. Those predefined data types are usually enough for modelling domain specific data, but occasionally there is a need for defining new data types which have their own names, for example, to assign a new name to a data type to be shared among several properties or to explicitly distinguish it from other data types, or the original data type, before associating the new name. Such data types are called “named data type” and shall be defined as instances of a data type parcel which is identified as MDC_C006 (data type meta-class).

In a data type parcel sheet, each named data type shall be described in each line in the data section. Each named data type shall have its base data type which is either a predefined data type in IEC 62656-1 or another named data type. It is assumed that any reference between named data types and nested reference among named types shall not create a loop structure.

In a similar way to the definition of a property, a named data type shall be applicable to a class in which the named data type is required for defining a property or another named data type. Such information shall be described in the attribute MDC_P015 (Applicable types) in a class parcel sheet. It is also possible to import a named data type from a class within another branch. In this case, the attribute MDC_P091 (Imported types) in the class parcel sheet shall be used to specify a set of named data types to be used and the attribute MDC_P013 (Is case of) shall be used to specify the classes from which those named data types are imported.

Properties and named data types which are defined in a class can use one of the named data types that are applicable to the class by means of the predefined data type NAMED_TYPE in their fields of the attribute MDC_P022 (Data type).

Note that if a named data type is one of the measurement types or currency types, then the named data type shall have a unit of measurement or currency, respectively. Besides, any property and another named data type which is defined as a named data type shall have the same unit of measurement or currency as the named data type. If such information is not specified in a property or another named data type, then the information of the referenced named data type will be implicitly applied to the property or named data type.

Figure 16 shows an example of sheets of conjunctive parcels to describe a data dictionary, which contains a property defined as a named data type. In this example, a property AAE211 is defined as a named data type AKA001, which is based on the predefined data type REAL_MEASURE_TYPE. Therefore, the named data type is defined in the data type parcel sheet at the bottom of the figure, and the identifier of the named data type is specified in the data type field of the property via the keyword “NAMED_TYPE”. Although the named data type is defined as REAL_MEASURE_TYPE, a unit of measurement is not specified for the property AAE211. In this case, the unit of measurement “m” of the named data type AKA001 is applied to the property AAE211 as its unit of measurement.

#CLASS_ID:= **MDC_C002**					
#CLASS_NAME.en:= **Class meta-class**					
#PROPERTY_ID	MDC_P001_5	MDC_P010	MDC_P011	MDC_P014	MDC_P015
#PROPERTY_ NAME.en	Code	Superclass	Class type	Applicable properties	Applicable types
	AAA210	UNIVERSE	ITEM_CLASS	**{AAE211}**	**{AKA001}**

applicable

#CLASS_ID:= **MDC_C003**			
#CLASS_NAME.en:= **Property meta-class**			
#PROPERTY_ID	MDC_P001_6	MDC_P021	MDC_P022
#PROPERTY_ NAME.en	Code	Definition class	Data type
	AAE211	AAA210	**NAMED_TYPE (AKA001)**

named data type

#CLASS_ID:= **MDC_C006**				
#CLASS_NAME.en:= **Datatype meta-class**				
#PROPERTY_ID	MDC_P001_7	MDC_P021	MDC_P022	MDC_P023_1
#PROPERTY_ NAME.en	Code	Definition class	Data type	Unit in text
	AKA001	AAA210	**REAL_MEASURE_TYPE**	**m**

applicable

IEC 2303/13

Figure 16 – Parcel implementation for named data types

7.3 Defining information of external resources

There are resources which exist outside of data parcels such as binary documents, still images, audio and video files. To define and specify such resources, a document parcel, which is identified as MDC_C007 (document meta-class), is used.

In a document parcel sheet, information regarding each outer resource is described in each line. Such information includes details of the organization that has responsibility to a document, how to access a document, etc.

In a similar manner to the property definition, resource information shall be applicable to a class in which the resource information is required. Such information shall be described in the attribute MDC_P094 (Applicable documents) in a class parcel sheet. It is also possible to import documents from a class within another branch. In this case, the attribute MDC_P093 (Imported documents) in the class parcel sheet shall be used to specify a set of documents to be used and the attribute MDC_P013 (Is case of) is used to specify the classes from which those documents are imported.

In other parcel sheets, the predefined attributes MDC_P004_4 (Name icon), MDC_P008_1 (Simplified drawing) and MDC_P008_2 (Graphics) are expected to have the identifier of a document which is defined in the document parcel sheet.

Figure 17 shows an example of sheets of conjunctive parcels for description of a data dictionary, which contains information of an external file referenced from a property. The information of the external file which is accessible from “http://iec.ch/image.jpg” is identified as the document AMA001, and described in the document parcel sheet at the bottom of the figure. In the property parcel sheet in the middle of the figure, the property AAE221 whose graphics is derived from the specified URL, is described. In the class parcel sheet at the top of the figure, both the property AAE221 and the document AMA001 become applicable to the class AAA220.

NOTE <http://iec.ch/image.jpg> is used as an example and is fictitious.

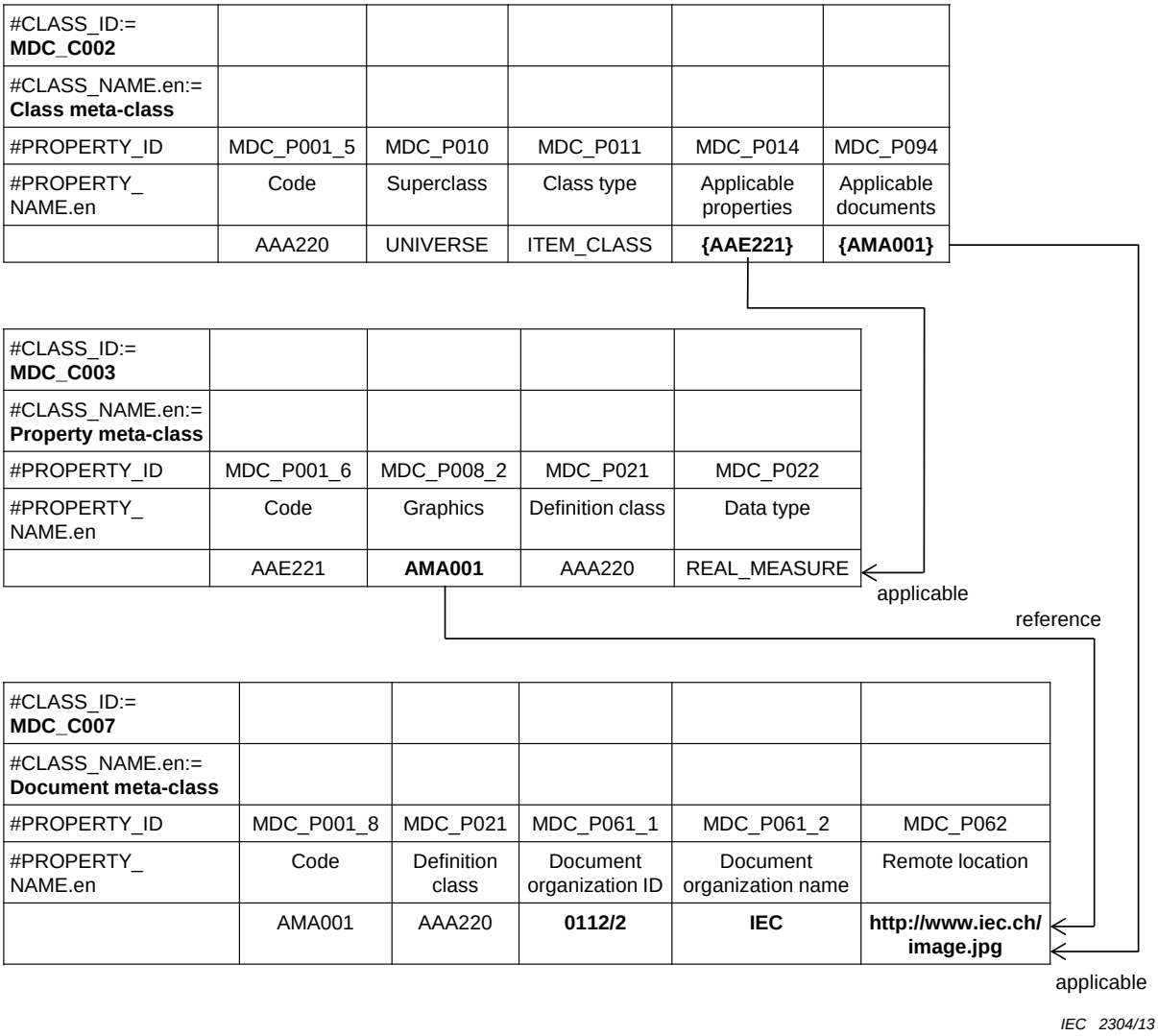


Figure 17 – Parcel implementation for document references

7.4 Defining units of measurement

The simplest way to specify a unit for a property is to describe it in the attributes MDC_P023 (Unit structure), MDC_P023_1 (Unit in text) and MDC_P023_2 (Unit in SGML). The attribute MDC_P023 is for a STEP expression of a unit [6]. Next, the attribute MDC_P023_1 is for a plain text expression of a unit. Finally, the attribute MDC_P023_2 is for an SGML [7] expression of a unit, including MathML [8] expression.

In addition to the above simple expression of a unit, there are some more complex cases in which units should be identified, such as:

– defining not only simple SI or non-SI units, but also a combination of them,

– distinguishing between units when they are described using the same reference unit, or the same set of units, but they may have different meanings due to the different domains in which they may be applied,

– permitting accurate conversion of a value from a unit to its alternative by a computer.

IEC 62720, which contains a unit dictionary for the whole electric and electronic domains, is a good example of the above requirement.

In IEC 62656-1, the unit of measurement parcel (UoM parcel), which is identified as MDC_C009 (UoM meta-class), is used for defining units of measurement. The UoM parcel sheet contains three attributes MDC_P023, MDC_P023_1 and MDC_P023_2 for describing information of a unit of measurement which is actually used by properties and named data types. Such a unit is identified by the attribute MDC_P001_10 (Code), which specifies an identifier of the defined unit.

A unit defined as an instance of a UoM parcel may be used by properties or named data types via the attributes MDC_P041 (Code for unit) and MDC_P042 (Codes for alternative units) of those parcels. The attribute MDC_P042 may have a set of identifiers of units which are alternatives to a primary unit described in the attribute MDC_P041. That means the attribute MDC_P041 shall have a value whenever the attribute MDC_P042 has a value.

Figure 18 shows sheets of conjunctive parcels consisting of a property parcel and UoM parcel. In the UoM parcel sheet, there are four units of measurement described which are based on the content defined in IEC 62720. The property AAE231 specifies “m” as the unit represented in text form. A property AAE232 specifies a unit of measurement UAA726 defined in the UoM parcel sheet. A property AAE233 specifies both a unit in text and a unit of measurement defined in the UoM parcel sheet. Therefore, the unit in text representation takes precedence for the property AAE233. The property AAE233 also specifies a set of alternative units UAB197 and UAA917 for its primary unit of measurement UAA594.

In the UoM parcel sheet, each unit of measurement defines its unit in text representation in the attribute MDC_P023_1 and in MathML description in the attribute MDC_P023_2.

#CLASS_ID:= **MDC_C003**					
#CLASS_NAME.en:= **Property meta-class**					
#PROPERTY_ID	MDC_P001_6	MDC_P022	MDC_P023_1	MDC_P041	MDC_P042
#PROPERTY_ NAME.en	Code	Data type	Unit in text	Code for unit	Codes for alternative units
	AAE231	**REAL_MEASURE**	m		
	AAE232	**REAL_MEASURE**		UAA726	
	AAE233	**LEVEL(MIN,MAX) OF REAL_MEASURE**	kg	UAA594	{UAB197,UAA917}

#CLASS_ID:= **MDC_C009**				
#CLASS_NAME.en:= **UoM meta-class**				
#PROPERTY_ID	MDC_P001_10	MDC_P004_1.en	MDC_P023_1	MDC_P023_2
#PROPERTY_ NAME.en	Code	Preferred name	Unit in text	Unit in SGML
	UAA726	metre	**m**	**<math xmlns="http://www.w3.org/1998/Math/MathML" display="block"><mrow><mrow><mi>m</mi></mrow></mrow></math>**
	UAA594	kilogram	**kg**	**<math xmlns="http://www.w3.org/1998/Math/MathML" display="block"><mrow><mrow><mi>k</mi><mi>g</mi></mrow></mrow></math>**
	UAB197	Pound (US)	**lb**	**<math xmlns="http://www.w3.org/1998/Math/MathML" display="block"><mrow><mrow><mi>lb</mi><mspace width="0.3em"/> <mfenced><mrow><mi>US</mi></mrow></mfenced></mrow></mrow></math>**
	UAA917	Ounce	**oz**	**<math xmlns="http://www.w3.org/1998/Math/MathML" display="block"><mrow><mi>oz</mi></mrow></math>**

primary unit

alternative units

IEC 2305/13

Figure 18 – Parcel implementation for unit of measurement

7.5 Defining relationships between ontological elements

In IEC 62656-1, various kinds of attributes are specified for defining relationships between ontological elements. For example, MDC_P010 (Superclass) is the attribute for describing an is-a relationship between classes. As another example, MDC_P014 (Applicable properties) is the attribute for describing a relationship between a class and a property which becomes applicable to the class. In addition to the predefined attributes, there is a need to define a kind of relationship between ontological elements. A typical use case is a logical grouping of properties which may be widely used for various applications. In IEC 62656-1, such a relationship may be implemented by relations defined in a relation parcel sheet.

Each relation is of either a “predicate” or “functional” type in accordance with the purpose of the relation. Such a type is specified in the attribute MDC_P200 (Relation type) in a relation parcel sheet. If a relation is defined as the functional type, then the keyword “FUNCTION” or its abbreviation “FUNC” shall be specified in the attribute MDC_P200, and its domain and codomain shall be specified in the attributes MDC_P202 (Domain of the function) and MDC_P203 (Codomain of the function), respectively. Otherwise, the keyword “PREDICATION” or its abbreviation “PRED” shall be specified in the attribute MDC_P200, and its domain shall be specified in the attribute MDC_P201 (Domain of the relation).

The attribute MDC_P208 (Type of the domain) is also available, if all ontological elements contained in the domain of a relation are of the same type as the parcel. In this case, the identifier of such a parcel shall be specified as a value of the attribute. Otherwise, the value of the attribute MDC_P208 shall be blank.

NOTE 1 Each relation can specify another relation as (a member of) its domain.

NOTE 2 How to use the relation and the specified ontological elements depends on each application.

The Unified Modeling Language (UML) [9], which is a de facto modelling language in the fields of object oriented programming defined by the Object Management Group (OMG), has a package mechanism that allows objects to be logically grouped, such as classes and use cases, into a package. A package may contain another package as its sub package in order to construct a nested hierarchical structure.

Figure 19 shows an example of a UML package diagram which consists of three packages. The package AYA900 contains the other packages AYA910 and AYA920. The package AYA910 contains the two classes AAA100 and AAA200 and the property AAE001, while the package AYA920 contains the class AAA300 and the two properties AAE002 and AAE003.

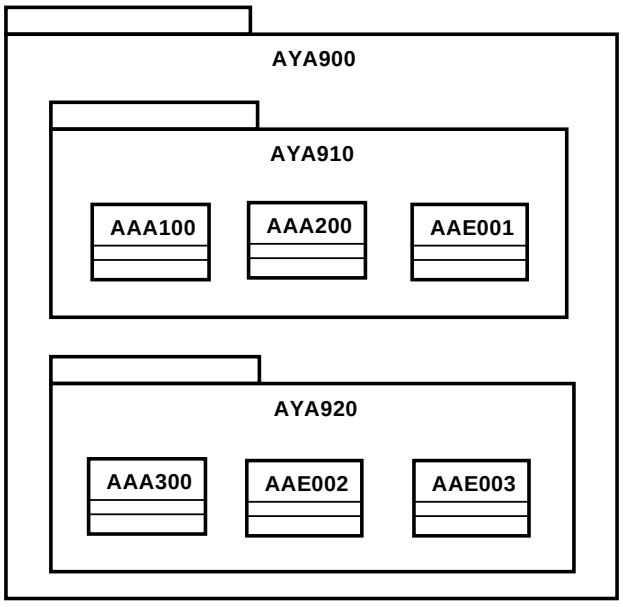


Figure 19 – UML package diagram by relations

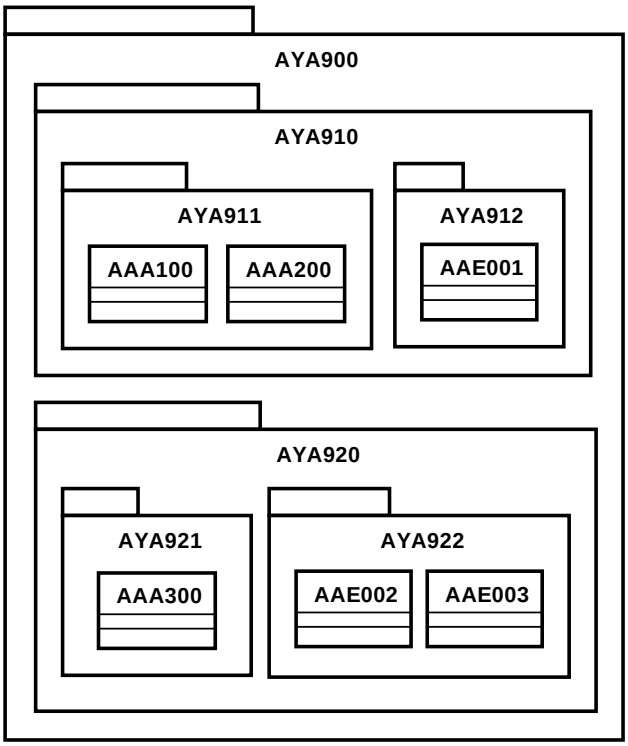
Figure 20 shows a relation parcel sheet as an implementation example for the UML package depicted in Figure 19 by predicate relations. In this example, the identifiers of the classes and properties in the packages AYA910 and AYA920 are simply specified in the attribute MDC_P201 of each package. The packages AYA910 and AYA920 are specified in the attribute MDC_P201 of the package AYA900, to represent the nested relation between packages.

#CLASS_ID:= **MDC_C011**					
#CLASS_NAME.en:= **Relation meta-class**					
#PROPERTY_ID	MDC_P001_13	MDC_P004_1.en	**MDC_P200**	**MDC_P201**	**MDC_P210**
#PROPERTY_ NAME.en	Code	Preferred name	Relation type	Domain of the relation	Role of the relation
	AYA900	Package 1	**PRED**	**{AYA910,AYA920}**	**package**
	AYA910	Package 2	**PRED**	**{AAA100,AAA200,AAE001}**	**package**
	AYA920	Package 3	**PRED**	**{AAA300,AAE002,AAE003}**	**package**

IEC 2307/13

Figure 20 – Parcel implementation of UML packages by predicate relations

Figure 21 shows another example of a UML package diagram which represents a similar data structure to that in Figure 19, but with a different approach. The difference from Figure 19 is that the four additional packages AYA911, AYA912, AYA921 and AYA922 are defined to categorize ontological elements corresponding to the relation for each parcel. For example, the package AYA911 contains the two classes AAA100 and AAA200 which represent the type of the class parcel. The property AAE001 is the type of the property parcel. Therefore the property AAE001 is contained in another relation, AYA912.



IEC 2308/13

Figure 21 – UML package diagram by functions

Figure 22 shows a relation parcel sheet which provides an implementation example of the UML package depicted in Figure 21. In this example, the seven relations AYA900 through AYA922 are defined to contain instances of each parcel. For example, the relation AYA911 actually specifies the two classes AAA100 and AAA200, which are members of the relation AYA910. Each relation specifies the type of the domain in its value of the attribute MDC_P208 for efficient calculation. For example, the relation AYA911 contains only classes in its domain. Therefore MDC_C002, which is the identifier of the class parcel, is specified in its value of the attribute MDC_P208.

A package which is contained within another package is specified by the attribute MDC_P203 of the latter package. For example, the relation AYA900 is specified by the two relations AYA910 and AYA920 via this attribute.

#CLASS_ID:= **MDC_C011**							
#CLASS_NAME.en:= **Relation meta-class**							
#PROPERTY_ID	MDC_ P001_13	MDC_P004_1.en	**MDC_P200**	**MDC_P202**	**MDC_P203**	**MDC_P208**	**MDC_P210**
#PROPERTY_ NAME.en	Code	Preferred name	Relation type	Domain of the function	Codomain of the function	Type of the domain	Role of the relation
	AYA900	Package 1	**FUNC**	**{AYA910,AYA920}**	**UNIVERSE**	**MDC_C011**	**package**
	AYA910	Package 2	**FUNC**	**{AYA911,AYA912}**	**AYA900**	**MDC_C011**	**package**
	AYA920	Package 3	**FUNC**	**{AYA921,AYA922}**	**AYA900**	**MDC_C011**	**package**
	AYA911	classes contained in Package 2	**FUNC**	**{AAA100,AAA200}**	**AYA910**	**MDC_C002**	**package**
	AYA912	properties contained in Package 2	**FUNC**	**{AAE001}**	**AYA910**	**MDC_C003**	**package**
	AYA921	classes contained in Package 3	**FUNC**	**{AAA300}**	**AYA920**	**MDC_C002**	**package**
	AYA922	properties contained in Package 3	**FUNC**	**{AAE002,AAE003}**	**AYA920**	**MDC_C003**	**package**

IEC 2309/13

Figure 22 – Parcel implementation of UML packages by functions

The former approach depicted in Figure 19 is simple, but each package may contain instances of heterogeneous parcels. In other words, there is no information provided in the relation parcel sheet to know what parcel each instance belongs to. Therefore, to reconstruct detailed information of each ontological element in a package, it is necessary to search each parcel. In this case, as the number of ontological elements increases, the processing requirements escalate considerably. In contrast, the latter approach depicted in Figure 21 is complex, but it is easy to determine the package structure and what parcel each ontological element belongs to. From a performance viewpoint of a software application, IEC 62656-3 introduces the latter implementation for describing the package structure of the common information model, or CIM for short, defined in IEC 61970-301.

Note that how to implement such a structure is not limited to the examples shown in Figure 20 and Figure 22. If another use of the relation parcel is possible and needed for a specific application, such a use should be improvised in the application.

8 Advanced concepts

8.1 Implementation of condition

A condition property is a property which may affect a value decision of another property (see IEC 61360-1).

A typical use case of such a property is to define an external environment, such as temperature and humidity, for measurement values. Another typical use case is to provide a way to explicitly configure the number of slots within a programmable logic controller at the Domain Library layer (see 8.2 to 8.4).

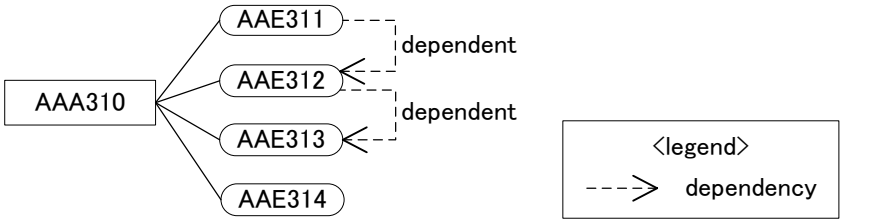
The attribute MDC_P020 (Property data element type) of the property parcel is an attribute for specifying the dependency of a property, which means i) whether that property is a condition property for another property, and ii) whether that property is dependent on another property. Table 1 shows what value shall be selected for each property in each combination of the above cases, i) and ii).

Table 1 – Property data element type for condition

condition for another property	depends on another property	type to be assigned
		NON_DEPENDENT_P_DET
✓		CONDITION_DET
	✓	DEPENDENT_P_DET
✓	✓	DEPENDENT_C_DET

In the case that a property depends on one or more other properties, DEPENDENT_P_DET or DEPENDENT_C_DET is assigned to the former property. The identifiers of the latter properties shall be specified in the attribute MDC_P028 (Condition) of the property parcel.

Figure 23 shows an example of a data dictionary including condition properties in which the class AAA310 has four properties. The property AAE311 depends on the property AAE312, and the property AAE312 also depends on the property AAE313.



IEC 2310/13

Figure 23 – Example of condition

Figure 24 shows an example of a property parcel sheet to describe the properties depicted in Figure 23. The property AAE311 depends on the property AAE312, therefore, the property AAE311 is defined as DEPENDENT_P_DET and its condition property is listed in the attribute MDC_P028. The property AAE312 is the condition property for the property AAE311 and it further depends on the property AAE313. Therefore, the property AAE312 is defined as DEPENDENT_C_DET. Next, the property AAE313 has no condition property; therefore the property is defined as CONDITION_DET. Finally, the property AAE314 is not a condition property for the others and does not depend on the others, therefore the property AAE314 is defined as NON_DEPENDENT_P_DET.

#CLASS_ID:= **MDC_C003**					
#CLASS_NAME.en:= **Property meta-class**					
#PROPERTY_ID	MDC_P001_6	MDC_P020	MDC_P022	MDC_P023_1	MDC_P028
#PROPERTY_ NAME.en	Code	Property data element type	Data type	Unit in text	Condition
	AAE311	**DEPENDENT_P_DET**	REAL_MEASURE	M	**{AAE312}**
	AAE312	**DEPENDENT_C_DET**	REAL_MEASURE	°C	**{AAE313}**
	AAE313	**CONDITION_DET**	REAL_MEASURE	1	
	AAE314	**NON_DEPENDENT_P_DET**	INT		

dependent
dependent

IEC 2311/13

Figure 24 – Parcel implementation for condition

8.2 Implementation of cardinality

Cardinality is for restricting the occurrence number of a property value, which includes components, parts and materials; namely, it defines the minimum and maximum number of the collection. A feature of this property is that heterogeneous elements may appear in the collection.

A typical example is a bicycle. It has exactly two tyres, one on the front and the other on the rear of a body. As another example, a computer having three embedded USB ports can support different devices, such as an optical mouse, a USB flash drive and a webcam, connected at the same time.

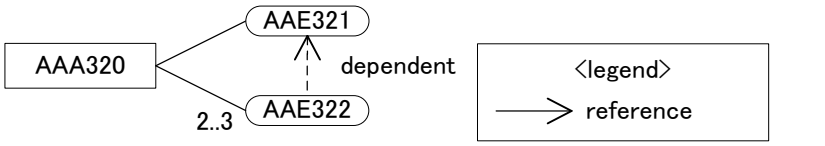
LIST is the most recommended data type for describing cardinality, because in many cases, the order of the items is recognized implicitly or explicitly. In the case that the order of the items is not actually needed, BAG may be also used.

The minimum and maximum numbers shall be specified as the arguments of those data types, e.g. if those numbers for a property are 2 and 3, respectively, the property will be defined as "LIST(2,3)".

By using a condition property (interface property) for such a property, which explicitly indicates the number of slots for the latter, it will be possible to control the interface for the latter. If such a property is not required, this property can be omitted.

NOTE If the parameters "minimum" and "maximum" are omitted, they are implicitly recognized as "0" and "?", the latter means the unlimited number.

Figure 25 shows an example of how to implement cardinality including an interface property. The property AAE322 of the class AAA320 is expected to contain two or three values. In this example, the property AAE321 is also assigned to the class AAA320, to indicate the actual number of slots for the property AAE322.



IEC 2312/13

Figure 25 – Example of cardinality

Figure 26 shows an example of a property parcel sheet for describing the two properties depicted in Figure 25. For the property AAE322, the minimum and maximum occurrence number are specified as arguments of LIST in the value of the attribute MDC_P022 such as "LIST(2,3) OF REAL_MEASURE_TYPE". The property AAE321 is defined as INT_TYPE for specifying the occurrence number of values of the property AAE322 at Domain Library layer. The relationship between those two properties is defined in the value of the attribute MDC_P028 for the property AAE322, which depends on the property AAE321.

#CLASS_ID:= **MDC_C003**					
#CLASS_NAME.en:= **Property meta-class**					
#PROPERTY_ID	MDC_P001_6	MDC_P020	MDC_P022	MDC_P023_1	MDC_P028
#PROPERTY_ NAME.en	Code	Property data element type	Data type	Unit in text	Condition
	AAE321	**CONDITION_DET**	**INT_TYPE**		
	AAE322	**DEPENDENT_P_DET**	**LIST(2,3) OF REAL_MEASURE_TYPE**	**m**	**{AAE321}**

IEC 2313/13

Figure 26 – Parcel implementation for cardinality

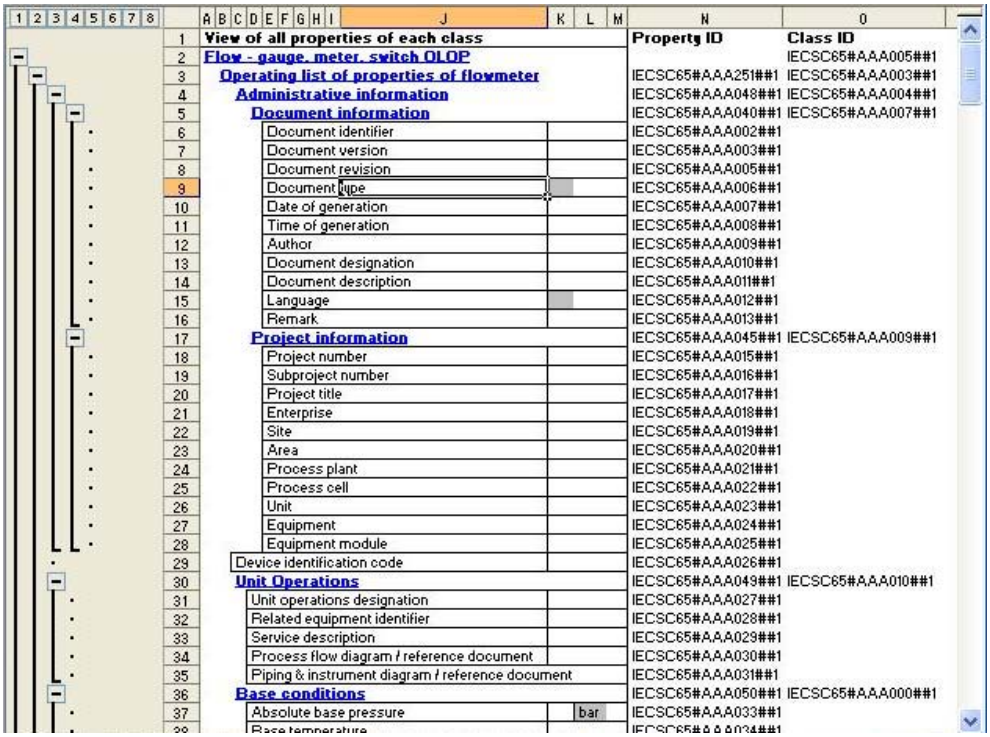
8.3 Implementation of blocks and lists of properties (LOPs)

The block and LOP (list of properties) are parts of a mechanism originally defined in IEC 61987-10 that allow free grouping properties. In the ISO 13584-42/IEC 61360-2 common dictionary model, each block and LOP is implemented as a feature class, to which its grouped properties are applicable. Each block and LOP may contain one or more blocks by either

CLASS_REFERENCE_TYPE or CLASS_INSTANCE_TYPE property, in accordance with the "has-a" relationship explained in 6.4, i.e. the nested structure of blocks and LOPs is allowed.

Cardinality, introduced in 8.2, is also used to indicate the repetition of blocks of properties or LOPs. This means the value of a cardinality property in the transaction data defines how many times a block or a LOP should be repeated. In this case, "LIST OF CLASS_REFERENCE_TYPE" or "LIST OF CLASS_INSTANCE_TYPE" shall be used for properties to define the minimum and maximum number of blocks.

Figure 27 shows an example of a typical presentation view for an LOP and its nested blocks in a spreadsheet application. In this example, line 2 shows the LOP, lines 3 through 5, 17, 30 and 36 show blocks and the other lines show the properties. Columns A through D show the level of each block. For example, the LOP "Flow – gauge, meter, switch OLOP", described in line 2, is the root of the structure, therefore it is described in the first column "A". The block "Operating list of properties of flowmeter", described in line 3, is contained in the LOP. Therefore the block is described in the second column "B". In this example, the hierarchy, which consists of LOPs and their blocks, is displayed in such a hierarchical manner.



IEC 2314/13

Figure 27 – View example of a LOP and nested blocks

To avoid redundant definitions of identical items, properties which may be shared among several LOPs or blocks should be defined only once and referenced from the other places in which they are used. If there is no general branch specified in the classification tree where such commonly used properties may reside, they should be imported from a LOP or block on another branch by means of the "case_of" mechanism explained in 6.3.

Figure 28 shows an example of a typical use case of a block, including a multiple block and a nested block. There are two branches in the example, one is a branch consisting of only the class AAA330 shown in the left hand side of the figure, and the other is a branch consisting of the three block classes BBA200, BBA300 and BBA400, shown in the right hand side of the figure. The class AAA330 has the three properties AAE331 through AAE333. The property AAE331 is defined as CLASS_REFERENCE_TYPE specifying the block class BBA200, while the property AAE332 is defined as LIST(2,3) OF CLASS_REFERENCE_TYPE specifying the block class BBA400. The latter property also specifies the property AAE333 as its control

property. The block class BBA200 has the property BBE201, which is defined as CLASS_REFERENCE_TYPE specifying the block class BBA300 (i.e. nested block).

NOTE The classes AAA330, BBA200, BBA300 and BBA400 are parts of the overall hierarchy of classes. For clarity the superclass is omitted and not shown in the figure.

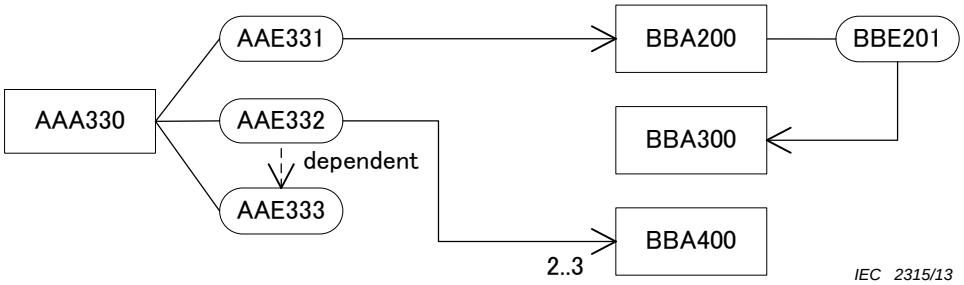


Figure 28 – Example of use case of blocks

Figure 29 shows an example of the composition view of AAA330 at Domain Library layer representing the structure within Figure 28. It illustrates how to use the control property AAE333 at Domain Library layer. Each line depicted by bold letters is a block, while each line depicted in plain letters is a normal property. In this figure, two slots are provided for the property AAE332 in accordance with the value of its control property AAE333. Note that the properties described in small letters, i.e. aaa through ddd, are properties of the blocks, but those properties are omitted from Figure 28, for simplicity.

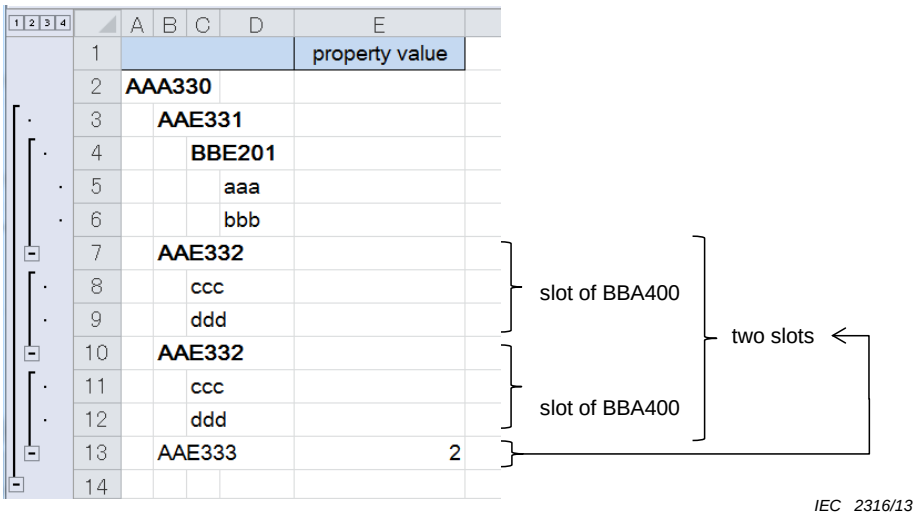


Figure 29 – Example of a composition view of an LOP

Figure 30 shows an example of conjunctive parcels to describe the data dictionary shown in Figure 28. In the class parcel sheet at the top of the figure, the class and each block class are described, while in the property parcel sheet at the bottom of the figure, each property is described.

#CLASS_ID:= **MDC_C002**			
#CLASS_NAME.en:= **Class meta-class**			
#PROPERTY_ID	MDC_P001_5	MDC_P011	MDC_P014
#PROPERTY_ NAME.en	Code	Class type	Applicable properties
	AAA330	ITEM_CLASS	**{AAE331,AAE332,AAE333}**
	BBA200	ITEM_CLASS	**{BBE201}**
	BBA300	ITEM_CLASS	
	BBA400	ITEM_CLASS	

applicable

#CLASS_ID:= **MDC_C003**				
#CLASS_NAME.en:= **Property meta-class**				
#PROPERTY_ID	MDC_P001_6	MDC_P020	MDC_P022	MDC_P028
#PROPERTY_ NAME.en	Code	Property data element type	Data type	Condition
	BBE201	**NON_DEPENDENT_P_DET**	**CLASS_REFERENCE_TYPE(BBA300)**	
	AAE331	**NON_DEPENDENT_P_DET**	**CLASS_REFERENCE_TYPE(BBA200)**	
	AAE332	**DEPENDENT_P_DET**	**LIST(2,3) OF CLASS_REFERENCE_TYPE(BBA400)**	**{AAE333}**
	AAE333	**CONDITION_DET**	**INT_TYPE**	

IEC 2317/13

Figure 30 – Parcel implementation for blocks

8.4 Implementation of polymorphism

Polymorphism (see IEC 61987-10) is a mechanism intended to give choices among blocks at Domain Library layer. In IEC 62656-1, polymorphism may be simply expressed by means of either ENUM_REFERENCE_TYPE or ENUM_INSTANCE_TYPE. Such a data type specifies an enumeration which represents the identifiers of classes to be selected as polymorphic choices. A list of those identifiers is specified as a value of the attribute MDC_P025_1 (Preferred letter symbol in text) of the term parcel.

For explicitly specifying a polymorphic choice at Domain Library layer, a condition property can be used as a control property (see 8.1). This condition property is defined as ENUM_STRING_TYPE, which specifies the same enumeration of the polymorphic property.

NOTE Traditionally, in the IEC CDD, properties are directly assigned to a class, each of which is of CLASS_REFERENCE_TYPE or of CLASS_INSTANCE_TYPE and points to a class to be selected as a polymorphic choice. In this case, for grouping those properties and for giving a function to select one of the choices at Domain Library layer, a property of enumeration type (non_quantitative type) as the condition for controlling the selection is also assigned in the same class and those properties depend on this control property.

Each polymorphic property may allow multiple uses of the associated block in accordance with the cardinality mechanism explained in 8.2. In this case, this property should be defined as LIST OF ENUM_REFERENCE_TYPE or LIST OF ENUM_INSTANCE_TYPE. If a control property is used to enumerate the possible polymorphic choices, this property should be defined as LIST OF ENUM_STRING_TYPE.

Figure 31 is an example of a use case of polymorphism. The property AAE342 acts as pointer to the available polymorphism choices. The polymorphic choices for the property AAE342 are provided by the enumeration AFA001 specifying the two terms AQA101 and AQA102 which in turn refer to the block classes BBA500 and BBA600 as the polymorphic choices. Through those relationships, the property AAE342 offers the choices of BBA500 and BBA600 as its values. In this example, the property AAE341 is additionally provided for allowing explicitly specifying a choice at Domain Library layer, therefore the property AAE341 is a condition

property for the property AAE342 and is defined as ENUM_STRING_TYPE using the same enumeration as used by the property AAE342.

NOTE1 The classes AAA340, BBA500 and BBA600 are parts of the overall hierarchy of classes. For clarity the superclass is omitted and not shown in the figure.

NOTE2 Traditionally, in the above example, two properties are assigned to AAA340, which are of CLASS_REFERENCE_TYPE and point to BBA500 and BBA600, respectively, as polymorphic choices. In this case, those properties depend on AAE341 as their control property.

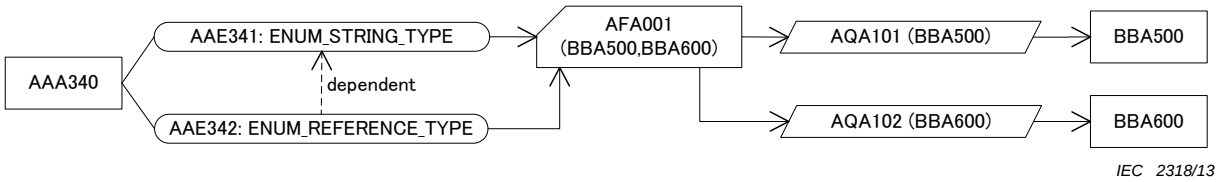


Figure 31 – Example of a use case of polymorphism

Figure 32 shows an example of the composition view of AAA340 at Domain Library layer representing the structure within Figure 31. Each line using bold letters stands for a block, while each line using plain letters represents a property. In this figure, the property AAE341 specifies “BBA500”, so the block BBA500 is selected as the actual choice of the property AAE342. Note that the properties described in small letters, i.e. mmm through nnn, are properties of the block BBA500. Properties mmm and nnn are omitted from Figure 31, for simplicity.

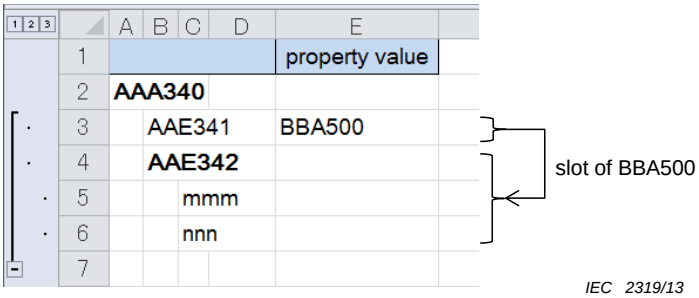


Figure 32 – Example of composition view for polymorphism

Figure 33 shows sheets of conjunctive parcels for the data dictionary shown in Figure 31. The properties AAE341 and AAE342 are defined as ENUM_STRING_TYPE and ENUM_REFERENCE_TYPE, respectively, both of which specify the identifier of the enumeration AFA001 between these parentheses. The blocks BBA500 and BBA600, which are the polymorphic choices for those properties, are specified in the attributes MDC_P025_1 (Preferred letter symbol in text) of the terms AQA101 and AQA102, respectively.

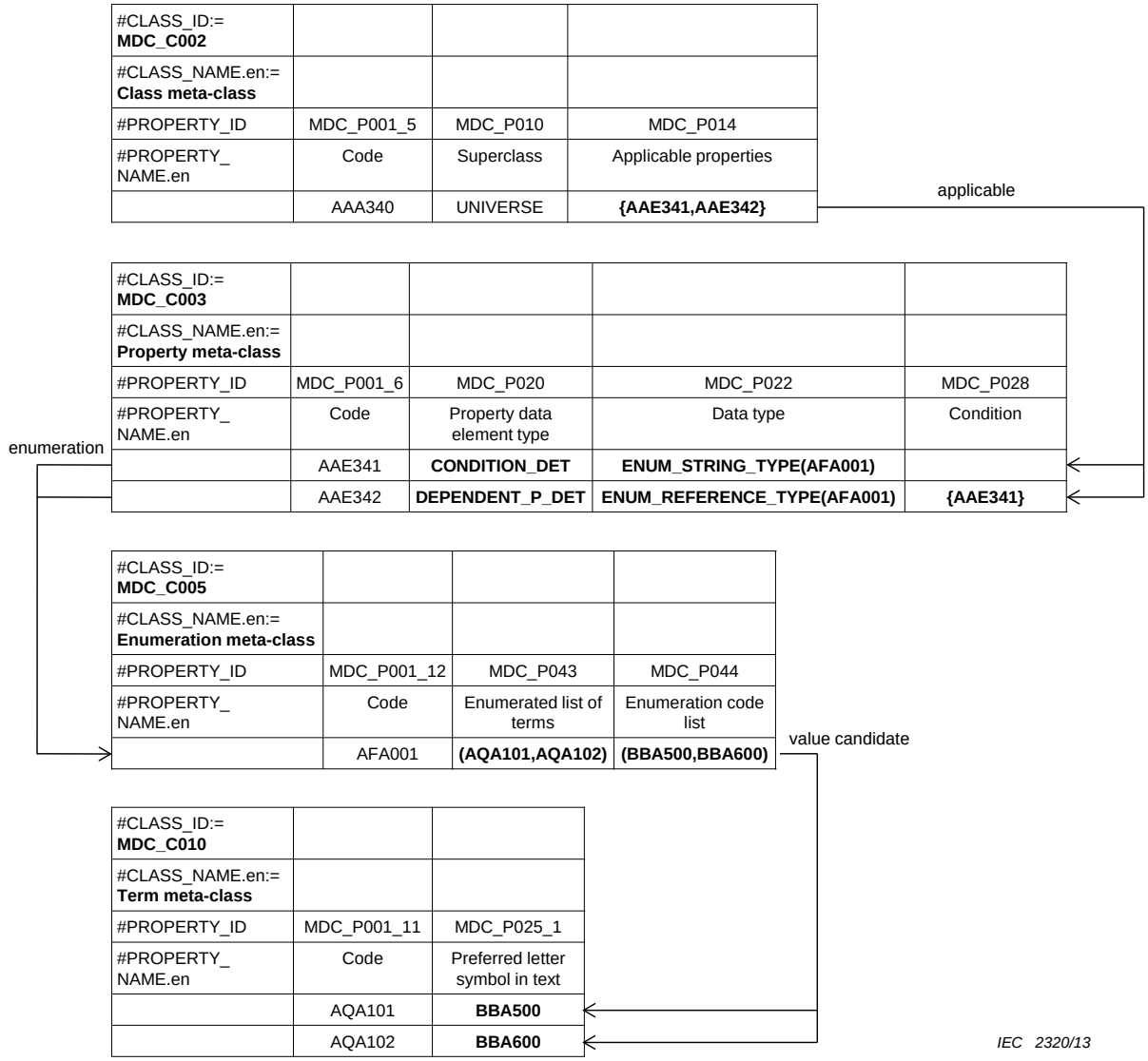


Figure 33 – Parcel implementation for polymorphism

Figure 34 is an extension of the example shown in Figure 31, allowing multiple choices for polymorphism. In this example, the properties AAE351 and AAE352 are defined as the LIST type, identifying the multiple choices available within these properties. The property AAE353 is also added as a control property for the properties AAE351 and AAE352, to indicate the repetition of those properties.

NOTE The classes AAA350, BBA500 and BBA600 are parts of the overall hierarchy of classes. For clarity the superclass is omitted and not shown in the figure.

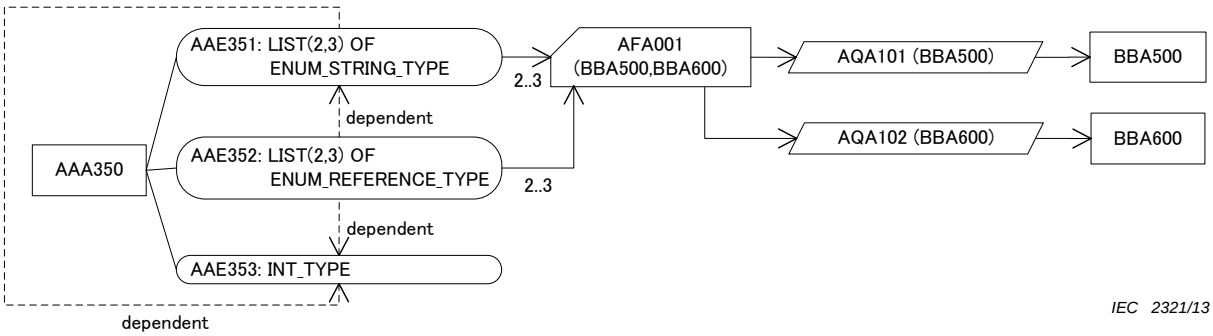


Figure 34 – Example of a use case of polymorphism with multiple choices

Figure 35 shows an example of the composition view of AAA350 at Domain Library layer to represent the structure shown in Figure 34. It also illustrates how to utilise the control property AAE353 at Domain Library layer. Each line depicted by bold letters is a block, while each line depicted by plain letters is a normal property. In this figure, two slots are provided for the property AAE351 and the polymorphic property AAE352 in accordance with the value of its control property AAE353. The first line of the property AAE351 specifies “BBA500”, so the block BBA500 is selected as the first slot of the property AAE352. The second line of the property AAE351 specifies “BBA600”, so the block BBA600 is selected as the second slot of the property AAE352. Note that the properties depicted by small letters, i.e. mmm through qqq, are properties for the blocks, but those properties are not shown in Figure 34, for simplicity.

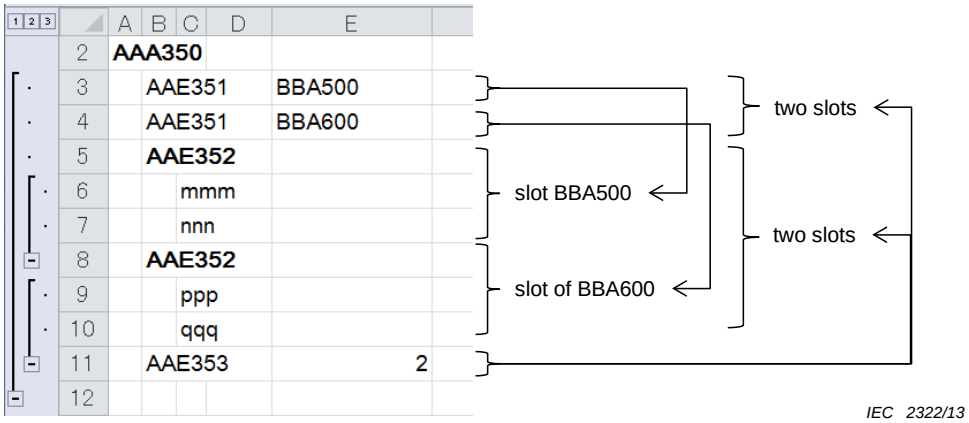


Figure 35 – Example of composition view for polymorphism with multiple choices

Figure 36 shows sheets of conjunctive parcels for the data dictionary shown in Figure 34. For multiple choices, the properties AAE351 and AAE352 are defined as LIST(2,3) OF ENUM_STRING_TYPE and LIST(2,3) OF ENUM_REFERENCE_TYPE, respectively, both of which specify the identifier of the enumeration AFA001 between these parentheses. The property AAE353 is defined as INT_TYPE, and is specified as a condition property using the two properties AAE351 and AAE352. The property AAE351 is the condition property for the property AAE352, and depends on the property AAE353. Therefore the property AAE351 is defined as DEPENDENT_C_DET in the manner explained in 8.1.

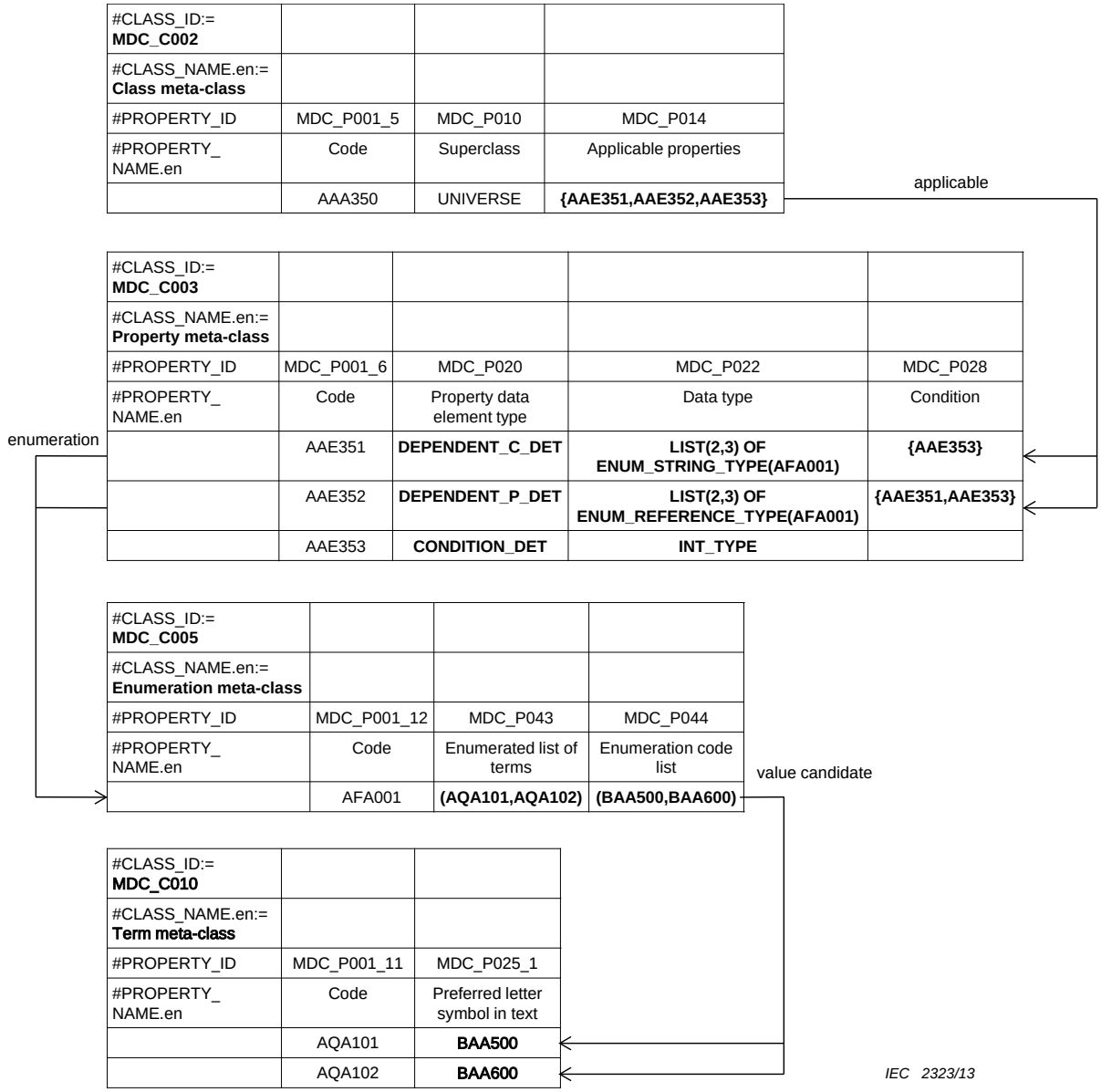


Figure 36 – Parcel implementation for polymorphism with multiple choices

8.5 Alternate IDs

A class and a property may have an alternate ID. The typical use case of alternate ID is to map between two similar or identical ontological elements from different data dictionaries, which are modelled as different classes or properties for historical or organizational reasons.

For a property, if there are properties to be specified as alternate property to it, the identifiers of the alternate properties shall be specified as a value of an attribute MDC_P101 (Alternate ID). For a class, if there is a class to be specified as alternate class to the former class, the identifier of the alternate class shall be specified as a value of an attribute MDC_P102 (Alternate class ID).

Figure 37 shows an example of a property parcel sheet, in which three properties and their alternate ID(s) are defined. The property AAE361 specifies the property BBE001 as its alternate ID, which is defined in another data dictionary. The property AAE362 also specifies a set of the identifiers of the properties BBE002 and CCE001, which are defined in other data dictionaries. Finally, the property AAE363 has no specific property as its alternate ID, therefore the field of the attribute MDC_P101 is kept blank.

#CLASS_ID:=**MDC_C003**		
#CLASS_NAME.en:=**Property meta-class**		
#PROPERTY_ID	MDC_P001_6	MDC_P101_6
#PROPERTY_NAME.en	Code	Alternate ID
	AAE361	**BBE001**
	AAE362	**{BBE002,CCE001}**
	AAE363	

IEC 2324/13

Figure 37 – Parcel implementation for alternate ID

9 Data file representation for storage and exchange

9.1 CSV format for representation of data parcels

IEC 62656-1 does not specify any file format for data storage and exchange, but the CSV (Comma-Separated Values see [10]) format is recommended because the CSV format is similar to the structure of a parcel sheet and is acceptable for various applications with no or minimal modification. Subclauses 9.2 to 9.5 provide valuable information for reading or writing data in a CSV file correctly.

9.2 Cell delimiter

As shown in the name CSV, a comma character is generally used as a cell delimiter. However, several applications use another character as the delimiter, when a comma character is used for another purpose, e.g. as a decimal point in some European languages. Thus, any parcelling tool should recognize what character is used as a cell delimiter to determine the value of each cell correctly.

If a cell value contains a cell delimiter, such a value shall be enclosed between text qualifiers.

EXAMPLE If a semi-colon character is a cell delimiter and a cell value is “yes;no;unknown”, this value is supposed to be enclosed between text qualifiers. In this case, a comma character is not a cell delimiter, therefore any value including “,”, for example “yes, no or unknown”, does not need to be enclosed between text qualifiers.

9.3 Line feed character

Not all commercial or non-commercial tools allow the use of a line feed character in a cell value. Thus, except for the case in which all the applications involved in data exchange can correctly handle data containing the line feed character, the line feed character shall not be used in any cell value. When the use of the line feed character is allowed, any cell value containing the line feed character shall be enclosed between text qualifiers.

NOTE The number of text qualifiers in a line are always even in a CSV file. Thus, if the number of text qualifiers in a line is odd, the line is concatenated with the line feed character and next line in order.

Figure 38 shows an example of CSV data containing the line feed character in a cell value.

The first line is valid CSV data, which is represented by one row and two columns in spreadsheet applications. The first element contains the line feed character within its value; therefore the value is enclosed between text qualifiers.

The second line is invalid CSV data, because the value of the first element is not enclosed between the text qualifiers. As a result, the data is incorrectly displayed as two rows and two columns in spreadsheet applications.

The third line is also invalid CSV data, because the value of the first element starts with the text qualifier, while the value does not end with the text qualifier. As a result, the data is incorrectly displayed as one row and one column on spreadsheet.

CSV data	View on a text editor	View on spreadsheet	Correctness
“sentence 1.<LF>sentence 2.”,sentence 3.	“sentence 1.<LF> sentence 2. “,sentence 3.	sentence 1. sentence 2. | sentence 3.	valid
sentence 1.<LF>sentence 2.,sentence 3.	sentence 1.<LF> sentence 2.sentence 3.	sentence 1. | sentence 2. | sentence 3.	invalid
“sentence 1.<LF>sentence 2.,sentence 3.	“sentence 1.<LF> sentence 2.sentence 3.	sentence 1. sentence 2., sentence 3.	invalid

IEC 2325/13

Figure 38 – Example of how to escape the line feed characters

9.4 Space character

An unnecessary space character which is inserted before or after a value for better readability or due to mistyping shall be ignored. If such a space character is due to remain, such a value which contains the space character(s) shall be enclosed with the text qualifiers. This rule should be also applied to every member value in an aggregation.

EXAMPLE 1 When applicable properties are described such as “{P001, P002, P003}”, the space characters before P002 and P003 are ignored.

EXAMPLE 2 When a “SET OF STRING_TYPE” property has a value “{abc, efg,”” xyz ”” }”, the space character before “efg” is ignored, but the space characters before and after “xyz” remain.

9.5 Character encoding

A data parcel may contain one or more multi-byte characters. In this case, a character encoding for a CSV file shall support such characters in order to avoid any problem in data storage and exchange. Therefore, UTF-8 or UTF-16 is recommended.

10 Conformance to implementation for the IEC CDD

IEC 62656-1 defines the POM (parcellized ontology model) conformance classes. The conformance classes are classified with the level number of the top layer of the parcels used in an exchange, according to the MoF (Meta-object facility [11]) layer scheme, and further sub-classified according to the layers that are included in the exchange. The classification can also specify the presence, or not, of extension(s) in modeling capabilities beyond IEC 61360. In addition, IEC 62656-1 allows further specification of an ontology model by name, which shall be used for the processing of data parcels. The name shall be specified in the parentheses added after the conformance class ID or CCID for short.

This part of IEC 62656 defines two additional conformance classes, 2(CDD) and 2X(CDD) as the specialization of the conformance classes 2 and 2X defined in IEC 62656-1, for uploading and downloading domain ontologies or data dictionaries to and from the IEC CDD. Note that if all attributes for the IEC CDD are provided in other conformance classes, which include the layers M3-M2 and M4-M3 as well as the M2-M1 layer, e.g. in the conformance classes 3B, 3BX, 4C and 4CX, such conformance classes also satisfy the requirement conformance for the IEC CDD.

Consequently, Table 2 lists all the conformance classes defined in IEC 62656-1 and the additional conformance classes, 2(CDD) and 2X(CDD).

Table 2 – POM conformance classes

CCID	Definition	MoF layers
1	Data parcel just for DL (Domain library)	M1-M0
1X	Data parcel only for DL with local extension of properties and possibly their instance values	M1-M0
2	Data parcel just for DO (Domain ontology)	M2-M1
2X	Data parcel just for DO with local extension of properties and possibly their instance values	M2-M1
2(CDD)	**Data parcel for DO to be registered into the IEC CDD**	**M2-M1**
2X(CDD)	**Data parcel for DO to be registered into the IEC CDD with local extension of properties**	**M2-M1**
2A	Data parcels for all layers below comprising DO and DL	M2-M1, M1-M0
2AX	Data parcels for all layers below comprising DO and DL with local extension of properties and possibly their instance values	M2-M1, M1-M0
3	Data parcel just for MO (Meta ontology)	M3-M2
3X	Data parcel only for MO with local extension of properties and possibly their instance values	M3-M2
3A	Data parcel with all layers below, comprising MO, DO, and DL	M3-M2, M2-M1, M1-M0
3AX	Data parcel with all layers below, comprising MO, DO, and DL with local extension of properties and possibly their instance values	M3-M2, M2-M1, M1-M0
3B	Data parcels with the layer below comprising MO and DO	M3-M2, M2-M1
3BX	Data parcels with the layer below comprising MO and DO with local extension of properties and possibly their instance values	M3-M2, M2-M1
4	Data parcel just for AO (Axiomatic ontology)	M4-M3
4X	Data parcel just for AO with local extension of properties and possibly their instance values	M4-M3
4A	Data parcels with all layers below, comprising AO, MO, DO, and DL	M4-M3, M3-M2, M2-M1, M1-M0
4AX	Data parcels with all layers below, comprising AO, MO, DO, and DL with local extension of properties and possibly their values	M4-M3, M3-M2, M2-M1, M1-M0
4B	Data parcels with the layer below comprising AO and MO	M4-M3, M3-M2
4BX	Data parcels with the layer below comprising AO and MO with local extension of properties and possibly their instance values	M4-M3, M3-M2
4C	Data parcels with all layers except DL comprising AO, MO and DO	M4-M3, M3-M2, M2-M1
4CX	Data parcels with all layers except DL comprising AO, MO and DO with local extension of properties and possibly their instance values	M4-M3, M3-M2, M2-M1

Annex A
(normative)

Information object registration – Document identification

In order to provide for unambiguous identification of an information object in an open system, the object identifier

{ iec standard 62656 part (2) version (1) }

is assigned to this part of IEC 62656. The meaning of this value is defined in ISO/IEC 8824-1 [12].

Annex B
(informative)

Examples of pattern constraints for attributes

Table B.1 gives examples of typical pattern constraints which may be applied to some attributes defined in IEC 62656-1. Each pattern constraint may be specified in the line of the preserved instruction “#PATTERN” in a parcel sheet, for syntactically restricting or checking values in the data section of the sheet. Attributes which do not appear in the following table mean that there are no specific rules to be applied to those attributes.

Each italic character string in the third column of the table is a pattern which may be commonly used by plural attributes. Its actual pattern is described in the footnote of the table.

Table B.1 – Examples of pattern constraints for attributes *(1 of 3)*

Code	Preferred name	Pattern constraint
MDC_P001_X	Code	id_pattern [a]
MDC_P002_1	Version number	[0-9]{1,10}
MDC_P002_2	Revision number	[0-9]{1,3}
MDC_P002_3	Content revision	[0-9]{1,3}
MDC_P003_1	Date of original definition	date_pattern [b]
MDC_P003_2	Date of current version	date_pattern
MDC_P003_3	Date of current revision	date_pattern
MDC_P004_2	Synonymous name	\{\(element_pattern,lang_pattern\)(,\(element_pattern,lang_pattern\))*\} [c d]
MDC_P004_4	Name icon	id_pattern
MDC_P008_1	Simplified drawing	id_pattern
MDC_P008_2	Graphics	id_pattern
MDC_P010	Superclass	id_pattern
MDC_P011	Class type	ITEM_CLASS|COMPONENT_CLASS|MATERIAL_CLASS| FEATURE_CLASS|ITEM_CLASS_CASE_OF| COMPONENT_CLASS_CASE_OF|MATERIAL_CLASS_CASE_OF| FEATURE_CLASS_CASE_OF
MDC_P012	Supplier	[a-zA-Z0-9_/]+
MDC_P013	Is case of	id_set_pattern [e]
MDC_P014	Applicable properties	id_set_pattern
MDC_P015	Applicable types	id_set_pattern
MDC_P016	Sub-class selection properties	id_set_pattern
MDC_P017	Class value assignment	\{\(id_pattern,element_pattern\)(,\(id_pattern,element_pattern\))*\}

Table B.1 *(2 of 3)*

Code	Preferred name	Pattern constraint
MDC_P020	Property data element type	NON_DEPENDENT_P_DET|DEPENDENT_P_DET|CONDITION_DET|DEPENDENT_C_DET
MDC_P021	Definition class	id_pattern
MDC_P028	Condition	id_set_pattern
MDC_P040	DET classification	[A-Z][0-9]{2}
MDC_P041	Code for unit	id_pattern
MDC_P042	Codes for alternative units	id_set_pattern
MDC_P043	Enumerated list of terms	id_list_pattern [f]
MDC_P044	Enumeration code list	\(element_pattern(,element_pattern)*\)
MDC_P051_11	E-mail	([a-zA-Z0-9][a-zA-Z0-9_\.+\\-]*)@(([a-zA-Z0-9][a-zA-Z0-9_\\-]*\.)+[a-zA-Z]{2,6})
MDC_P062	Remote location	url_pattern [g]
MDC_P065_3	Main content encoding	7bit|8bit|binary|quoted-printable|base64
MDC_P065_9	Main content remote access	url_pattern
MDC_P068	Property constraint	\{property_constraint_pattern(,property_constraint_pattern)*\} [h]
MDC_P072	LIIM status	WD|CD|DIS|FDIS|IS|TS|PAS|ITA
MDC_P074	LIIM date	[0-9]{1,4}
MDC_P075	LIIM application	[1-6]E?
MDC_P080	Global language	lang_pattern
MDC_P081	Source language	lang_pattern
MDC_P090	Imported properties	id_set_pattern
MDC_P091	Imported types	id_set_pattern
MDC_P093	Imported documents	id_set_pattern
MDC_P094	Applicable documents	id_set_pattern

Table B.1 *(3 of 3)*

Code	Preferred name	Pattern constraint
MDC_P096	Property classification	\{\(id_pattern,[1-9]*[0-9],element_pattern\)(,\(id_pattern,[1-9]*[0-9],element_pattern\))*\}
MDC_P097	Requisity of properties	\{\(id_pattern,requisity_pattern\)(,\(id_pattern,requisity_pattern\))*\} [i]
MDC_P110	Super property	id_pattern
MDC_P200	Relation type	FUNCTION|FUNC|PREDICATION|PRED
MDC_P201	Domain of the relation	id_list_pattern
MDC_P202	Domain of the function	id_list_pattern
MDC_P203	Codomain of the function	id_pattern
MDC_P205	Language for formula interpretation	lang_pattern
MDC_P207	Trigger event	\(id_pattern(,id_pattern)+\)

[a] id_pattern:= ([a-zA-Z0-9_/]+#)?[a-zA-Z0-9]+(##[0-9]{1,10})?(###.*)?

[b] date_pattern:= ([1-9]{0,3}[0-9]\-((0[1-9])|(1[012]))\-((0[1-9])|([12][0-9])|(3[01])))

[c] element_pattern:= ([^\(\)\{\},"]+|"(([^"])|(""))*")

[d] lang_pattern:= [a-z]{2,3}

[e] id_set_pattern:= \{id_pattern(,id_pattern)*\}

[f] id_list_pattern:= \(id_pattern(,id_pattern)*\)

[g] url_pattern:= ((https?|ftp)(:\/\/[-_.!~*\'()a-zA-Z0-9;\/?:\@&=+\$,%#]+))

[h] property_constraint_pattern:= ((SUBCLASS_CONSTRAINT\(id_set_pattern\))
|(ENTITY_SUBTYPE_CONSTRAINT\(element_pattern(,element_pattern)*\))
|(ENUMERATION_CONSTRAINT\(element_pattern(,element_pattern)*\))
|(RANGE_CONSTRAINT\(real_pattern,(real_pattern|\?)(,(true)|(false)){2}\))
|(STRING_SIZE_CONSTRAINT\((0|([1-9][0-9]*)),(0|\?|([1-9][0-9]*))\))
|(STRING_PATTERN_CONSTRAINT\(element_pattern\))
|(CARDINALITY_CONSTRAINT\((0|[1-9][0-9]*),(0|\?|[1-9][0-9]*)\))) [j]

[i] requisity_pattern:= (KEY|MANDATORY|MAND|NOT NULL|OPTIONAL|OPT|DER)

[j] real_pattern:= (-?[1-9][0-9]*(\.[0-9]+)?|-0\.[0-9]*[1-9]|0\.[0-9]+|0)

Annex C
(informative)

Examples for attribute values

Table C.1 gives a pair of examples of valid and invalid values for some of the attributes defined in IEC 62656-1.

Table C.1 – Examples of attribute values *(1 of 3)*

Expected value	Attribute	Valid value example	Cases of invalid values
ICID	MDC_P001_X (Code) MDC_P004_4 (Name icon) MDC_P008_1 (Simplified drawing) MDC_P008_2 (Graphics) MDC_P010 (Superclass) MDC_P021 (Definition class) MDC_P041 (Code for unit) MDC_P110 (Super property) MDC_P203 (Codomain of the function)	– 0112/2///62656_2#BBB001##1 – 0112/2///62656_2#BBB001##1###comments	a) ID format violation – 0112/2///62656_2#BBB001#1 b) either RAI or VI is missing – BBB001##1 – 0112/2///62656_2#BBB001 – BBB001
a set of ICID	MDC_P013 (Is case of) MDC_P014 (Applicable properties) MDC_P015 (Applicable types) MDC_P016 (Sub-class selection properties) MDC_P028 (Condition) MDC_P042 (Codes for alternative units) MDC_P090 (Imported properties) MDC_P091 (Imported types) MDC_P093 (Imported documents) MDC_P094 (Applicable documents) MDC_P207 (Trigger event)	– {0112/2///62656_2#BBB001##1, 0112/2///62656_2#BBB002##1} – {0112/2///62656_2#BBB001##1###comment, 0112/2///62656_2#BBB002##1###comment}	a) ID format violation b) identifiers are enclosed between not curly brackets but parentheses c) either RAI or VI is missing at an element of a set of identifiers – {BBB001_X##1, BBB002##1} – {0112/2///62656_2#BBB001_X, 0112/2///62656_2#BBB002} – {BBB001_X, BBB002}

Table C.1 *(2 of 3)*

Expected value	Attribute	Valid value example	Cases of invalid values
a list of ICID	MDC_P043 (Enumerated list of terms) MDC_P201 (Domain of the relation) MDC_P202 (Domain of the function)	– (0112/2///62656_2#BBB001##1, 0112/2///62656_2#BBB002##1) – (0112/2///62656_2#BBB001##1###comment, 0112/2///62656_2#BBB002##1###comment)	a) ID format violation b) identifiers are enclosed between not curly brackets but parentheses c) either RAI or VI is missing at an element of a set of identifiers – {BBB001_X##1, BBB002##1} – {0112/2///62656_2#BBB001, 0112/2///62656_2#BBB002} – {BBB001, BBB002}
ICID mixed value	MDC_P017 (Class value assignment) MDC_P096 (Property classification) MDC_P097 (Requirement of properties)	– Class value assignment – {(0112/2///62656_2#BBB001##1, abc), (0112/2///62656_2#BBB002##1, efj)} – Property classification – {(0112/2///62656_2#BBB001##1,1,abc), (0112/2///62656_2#BBB002##1,2,efj)} – Requirement of properties – {(0112/2///62656_2#BBB001##1,KEY), (0112/2///62656_2#BBB001##1,NOT NULL)}	a) ID format violation b) identifiers are enclosed between not curly brackets but parentheses c) either RAI or VI is missing at an element of a set of identifiers – {(BBB001##1, abc),(BBB002##1, efj)} – {(0112/2///62656_2#BBB001,1,abc), (0112/2///62656_2#BBB002,2,efj)} – {(BBB001,KEY),(BBB001,NOT NULL)}

Table C.1 *(3 of 3)*

Expected value	Attribute	Valid value example	Cases of invalid values

data type	MDC_P022 (Data type)	– CLASS_REFERENCE and CLASS_INSTANCE
		  – CLASS_REFERENCE_TYPE(0112/2///62656_2#BBB001##1)
		– ENUM_TYPE
		  – ENUM_CODE_TYPE(0112/2///62656_2#BBB001##1)
		  – ENUM_CODE_TYPE(0112/2///62656_2#BBB001##1(a,b,c))
		– Named type
		  – NAMED_TYPE(0112/2///62656_2#BBB001##1)
		– Aggregation
		  – SET(0,?) OF LIST(2,2) OF STRING_TYPE
			a) ID format violation
			b) either RAI or VI is missing at a specified identifier
			  – CLASS_REFERENCE_TYPE(BBB001##1)
			  – ENUM_CODE_TYPE(BBB001(a,b,c))
			  – NAMED_TYPE(0112/2///62656_2#BBB001)
			c) invalid character for aggregation cardinality
			  – SET[0,?] OF LIST[2,2] OF STRING_TYPE

date	MDC_P003_1 (Date of original definition)
	MDC_P003_2 (Date of current version)
	MDC_P003_3 (Date of current revision)
		– 2010-05-01	a) format error
			  – 2010-5-1
			  – 2010/05/01

synonym	MDC_P004_2 (Synonymous name)	– {(motor,en)}
		– {(“display (computer)”,en), (“étalage (ordinateur)”,fr)}
		– {(“””computer””, display”,en), (“””ordinateur””, étalage”,fr)}
			a) aggregation format violation
			  – ABC
			b) invalid escape character syntax
			  – {(display (computer),en)}
			  – {(“computer” display,en)}

Annex D
(informative)

Sample data

During the development of the standard period, sample data parcels for data dictionaries described in Clause 5 will be available at the following URL: <http://std.iec.ch/iec61360>.

Annex E
(informative)

Parcelling tools

For the convenience of readers, parcelling tools which are known to be conformant with this part of IEC 62656 at the date of publication of this Technical Specification are summarized in the following table. For specific use conditions, please contact the relevant URI in the table.

Name	Description	Contact
ParcelMaker™ [5]	Community version is freely available from Toshiba Corporation for registered IEC and ISO experts.	parcel@sel.rdc.toshiba.co.jp

5 ParcelMaker™ is the trade name of a product supplied by Toshiba Corporation. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

Bibliography

[1] IEC 61360-4:1997, *Standard data element types with associated classification scheme for electric components – Part 4: IEC reference collection of standard data element types, component classes and terms*
Available from: <http://std.iec.ch/iec61360>

[2] IEC 61970-301:2011, *Energy management system application program interface (EMS-API) – Part 301: Common information model (CIM) base*

[3] IEC 62656-3:-[6], *Standardized product ontology register and transfer by spreadsheets – Part 3: Interface for common information model*

[4] IEC 61360-6:-[7], *Quality guide*

[5] ISO/IEC 6523 (all parts), *Information technology – Structure for the identification of organizations and organization parts*

[6] ISO 10303-41:2005, *Industrial automation systems and integration – Product data representation and exchange – Part 41: Integrated generic resources: Fundamentals of product description and support*

[7] ISO 8879:1986, *Information processing – Text and office systems – Standard Generalized Markup Language (SGML)*

[8] Mathematical Markup Language (MathML). Third Edition. World Wide Web Consortium Recommendation 21 October 2010. Available from: <http://www.w3.org/TR/MathML3/>

[9] OMG. *OMG Unified Modeling Language (OMG UML), Infrastructure, V2.3.* Needham, MA: OMG (Object Management Group Inc.), 3 May 2010.
Available from: <http://www.omg.org/spec/UML/2.3/Infrastructure/PDF>

[10] NETWORK WORKING GROUP. *Common Format and MIME Type for Comma-Separated Values (CSV) Files.* Network Working Group, 2005-10.
Available from: <http://www.ietf.org/rfc/rfc4180.txt>

[11] OMG. Meta Object Facility (MOF) Core Specification. OMG Available Specification, version 2.0. Needham, MA: OMG (Object Management Group Inc.), 1 January 2006.
Available from: <http://www.omg.org/spec/MOF/2.0>

[12] ISO/IEC 8824-1:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*

_____

6 Under consideration.

7 Under consideration.

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

ENREGISTREMENT D'ONTOLOGIE DE PRODUITS NORMALISÉS ET TRANSFERT PAR TABLEURS –

Partie 2: Guide d'application pour l'utilisation avec le Dictionnaire de données communes de la CEI (le CEI CDD)

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI 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, la CEI – 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 la CEI"). 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 la CEI, participent également aux travaux. La CEI 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 la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI 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 la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) La CEI 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 la CEI. La CEI 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 la CEI, 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 la CEI, 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 la CEI ou de toute autre Publication de la CEI, 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 la CEI peuvent faire l'objet de droits de brevet. La CEI 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 tâche principale des comités d'études de la CEI est l'élaboration des Normes internationales. Exceptionnellement, un comité d'études peut proposer la publication d'une spécification technique

- lorsqu'en dépit de maints efforts, l'accord requis ne peut être réalisé en faveur de la publication d'une Norme internationale, ou
- lorsque le sujet en question est encore en cours de développement technique ou quand, pour une raison quelconque, la possibilité d'un accord pour la publication d'une Norme internationale peut être envisagée pour l'avenir, mais pas dans l'immédiat.

Les spécifications techniques font l'objet d'un nouvel examen trois ans au plus tard après leur publication afin de décider éventuellement de leur transformation en Normes internationales.

La CEI 62656-2, qui est une spécification technique, a été établie par le sous-comité 3D, Propriétés et classes des produits et leur identification, du comité d'études 3 de la CEI: Structures d'informations, documentation et symboles graphiques.

Le texte de cette spécification technique est issu des documents suivants:

Projet d'enquête	Rapport de vote
3D/202/DTS	3D/213/RVC

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

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

Une liste de toutes les parties de la série CEI 62656 publiées sous le titre général *Enregistrement d'ontologie de produits normalisés et transfert par tableurs*, peut être consultée sur le site web de la CEI.

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 la CEI sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. À cette date, la publication sera

- transformée en Norme internationale,
- 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 CEI 62656 intitulée *Enregistrement d'ontologie de produits normalisés et transfert par tableurs* définit les moyens et méthodes d'enregistrement et d'échange d'ontologie des produits exprimés sous forme de tableurs.

La série CEI 62656 est constituée des parties suivantes:

– Partie 1: Structure logique de paquets de données[1];

– Partie 2: Guide d'application pour l'utilisation avec le Dictionnaire de données communes de la CEI (CEI CDD)

– Partie 3: Interface de paquet pour un modèle d'informations commun[2].

1 A paraître.

2 A paraître.

ENREGISTREMENT D'ONTOLOGIE DE PRODUITS NORMALISÉS ET TRANSFERT PAR TABLEURS –

Partie 2: Guide d'application pour l'utilisation avec le Dictionnaire de données communes de la CEI (le CEI CDD)

1 Domaine d'application

La présente partie de la CEI 62656 propose un guide d'application des paquets de données spécifiés dans la CEI 62656-1, utilisé pour définir un dictionnaire de données d'un domaine qui peut être importé et exporté du Dictionnaire de données communes de la CEI (ou CEI CDD) et géré en tant que base de données CEI 61360-4 [1][3]. La présente partie de la CEI 62656 donne des instructions d'interprétation et d'utilisation de la spécification technique définie dans la CEI 62656-1 au sein d'une application logicielle, afin d'éviter les mauvais usages des constructions de données de la CEI 62656-1.

Le domaine d'application de ce guide présente les éléments suivants:

- informations principales de mise en œuvre des paquets de données des dictionnaires de données depuis/vers le CEI CDD,
- exemples classiques de mise en œuvre de fonctions typiques sur des paquets de données,
- extension des classes de conformité pour la mise en œuvre de systèmes reposant sur des paquets afin d'importer/exporter des paquets de données depuis/vers le CEI CDD.

Les éléments suivants ne font pas partie du domaine d'application de la présente partie de la CEI 62656:

- procédures de génération de dictionnaires de données de domaine conformes à la CEI 61360,
- sémantique du dictionnaire de données normalisé,
- explication théorique de la structure logique des paquets de données, abordée dans la CEI 62656-1,
- interface du modèle d'informations commun (CEI 61970-301 [2]), abordée dans la CEI 62656-3 [3].

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).

CEI 61360-1, *Types normalisés d'éléments de données avec plan de classification pour composants électriques – Partie 1: Définitions – Principes et méthodes*

CEI 61360-2, *Types normalisés d'éléments de données avec plan de classification pour composants électriques – Partie 2: Schéma d'un dictionnaire EXPRESS*

---

3 Les chiffres entre crochets se réfèrent à la Bibliographie.

CEI 61987-10:2009, *Mesure et commande des processus industriels – Structures de données et éléments dans les catalogues d'équipement de processus – Partie 10: Liste de propriétés (LOP) pour l'échange électronique de données pour la mesure et le contrôle de processus industriels – Principes essentiels*

CEI 62656-1:—[4], *Enregistrement d'ontologie de produits normalisés et transfert par tableurs - Partie 1: Structure logique pour les paquets de données*

CEI 62720, *Identification des unités de mesure pour le traitement assisté par ordinateur*

ISO 13584-42, *Systèmes d'automatisation industrielle et intégration – Bibliothèque de composants – Partie 42: Méthodologie descriptive: Méthodologie appliquée à la structuration des familles de pièces*

ISO/IEC Guide 77-2:2008, *Guide pour la spécification des classes et des propriétés du produit – Partie 2: Principes techniques et directives*

3 Termes et définitions

Pour les besoins du présent document, les termes et définitions de la CEI 62656-1:—, ainsi que les suivants, s'appliquent.

3.1
cardinalité
nombre minimal et maximal d'occurrences des éléments dans un ensemble

3.2
arbre de classification
arbre d'héritage
arbre is-a
graphe acyclique de classes et de relations présenté sous la forme de nœuds et de branches, dans lequel chaque nœud représente un concept et chaque branche une spécialisation, ou une relation dite "is-a" entre les deux concepts reliés par une branche

3.3
arbre de composition
arbre has-a
graphe acyclique de classes et de relations présenté sous la forme de nœuds et de branches, dans lequel chaque nœud représente un concept et chaque branche une relation partie-tout, ou une relation dite "has-a" entre les deux concepts reliés par la branche

Note 1 à l'article: Un arbre de composition, également appelé agrégation, permet à un nœud de contenir plusieurs sous-nœuds.

3.4
propriété de condition
propriété dont la valeur affecte la décision de valeur d'une autre propriété

3.5
élément ontologique
artefact instancié par une méta-classe, faisant office de métadonnées et utilisé pour préciser la sémantique d'une propriété ou d'une classe ou pour lui ajouter des informations

4 A paraître.

Note 1 à l'article: En général, la notion d'élément ontologique inclut les propriétés. Toutefois, elle fait souvent référence à des artefacts qui ne sont pas des propriétés (des énumérations, des types de données, des documents, des unités de mesure, des termes, des relations, etc.).

Note 2 à l'article: Dans la présente partie de la CEI 62656, toutes les occurrences de "méta-classe" sont remplacées par "feuille de paquet" pour faciliter la compréhension.

3.6
polymorphisme
modèle permettant de remplacer dans le même contexte un concept simple par un concept différent

[SOURCE: CEI 61987-10:2009, 3.1.21, ne s'applique qu'à la version anglaise]

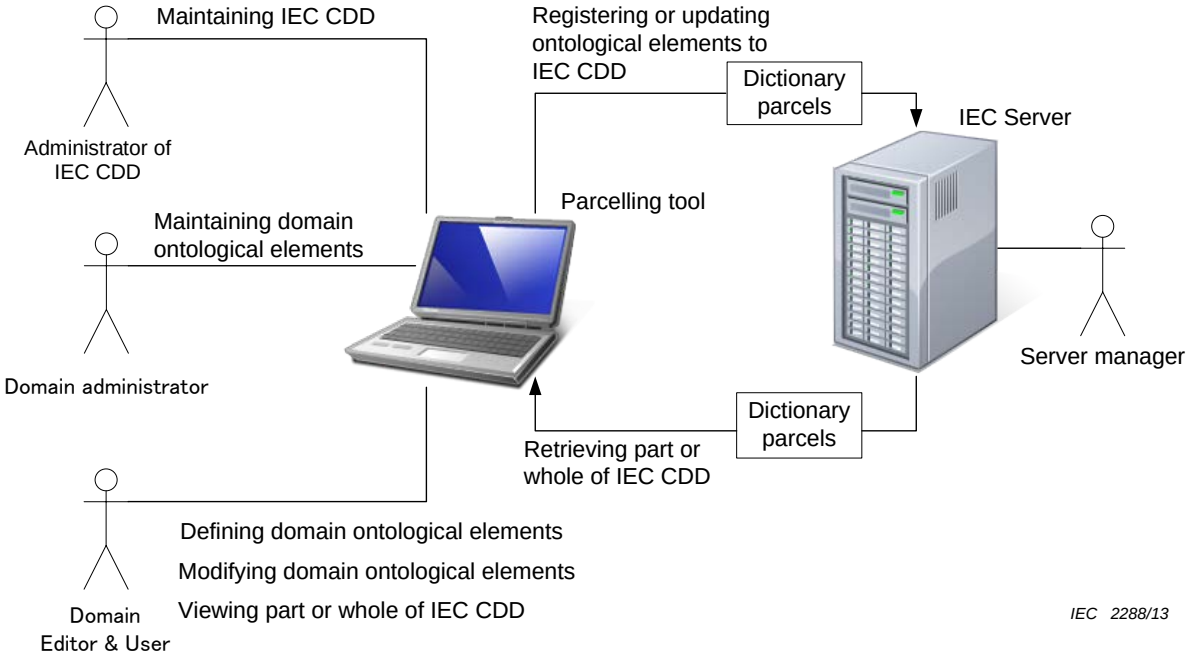
4 Vue d'ensemble

4.1 Généralités

La présente partie de la CEI 62656 est un guide d'application destiné aux utilisateurs de paquets, tels que:

– les experts d'un domaine qui mettent en œuvre des paquets de données pour leur dictionnaire de données, pour l'enregistrement dans la base de données en ligne CEI CDD par des outils de paquetage,
– les utilisateurs qui téléchargent un (morceau du) dictionnaire de données depuis la base de données en ligne CEI CDD,
– les utilisateurs qui éditent ou échangent un (morceau du) dictionnaire de données,
– les fournisseurs d'application qui développent un outil de paquetage (un éditeur, un afficheur ou similaire, par exemple).

Un scénario d'utilisation classique de la série CEI 62656 pour le CEI CDD est présenté à la Figure 1.



Légende

Anglais	Français
Maintaining IEC CDD	Gestion du CEI CDD
Registering or updating ontological elements to IEC CDD	Enregistrement ou mise à jour des éléments ontologiques dans le CEI CDD
Dictionary parcels	Paquets de dictionnaire
IEC server	Serveur CEI
Maintaining domain ontological elements	Gestion des éléments ontologiques du domaine
Parcelling tool	Outil de paquetage
Secretary of SC3D	Secrétaire du SC3D
Server manager	Gestionnaire de serveur
Domain administrator	Administrateur de domaine
Retrieving part or whole of IEC CDD	Extraction de tout ou partie du CEI CDD
Defining domain ontological elements	Définition des éléments ontologiques du domaine
Modifying domain ontological elements	Modification des éléments ontologiques du domaine
Viewing part or whole of IEC CDD	Affichage de tout ou partie du CEI CDD
Domain Editor & User	Éditeur et utilisateur de domaine

Figure 1 – Scénario classique d'utilisation

Pour faciliter la lecture de la présente partie de la CEI 62656, "paquet" et "attribut" sont utilisés à la place de "méta-classe" et "méta-propriété", respectivement. Par exemple, "méta-classe de classes" est reformulé "paquet de classes".

4.2 Dictionnaire de données

L'ISO/CEI Guide 77-2 recommande qu'il convient que chaque dictionnaire de données soit conforme au modèle de dictionnaire commun ISO 13584-42/CEI 61360-2. Dans le modèle de dictionnaire commun ISO 13584-42/CEI 61360-2, chaque dictionnaire de données est représenté par un ensemble de classes et leurs propriétés caractéristiques associées. Le

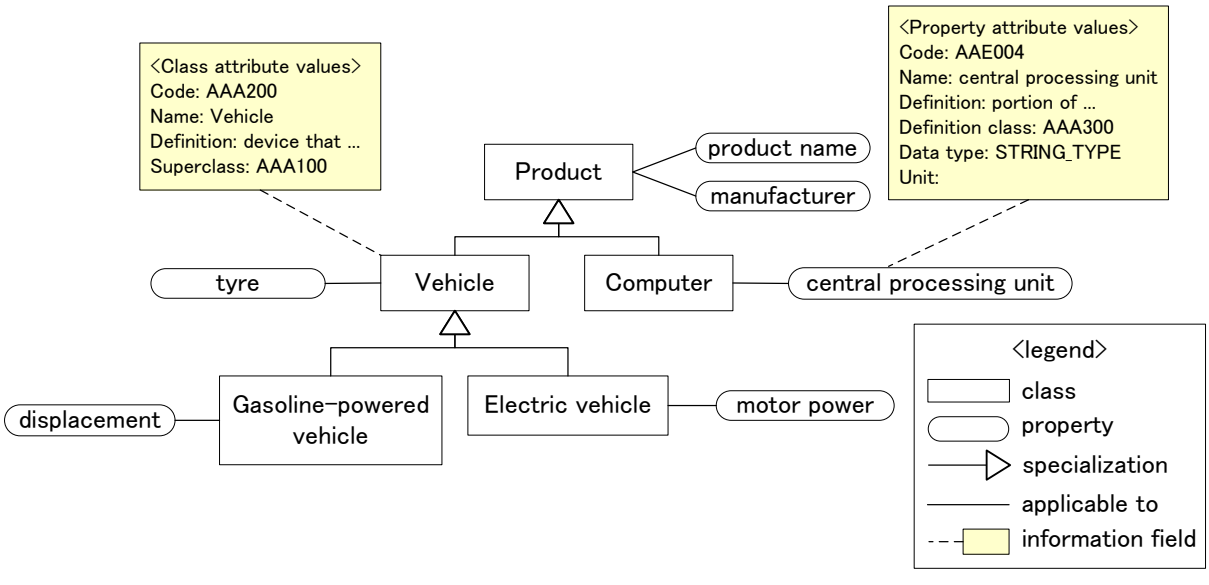
CEI CDD est un dictionnaire de données géré comme une base de données et qui définit une ontologie des produits et services (y compris les composants, les matériaux, les systèmes et les concepts) essentiels dans les domaines électrotechniques.

Pour un dictionnaire de données, les classes et les propriétés sont des éléments fondamentaux. Une classe est un concept intégré sous la forme d'une structure de données permettant de représenter un objet du monde réel (un produit, un matériau, etc.), une propriété étant un concept intégré sous la forme d'une autre structure de données permettant de caractériser une ou plusieurs classes. Une classe est composée d'un ensemble de propriétés caractéristiques (dans un cas extrême, elle ne comporte qu'une seule propriété) et doit pouvoir être facilement distinguée des autres classes par les propriétés de membre de l'ensemble. En d'autres termes, si deux classes comportent exactement les mêmes ensembles de propriétés, elles ne sont pas différentiables par une machine. Toutefois, ce type de style de modélisation de classe n'est pas recommandé dans la CEI 61360-1.

Si, d'un point de vue conceptuel, deux classes partagent certaines caractéristiques, une classe généralisée les contenant peut être définie. Cette classe généralisée est appelée "superclasse" de ces classes groupées, chacune d'elles étant appelée "sous-classe" par rapport à la superclasse. Ce type de relation entre une superclasse et une sous-classe est familièrement appelée relation "is-a". Noter que, conformément au modèle de dictionnaire commun ISO 13584-42/CEI 61360-2, chaque classe peut comporter une classe faisant office de superclasse, et peut comporter plusieurs sous-classes. Si un certain nombre de classes ne dispose d'aucune superclasse apparente, une classe virtuelle (dite "classe universelle") est censée exister et se comporter comme une seule superclasse commune. En conséquence, la totalité des relations qu'entretiennent les classes forme une arborescence (plus exactement, un graphe acyclique).

Si des caractéristiques sont partagées entre plusieurs classes, elles sont modélisées en "propriétés" et doivent être définies au niveau d'une classe générale de classes, puis héritées dans ses sous-classes.

La Figure 2 donne un exemple simple de dictionnaire de données. Dans cette figure, les concepts de "Véhicule à moteur à essence" et de "Véhicule électrique" sont deux types différents de véhicule, leur superclasse "Véhicule" peut donc être définie avec des propriétés communes, c'est-à-dire que le "nom du produit", le "fabricant" et le "pneu" sont définis à ce niveau dans l'arbre des classes. De même, "Véhicule" et "Ordinateur" sont deux concepts de produit différents. Leur superclasse "Produit" est donc défianie avec des propriétés communes, c'est-à-dire que le "nom de produit" et le "fabricant" sont définis à ce niveau. En conséquence, le dictionnaire de données contenant ces classes est composé d'une arborescence (voir Figure 2).



IEC 2289/13

Légende

Anglais	Français
<Class attribute values>	<Valeurs des attributs de la classe>
Code	Code
Name	Nom
Vehicle	Véhicule
Definition	Définition
device that ...	dispositif qui...
Superclass	Superclasse
Product	Produit
product name	nom du produit
manufacturer	fabricant
<Property attribute values>	<Valeurs des attributs de la propriété>
central processing unit	unité centrale de traitement
portion of ...	partie de...
Definition class	Classe de définition
Data type	Type de données
Unit	Unité
tyre	pneu
Computer	Calculateur/Ordinateur
displacement	déplacement
Gasoline-powered vehicle	Véhicule à moteur à essence
Electric vehicle	Véhicule électrique
motor power	puissance moteur
<legend>	<légende>
class	classe
property	propriété
specialization	spécialisation
applicable to	applicable à
information field	zone d'information

Figure 2 – Dictionnaire de données

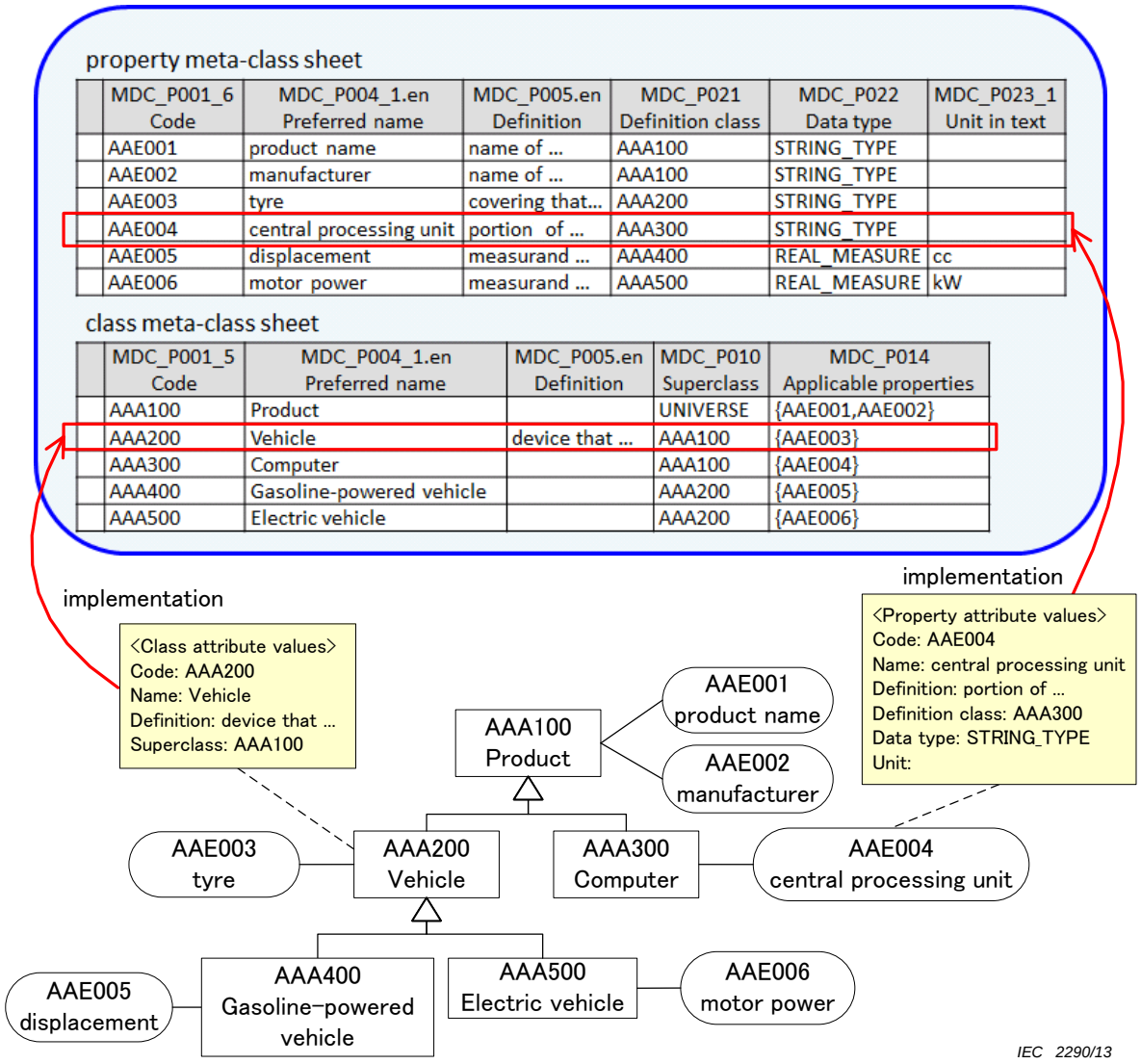
Conformément au modèle de dictionnaire commun ISO 13584-42/CEI 61360-2, chaque entité dispose de son identificateur unique, contenant des informations structurelles, permettant de distinguer les entités. Par exemple, une classe comporte des zones d'informations telles que le nom, la définition, la superclasse, alors qu'une propriété comporte des zones d'informations telles que le nom, la définition, la classe de définition, le type de données et l'unité. Les cases reliées par une ligne en pointillés à la Figure 2 sont des exemples de zones d'informations contenues dans une classe et une propriété.

4.3 Paquet de données

La CEI 62656-1, parfois appelée par son nom abrégé "paquet de données", définit un ensemble de conteneurs d'informations d'ontologie de produits (c'est-à-dire, un arbre de familles de produit et leurs caractéristiques), en triant les informations dans des ensembles de données homogènes (une liste de classes, de propriétés et d'énumérations, par exemple). Chaque ensemble est appelé "paquet de données". Pour être précis, un paquet de données peut être utilisé non seulement pour définir un dictionnaire de données, mais également pour représenter une bibliothèque ou un catalogue de produits avec des valeurs particulières de produits individuels. Cependant, pour les lecteurs du présent document, l'intérêt principal porte sur l'utilisation des paquets de données pour représenter une ontologie de produits. Dans ce contexte, les lecteurs sont censés comprendre que chacun des paquets de données contient un ensemble homogène d'ontologies de produits. Une feuille placée dans un tableur est une forme classique de mise en œuvre de ce type d'ensemble de données, souvent utilisée dans l'ingénierie quotidienne; Par conséquent, dans la suite de la présente

Spécification technique, un paquet de données est reformulé "feuille de paquet" afin de représenter visuellement la forme probable de mise en œuvre.

Un paquet de classes (c'est-à-dire une méta-classe de classe) permet de désigner et d'instancier des classes. Le paquet de classes contient des attributs (c'est-à-dire des méta-propriétés) permettant de décrire les caractéristiques d'une classe (son code de classe, son nom préférentiel, sa définition et sa superclasse, par exemple). De même, un paquet de propriétés (c'est-à-dire une méta-classe de propriétés) permet de concevoir des propriétés. Le paquet de propriétés contient des attributs permettant de décrire des informations de propriété (le code de propriété, le nom préférentiel, la définition, la classe de définition, le type de données et l'unité, par exemple). Pour mettre en œuvre un dictionnaire de propriétés au sein d'ensembles de données homogènes de paquets, il convient de prépare des tableurs, chacun mettant en œuvre une seule catégorie de l'ensemble du paquet. Par exemple, dans le cas présenté à la Figure 2, deux feuilles correspondant respectivement à un paquet de classes et à un paquet de propriétés sont requises (voir Figure 3).



Légende

Anglais	Français
property meta-class sheet	feuille de propriété de méta-classe
Code	Code
Preferred name	Nom préférentiel

Anglais	Français
Definition	Définition
Definition class	Classe de définition
Data type	Type de données
Unit in text	Unité en texte
product name	nom du produit
name of …	Nom de…
manufacturer	fabricant
tyre	pneu
covering that…	couvrant…
portion of …	partie de…
central processing unit	unité de traitement central
displacement	déplacement
measurand …	mesurande…
motor power	puissance moteur
class meta-class sheet	feuille de classe de méta-classe
Superclass	Superclasse
Applicable properties	Propriétés applicables
Product	Produit
Vehicle	Véhicule
device that…	dispositif qui…
Computer	Ordinateur
Gasikube-powered vehicle	Véhicule à moteur à essence
Electric vehicle	Véhicule électrique
Implementation	Mise en œuvre
<Class attribute values>	<Valeurs des attributs de la classe>
Code	Code
Name	Nom
Vehicle	Véhicule
Definition	Définition
device that …	dispositif qui…
Superclass	Superclasse
<Property attribute values>	<Valeurs des attributs de la propriété>
portion of …	partie de…
Definition class	Classe de définition
Data type	Type de données
Unit	Unité
Product	Produit
product name	nom du produit
manufacturer	fabricant
tyre	pneu
Vehicle	Véhicule
Computer	Calculateur/Ordinateur
central processing unit	unité centrale de traitement
displacement	déplacement

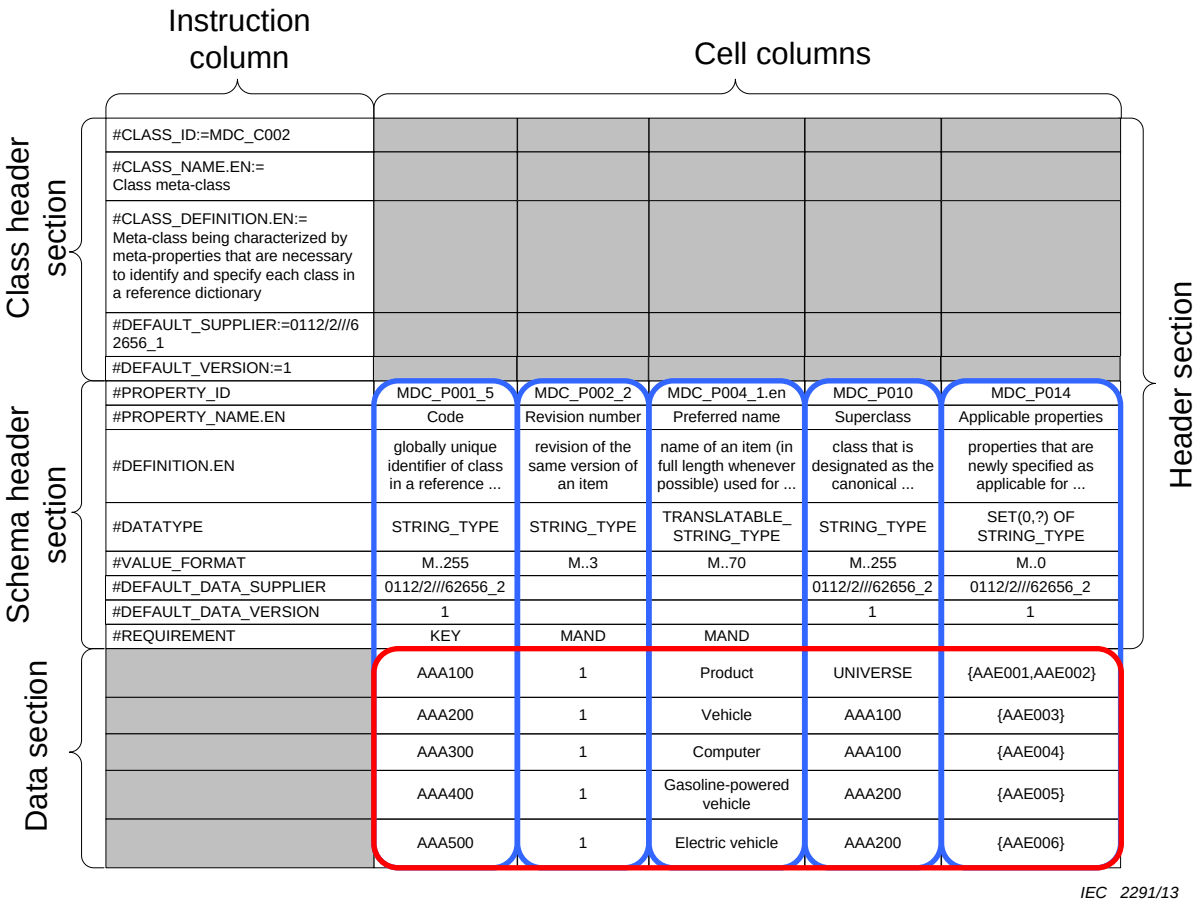
Anglais	Français
Gasoline-powered vehicle	Véhicule à moteur à essence
Electric vehicle	Véhicule électrique
motor power	puissance moteur

Figure 3 – Mise en œuvre d'un tableur

La Figure 4 illustre la structure de base d'une feuille de paquet. Chaque feuille de paquet se compose de sa section d'en-tête et de sa section de données.

La section d'en-tête est en outre composée d'une section en-tête de classe et d'une section en-tête de schéma. Dans la section en-tête de classe, les informations contenues dans la feuille de paquet sont décrites, comme par exemple, l'identificateur du paquet de données et les valeurs par défaut qui peuvent être appliquées à un attribut du paquet dans la section d'en-tête (voir 5.2). Dans la section en-tête de schéma, qui contient les métadonnées de description des valeurs de la section de données, chaque attribut du paquet est spécifié dans chaque cellule de la colonne.

Dans la section de données, chaque élément ontologique est décrit dans une ligne. Dans chaque cellule, la valeur de l'attribut, qui est spécifiée dans sa colonne correspondante, est décrite pour définir un tel élément ontologique.



Légende

Anglais	Français
Instruction column	Colonne d'instruction
Cell columns	Colonnes de cellule
Class header section	Section en-tête de classe

Anglais	Français
Schema header section	Section en-tête de schéma
Data section	Section de données
Header section	Section en-tête
#CLASS_NAME.EN:= Class meta-class	#CLASS_NAME.EN:= Méta-classe de classes
#CLASS_DEFINITION.EN:= Meta-class being characterized by meta-properties that are necessary to identify and specify each class in a reference dictionary	#CLASS_DEFINITION.EN:= Méta-classe caractérisée par des méta-propriétés qui sont nécessaires pour identifier et spécifier chaque classe dans un dictionnaire de référence
Code	Code
Revision number	Numéro de révision
Preferred name	Nom préférentiel
Superclass	Superclasse
Applicable properties	Propriétés applicables
globally unique identifier of class in a reference ...	identificateur global unique de classe dans un dictionnaire de référence...
revision of the same version of an item	révision de la même version d'un élément
name of an item (in full length whenever possible) used for ...	nom d'un élément (complet, si possible) utilisé pour...
class that is designated as canonical ...	classe qui est désignée comme canonique...
Properties that are newly specified as applicable for ...	propriétés qui sont nouvellement spécifiées et applicables...
Product	Produit
Vehicle	Véhicule
Computer	Calculateur/Ordinateur
Gasoline-powered vehicle	Véhicule à moteur à essence
Electric vehicle	Véhicule électrique

Figure 4 – Feuille d'un paquet

4.4 Feuilles de paquet vierges

Il est possible d'obtenir des feuilles de paquet vierges pour éditer un dictionnaire de données de l'une des manières suivantes:

– sur le site Web du CEI CDD à l'adresse suivante: <http://std.iec.ch/iec61360>

ou de la générer:

– à l'aide d'un outil de paquetage.

Une liste des outils conformes à la présente Spécification technique est donnée en Annexe E.

Quatre feuilles contenant des feuilles de dictionnaire, de classe, de propriété et de fournisseur sont obligatoires pour mettre en œuvre un dictionnaire de données. Les autres feuilles sont facultatives et sont préparées uniquement en cas de besoin dans le processus de renseignement complet des informations décrites dans les quatre feuilles obligatoires.

5 Cas communs de définition d'éléments ontologiques

5.1 Sémantique

Dans la série CEI 62656 et la série ISO 13584/CEI 61360, les principaux concepts de produits sont représentés par des classes, et leurs caractéristiques modélisées par des propriétés. Certains attributs des classes et propriétés impliquent d'utiliser d'autres paquets

(les paquets de type de données, d'énumération et de terme, par exemple). Par conséquent, dans certains cas, la première étape consiste à déterminer la sémantique de chaque élément ontologique modélisé par le paquet correspondant pour permettre aux paquets de données de décrire le dictionnaire de données. Les paquets mentionnés ci-dessus comportent un ensemble d'attributs de base permettant de décrire les noms et significations de leurs éléments ontologiques.

Pour décrire les noms, il existe 3 types d'attribut définis dans la CEI 62656-1: MDC_P004_1 (Nom préférentiel), MDC_P004_2 (Nom synonyme) et MDC_P004_3 (Nom abrégé). De même, pour décrire ou préciser la signification d'un élément ontologique, il existe 3 types d'attribut: MDC_P005 (Définition), MDC_P007_1 (Note) et MDC_P007_2 (Remarque).

Les attributs ci-dessus, à l'exception de MDC_P004_2, sont définis en tant que TRANSLATABLE_STRING_TYPE pour la localisation du contenu décrit dans une langue spécifiée comme étant la langue source. Ces attributs peuvent comporter plusieurs colonnes identifiées par un code de langue particulier (qui peut être combiné à un code de pays pour préciser, le cas échéant une variante linguistique). Par exemple, si une feuille de paquet contient des noms en anglais et français, des colonnes identifiées par "MDC_P004_1.en" et "MDC_P004_1.fr" doivent être préparées pour décrire les noms préférentiels dans la feuille.

A l'inverse, l'attribut MDC_P004_2 est défini en tant que SET(0,?) OF LIST(2,2) OF STRING_TYPE. Par conséquent, une seule colonne est prévue et un nom synonyme dans chaque langue doit être décrit comme étant une valeur de cet attribut. Dans chaque zone de cet attribut, un ensemble contenant la combinaison d'un nom et d'un code de langue (accompagné le cas échéant d'un code de pays) dans l'ordre est attendu comme valeur valide. Par exemple, si le terme anglais "battery" et sa traduction en français "batterie" sont indiqués comme étant des noms synonymes d'un élément ontologique, la valeur est décrite "{(battery,en),(batterie,fr)}" ou "{(batterie,fr),(battery,en)}".

Chaque feuille de paquet contient une instruction facultative #SOURCE_LANGUAGE spécifiant la langue dans laquelle le contenu sémantique d'origine du paquet est préparé. Par exemple, si la colonne d'instruction d'un paquet contient une description "#SOURCE_LANGUAGE:=en", le contenu en anglais doit être la source, le contenu dans les autres langues doit être considéré comme étant une traduction du contenu anglais dans la feuille. Si aucune langue n'est spécifiée dans la zone ou si l'instruction ne peut être trouvée dans la feuille de paquet, l'anglais doit par défaut être considéré comme la langue source de la feuille.

NOTE La description des valeurs des attributs susmentionnés suit la CEI 61360-6 [4].

La Figure 5 donne un exemple de feuille de paquet de classes contenant uniquement les attributs décrivant la sémantique des éléments ontologiques. Trois classes sont décrites dans la section de données, c'est-à-dire un condensateur et ses sous-classes, qui sont susceptibles de se trouver réellement dans le CEI CDD.

#CLASS_ID:= MDC_C002						
#CLASS_NAME.en:= Class meta-class						
#PROPERTY_ID	MDC_ P004_1.en	MDC_P004_2	MDC_ P004_3.en	MDC_P005.en	MDC_P007_1.en	MDC_ P007_2.en
#PROPERTY_ NAME.en	Preferred name	Synonymous name	Short name	Definition	Note	Remark
#DEFAULT_ DATA_SUPPLIER						
#DEFAULT_ DATA_VERSION						
	Capacitor	{(capacitor,en)}		system of two conductors (plates) separated over the extent of its surfaces by a thin insulating medium (dielectric), its intended characteristic being capacitance	Since capacitance is a function of temperature, it may still vary with temperature.	
	Fixed capacitor	{(fixed,en)}		capacitor that has no designed provision for changing its capacitance value	Since capacitance is a function of temperature, it may still vary with temperature.	
	Variable capacitor	{(variable,en)}		capacitor designed so that its main property can be varied by mechanically changing the spatial relationship of their parts		

IEC 2292/13

Légende

Anglais	Français
Preferred name	Nom préférentiel
Synonymous name	Nom synonyme
Short name	Nom abrégé
Definition	Définition
Note	Note
Remark	Remarque
Capacitor	Condensateur
system of two conductors (plates) separated over the extent of its surfaces by a thin insulating medium (dielectric), its intended characteristics being capacitance	système de deux conducteurs (plaques) séparés sur toute la surface par un mince support isolant (diélectrique), sa caractéristique prévue étant la capacitance
Since capacitance is a function of temperature, it may still vary with temperature.	La capacitance étant fonction de la température, elle peut varier avec la température.
Fixed capacitor	Condensateur fixe
capacitor that has no designed provision for changing its capacitance value	condensateur n'ayant pas la possibilité par conception de modifier la valeur de sa capacitance
Variable capacitor	Condensateur variable
capacitor designed so that its main property can be varied by mechanically changing the spatial relationship of their parts	condensateur conçu de manière à pouvoir faire varier sa propriété principale en modifiant mécaniquement la relation spatiale de ses composants

Figure 5 – Définitions sémantiques des éléments ontologiques

5.2 Attribution d'identificateur

Chaque élément ontologique doit être identifié conformément à l'identificateur de concept international (ICID (International Concept Identifier)), qui est défini comme étant le principal schéma d'identification dans la CEI 62656-1. Ce type d'identificateur permet non seulement de distinguer les éléments ontologiques, mais également d'identifier une relation entre eux.

Chaque feuille de paquet d'un dictionnaire de données contient son propre attribut pour décrire les identificateurs de ses éléments ontologiques. Ce type d'attribut est identifié en MDC_P001_X (Code), où X est un entier positif. Par exemple, MDC_P001_5 est fourni pour le paquet de classes et MDC_P001_6 pour le paquet de propriétés.

Chaque identificateur est composé d'une concaténation de RAI (Registration authority identifier – "Identificateur d'autorité d'enregistrement"), de DI (Data identifier – "Identificateur de données") et de VI (Version identifier – "Identificateur de version"), dans cet ordre. En premier lieu, le RAI indique le fournisseur responsable d'un élément de données conforme à l'ISO/CEI 6523 [5]. Par exemple, 0112/2///61360_4 est le RAI par défaut du CEI CDD. Ensuite, le DI indique le code attribué à chaque élément ontologique qui doit être attribué de manière unique dans les éléments ontologiques gérés par le même RAI. Dans le CEI CDD, il est demandé que le DI soit composé de six lettres, les trois premières étant des caractères alphabétiques romains, et les trois dernières des nombres entiers (format AAANNN). Enfin, le VI indique le numéro de version de chaque élément ontologique qui doit être incrémenté à chaque nouvelle version. Une nouvelle version d'un élément ontologique doit avoir une compatibilité ascendante avec toutes les anciennes versions du même concept.

Il convient d'éviter d'utiliser les lettres majuscules I et O dans les identificateurs, car, sur certains systèmes informatiques, il est difficile de les distinguer respectivement des caractères numériques 1 et 0, ce qui peut poser des problèmes d'erreur de lecture des identificateurs.

La CEI 62656-1 propose une méthode de notation sténographique des identificateurs dans la section d'en-tête et la section de données. Étant donné que presque tous les éléments d'un dictionnaire de données au niveau de chaque méta-couche peuvent comporter un RAI et un VI communs, la présente partie de la CEI 62656 recommande que toutes les applications utilisent ce mécanisme pour:

– simplifier la maintenance d'un dictionnaire de données,
– attribuer un RAI et un VI provisoires en concevant une ossature de dictionnaire de données,
– éviter d'entrer plusieurs fois la même valeur de RAI et de VI,
– réduire la taille des données,
– assurer une meilleure lisibilité.

Dans le présent document, le RAI et le VI de chaque attribut de la section d'en-tête de chaque paquet sont omis pour des raisons de clarté. D'une manière totalement conforme avec la CEI 62656-1, afin d'être omis dans la description, le RAI et le VI d'une feuille de paquet doivent être explicitement déclarés dans les instructions "#DEFAULT_SUPPLIER" et "#DEFAULT_VERSION" de la section en-tête de classe de la feuille de paquet, mais dans le présent document, ces instructions sont également omises dans chaque figure, qui est utilisée pour expliquer les feuilles de paquet.

De plus, dans le présent document, le RAI et le VI de chaque attribut sont omis pour faciliter la lecture. Dans un style en totale conformité avec la CEI 62656-1, afin d'être omis dans une valeur d'attribut, le RAI et le VI doivent respectivement être explicitement déclarés dans l'instruction "#DEFAULT_DATA_SUPPLIER" et "#DEFAULT_DATA_VERSION" de l'attribut, mais dans le présent document, ces lignes d'instruction sont également omises dans chaque figure, qui est utilisée pour expliquer les feuilles de paquet.

La Figure 6 donne un exemple de feuille de paquet de classes contenant l'attribut MDC_P001_5 pour l'identification des éléments ontologiques (c'est-à-dire que la feuille est l'extension de celle décrite à la Figure 5). Dans la colonne de l'attribut MDC_P001_5, l'identificateur de chaque classe est décrit. Dans cet exemple, le RAI par défaut "0112/2///61360_4" et le VI par défaut "003" sont donnés, respectivement. Conformément à la notation sténographique, le RAI par défaut est appliqué aux identificateurs du deuxième élément ontologique ("condensateur fixe") et du troisième élément ontologique

("condensateur variable"). De même, le VI par défaut est appliqué au troisième élément ontologique.

#CLASS_ID:= MDC_C002							
#CLASS_NAME.en:= Class meta-class							
#PROPERTY_ID	MDC_P001_5	MDC_ P004_1.en	MDC_P004_2	MDC_ P004_3.en	MDC_P005.en	MDC_P007_1.en	MDC_ P007_2.en
#PROPERTY_ NAME.en	Code	Preferred name	Synonymous name	Short name	Definition	Note	Remark
#DEFAULT_ DATA_SUPPLIER	0112/2///61360_4						
#DEFAULT_ DATA_VERSION	003						
	0112/2///61360_4# AAA020##002	Capacitor	{(capacitor,en)}		system of two conductors (plates) separated over the extent of its surfaces by a thin insulating medium (dielectric), its intended characteristic being capacitance	Since capacitance is a function of temperature, it may still vary with temperature.	
	AAA021##004	Fixed capacitor	{(fixed,en)}		capacitor that has no designed provision for changing its capacitance value	Since capacitance is a function of temperature, it may still vary with temperature.	
	AAA031	Variable capacitor	{(variable,en)}		capacitor designed so that its main property can be varied by mechanically changing the spatial relationship of their parts		

IEC 2293/13

Légende

Anglais	Français
Code	Code
Preferred name	Nom préférentiel
Synonymous name	Nom synonyme
Short name	Nom abrégé
Definition	Définition
Note	Note
Remark	Remarque
Capacitor	Condensateur
system of two conductors (plates) separated over the extent of its surfaces by a thin insulating medium (dielectric), its intended characteristics being capacitance	système de deux conducteurs (plaques) séparés sur toute la surface par un mince support isolant (diélectrique), sa caractéristique prévue étant la capacitance
Since capacitance is a function of temperature, it may still vary with temperature.	La capacitance étant fonction de la température, elle peut varier avec la température.
Fixed capacitor	Condensateur fixe
capacitor that has no designed provision for changing its capacitance value	condensateur n'ayant pas la possibilité de modifier la valeur de sa capacitance
Variable capacitor	Condensateur variable
capacitor designed so that its main property can be varied by mechanically changing the spatial relationship of their parts	condensateur conçu de manière à pouvoir faire varier sa propriété principale en modifiant la relation spatiale de ses composants

Figure 6 – Identification des éléments ontologiques

5.3 Attribution d'une classe de définition

Conformément à la CEI 62656-1, chaque élément ontologique (outre le paquet de classes) doit comporter une classe de définition qui précise son domaine d'application. Ces informations doivent être décrites en tant que valeur de l'attribut obligatoire MDC_P021 (Classe de définition) contenu dans chaque feuille de paquet.

Les propriétés, types de données et documents doivent être en outre applicables aux classes dans lesquelles ces éléments ontologiques sont utilisés. Les attributs de description de ces

informations sont fournis dans une feuille de paquet de classes, dont les identificateurs sont MDC_P014 (Propriétés applicables), MDC_P015 (Types applicables) et MDC_P094 (Documents applicables). D'autres informations sont disponibles en 6.2.

5.4 Attributs à prendre en considération

L'Annexe E et l'Annexe F de la CEI 62656-1 définissent l'exigence relative aux attributs dans chaque paquet. Dans la plupart des cas, les attributs KEY, NOT NULL ou MANDATORY sont essentiels ou importants pour définir des éléments ontologiques. Par conséquent, il convient de les afficher.

L'instruction prédéfinie "#REQUIREMENT" indique l'exigibilité de chaque attribut. Si une feuille de paquet contient ce type d'instruction, il convient de l'afficher de manière explicite lors d'une implémentation.

6 Spécification des structures des dictionnaires de données

6.1 Généralités

Il existe deux types fondamentaux de structures de dictionnaire de données, à savoir l'arbre de classification et l'arbre de composition. Parfois, le premier est appelé arbre "is-a" ou arbre d'héritage, et le deuxième arbre "has-a" dans d'autres ouvrages de référence. Ces deux structures doivent être conformes aux principes du modèle de dictionnaire commun ISO 13584-42/CEI 61360-2.

6.2 Arbre de classification

Un arbre de classification est composé d'une relation parent-enfant entre classes, dans laquelle une classe enfant (appelée "sous-classe") hérite de toutes les propriétés de sa classe parent (appelée "superclasse"). Chaque classe doit simplement disposer d'une superclasse et peut comporter une ou plusieurs sous-classe(s).

Chaque dictionnaire de données de domaine est censé comporter une classe racine pour la conformité logique. Conformément au modèle de dictionnaire commun ISO 13584-42/CEI 61360-2, une classe universelle abstraite intitulée UNIVERSE est spécifiée à la racine pour collecter tous les dictionnaires de données de domaine. La classe UNIVERSE ne doit contenir aucune propriété.

Pour éviter de répliquer les mêmes propriétés, type de données et document dans plusieurs branches, il convient de définir ces éléments ontologiques qui peuvent être utilisés en commun par plusieurs classes dans leur classe générale commune.

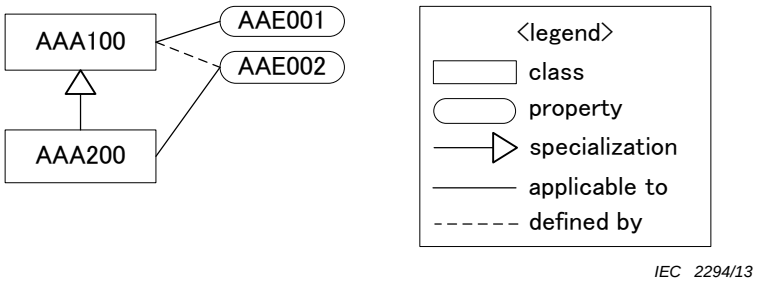
Les énumérations, les unités de mesure et les termes deviennent applicables à la classe de définition, l'applicabilité des propriétés, types de données et documents doit être déclarée de manière explicite dans la classe de définition ou une sous-classe correspondant à la classe de définition. Dès qu'un élément ontologique est applicable à une classe, il s'applique dans toutes les sous-classes de la classe.

L'attribut MDC_P010 (Superclasse) du paquet de classes spécifie l'identificateur d'une superclasse d'une classe.

Les attributs MDC_P014 (Propriétés applicables), MDC_P015 (Types applicables) et MDC_P094 (Documents applicables) du paquet de classes spécifient respectivement une liste d'identificateurs de propriétés, de types de données et de documents, qui s'appliquent à une classe. Étant donné que les propriétés, types de données et documents hérités de classes supérieures (propriétés, types de données et documents connus) doivent déjà être décrits comme étant applicables dans une classe supérieure pertinente, au point qu'ils deviennent en premier lieu applicables, les attributs de description des propriétés, types de données et

documents hérités, c'est-à-dire "XYZ applicable connu" doivent être omis afin d'éviter les incohérences dans un paquet de données utilisé pour échanger des données avec des systèmes ou outils externes. Étant donné que toutes les modifications apportées à une classe supérieure, concernant une propriété, un type de données ou un document hérité, doivent être propagées à toutes ses sous-classes, des incohérences sont susceptibles de se produire compte tenu des capacités de validation limitées et différentes de chaque système et outil. Pour des raisons pratiques, ces attributs peuvent être présentés dans un outil en tant qu'attributs dérivés, et doivent être marqués DER pour leurs exigences (#REQUIREMENT), mais ne doivent pas être utilisés pour échanger des données avec des partenaires externes.

La Figure 7 présente un exemple d'arbre de classification simple contenant les deux classes AAA100 et AAA200 et les deux propriétés AAE001 et AAE002. La classe AAA200 est définie en tant que sous-type de la classe AAA100. Dans cette figure, les propriétés AAE001 et AAE002 sont définies dans la classe AAA100, mais seule la propriété AAE001 s'applique à la classe AAA100. La propriété AAE002 devient applicable à la sous-classe AAA200 par héritage et, par conséquent, la classe AAA100 ne comportant qu'une propriété applicable, la classe AAA200 en comporte deux.



Légende

Anglais	Français
legend	légende
class	classe
property	propriété
specialization	spécialisation
applicable to	applicable à
defined by	défini par

Figure 7 – Exemple d'arbre de classification simple

La Figure 8 présente des paquets conjonctifs permettant de décrire l'arbre de classification présenté à la Figure 7. Dans la feuille de paquet de classes, les deux classes sont définies et la relation parent-enfant entre les classes est décrite dans l'attribut MDC_P010. Dans la feuille de propriété, les deux propriétés sont définies. La propriété AAE001 n'est pas énumérée dans l'attribut MDC_P014 de la classe AAA200, car il s'agit d'une propriété applicable connue, qui s'applique déjà à sa superclasse AAA100.

#CLASS_ID:= **MDC_C002**				
#CLASS_NAME.en:= **Class meta-class**				
#PROPERTY_ID	MDC_P001_5	MDC_P010	MDC_P011	MDC_P014
#PROPERTY_ NAME.en	Code	Superclass	Class type	Applicable properties
	AAA100	**UNIVERSE**	ITEM_CLASS	**{AAE001}**
	AAA200	**AAA100**	ITEM_CLASS	**{AAE002}**

#CLASS_ID:= **MDC_C003**					
#CLASS_NAME.en:= **Property meta-class**					
#PROPERTY_ID	MDC_P001_6	MDC_P020	MDC_P021	MDC_P022	MDC_P023_1
#PROPERTY_ NAME.en	Code	Property data element type	Definition class	Data type	Unit in text
	AAE001	NON_DEPENDENT_ P_DET	**AAA100**	STRING	
	AAE002	NON_DEPENDENT_ P_DET	**AAA100**	REAL_MEASURE	m

IEC 2295/13

Légende

Anglais	Français
Code	Code
Superclass	Superclasse
Class type	Type de classe
Applicable properties	Propriétés applicables
Property data element type	Type de propriétés de l'élément de données
Definition class	Classe de définition
Data type	Type de données
Unit in text	Unité en texte

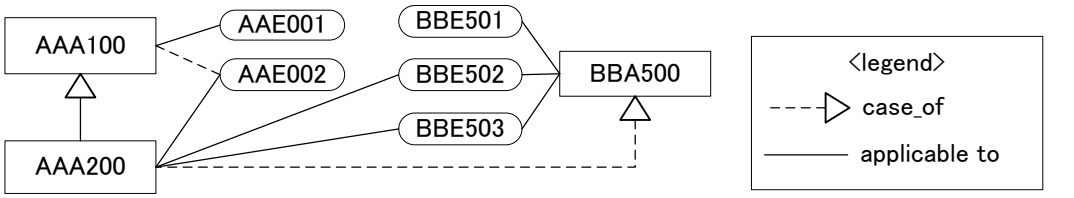
Figure 8 – Mise en œuvre de paquet pour des arbres de classification simples

6.3 Réutilisation des propriétés, des types de données et des documents dans d'autres branches

Pour pouvoir réutiliser des propriétés, types de données et documents existants définis dans une autre branche d'un arbre de classification, un mécanisme appelé case_of est prévu (voir CEI 61360-2). Le mécanisme case_of permet à une classe de spécifier d'autres classes se trouvant dans une autre branche du même dictionnaire de données ou d'un autre dictionnaire de données, puis d'importer des propriétés, types de données et/ou documents applicables depuis les classes spécifiées. Noter que les propriétés, types de données et documents importés sont immédiatement applicables à la classe d'importation et qu'ils sont hérités par ses sous-classes.

Les attributs MDC_P090 (Propriétés importées), MDC_P091 (Types importés) et MDC_P093 (Documents importés) spécifient respectivement une liste d'identificateurs de propriétés, types de données et documents importés. Dans ce cas, un ensemble d'identificateurs de classes à partir duquel ces propriétés, types de données et documents sont importés doit être décrit dans l'attribut MDC_P013 (Is case of).

La Figure 9 présente un exemple de réutilisation de propriétés existantes en important ces propriétés de la classe existante respective. La classe AAA100 est celle du dictionnaire de données présenté à la Figure 7. La classe BBA500 est une classe se trouvant dans un autre dictionnaire de données composé d'une classe et des trois propriétés BBE501 à BBE503. Dans cet exemple, la classeAAA200 est un cas particulier de la classe BBA500, puisqu'elle importe certaines propriétés de la classe BBA500. Par suite de la relation case_of, la classe AAA200 importe les deux propriétés BBE502 et BBE503 depuis la classe BBA500. Noter que ces propriétés sont un sous-ensemble des propriétés applicables de la classe BBA500.



IEC 2296/13

Légende

Anglais	Français
<legend>	<légende>
case_of	case_of
applicable to	applicable à

Figure 9 – Exemple de mécanisme d'importation

La Figure 10 présente une feuille mise à jour du paquet de classes présenté à la Figure 8, pour décrire le dictionnaire de données de la Figure 9. Pour la classe AAA200, la classe BBA500 est spécifiée par l'attribut MDC_P013 (Is case of), pour définir la relation case_of entre ces classes. De même, les deux propriétés importées BBE502 et BBE503 sont énumérées dans l'attribut MDC_P090 (Propriétés importées) de la classe AAA200.

#CLASS_ID:= **MDC_C002**						
#CLASS_NAME.en:= **Class meta-class**						
#PROPERTY_ID	MDC_P001_5	MDC_P010	MDC_P011	MDC_P013	MDC_P014	MDC_P090
#PROPERTY_ NAME.en	Code	Superclass	Class type	Is case of	Applicable properties	Imported properties
	AAA100	UNIVERSE	ITEM_CLASS		{AAE001}	
	AAA200	AAA100	ITEM_CLASS	**{BBA500}**	{AAE002}	**{BBE502,BBE503}**

IEC 2297/13

Légende

Anglais	Français
Code	Code
Superclass	Superclasse
Class type	Type de classe
Is case of	Est cas de
Applicable properties	Propriétés applicables
Imported properties	Propriétés importées

Figure 10 – Mise en œuvre de paquet pour les relations case_of

6.4 Arbre de composition

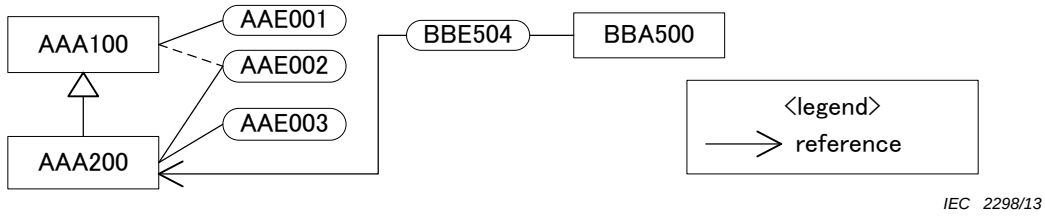
Chaque arbre de composition est composé de relations tout-partie entre les classes. Ce type de structure est souvent présent lorsqu'il est nécessaire de représenter des produits comportant plusieurs parties ou matériaux ou pour associer des informations de gestion du cycle de vie du produit.

Conformément à la CEI 62656-1, une relation tout-partie entre des classes doit être décrite par une propriété définie en CLASS_REFERENCE_TYPE ou CLASS_INSTANCE_TYPE. Les types de données spécifient l'identificateur d'une classe (classe partielle) faisant partie d'une autre classe (classe d'ensemble), mais il existe des différences:

- CLASS_REFERENCE_TYPE est utilisé si la classe partielle de la classe Part (référencée par CLASS_REFERENCE_TYPE) contient des instances indépendantes de la classe d'ensemble. Par exemple, des instances du produit "pneu" peuvent être référencées par plusieurs instances du produit "véhicule". Pour mettre en œuvre les relations entre des instances de "véhicule" et "pneu", CLASS_REFERENCE_TYPE doit être utilisé.
- CLASS_INSTANCE_TYPE est utilisé lorsque les instances de la classe partielle sont intégrées à la classe d'ensemble (c'est-à-dire à l'endroit où réside CLASS_INSTANCE_TYPE). Dans ce cas, si la classe d'ensemble est détruite, ces instances de la classe partielle doivent l'être.

D'un point de vue légèrement différent, dans le cas de CLASS_INSTANCE_TYPE, la classe référencée fait simplement office de générateur de propriété composite, parfois appelée "propriété de type d'objet" dans d'autres ouvrages de référence.

La Figure 11 présente un exemple de relation de composition entre deux branches. À la Figure 11, la classe AAA200 fait partie de la classe BBA500. Pour définir la relation tout-partie entre la classe AAA200 (classe partielle) et la classe BBA500 (classe d'ensemble), la classe BBA500 contient la propriété CLASS_REFERENCE_TYPE BBE504, qui spécifie la classe AAA200.

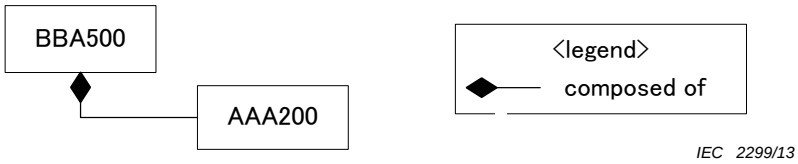


Légende

Anglais	Français
<legend>	<légende>
reference	référence

Figure 11 – Relation de composition entre deux branches

La Figure 12 donne une vue de composition de la classe BBA500 déduite des arbres de classification de la Figure 11. La relation de composition entre les deux classes BBA500 et AAA200 est illustrée par un losange noir et une ligne continue.



IEC 2299/13

Légende

Anglais	Français
<legend>	<légende>
composed of	composé de

Figure 12 – Exemple d'arbre de composition

La Figure 13 présente des feuilles de paquets conjonctifs pour la mise en œuvre de l'arbre de composition illustré à la Figure 11. La propriété BBE504 est énumérée sous la forme d'une propriété applicable à la classe BBA500, afin de décrire une relation tout-partie entre la classe AAA200 (classe partielle) et la classe BBA500 (classe d'ensemble).

#CLASS_ID:= **MDC_C002**				
#CLASS_NAME.en:= **Class meta-class**				
#PROPERTY_ID	MDC_P001_5	MDC_P010	MDC_P011	MDC_P014
#PROPERTY_ NAME.en	Code	Superclass	Class type	Applicable properties
	BBA500	UNIVERSE	ITEM_CLASS	**{BBE504}**

#CLASS_ID:= **MDC_C003**				
#CLASS_NAME.en:= **Property meta-class**				
#PROPERTY_ID	MDC_P001_6	MDC_P020	MDC_P021	MDC_P022
#PROPERTY_ NAME.en	Code	Property data element type	Definition class	Data type
	BBE504	NON_DEPENDENT_ P_DET	BBA500	**CLASS_REFERENCE (AAA200)**

IEC 2300/13

Légende

Anglais	Français
#CLASS_NAME.en:=Class meta-class	#CLASS_NAME.en:= méta-classe de classes
Code	Code
Superclass	Superclasse
Class type	Type de classe
Applicable properties	Propriétés applicables
#CLASS_NAME.en:=Property meta-class	#CLASS_NAME.en:= méta-classe de propriétés
Property data element type	Type d'élément de données de propriétés
Definition class	Classe de définition
Data type	Type de données

Figure 13 – Mise en œuvre de paquet pour les arbres de composition

7 Définition des éléments ontologiques par des paquets facultatifs

7.1 Définition des énumérations

S'il est nécessaire d'attribuer une liste de valeurs possibles à une propriété, cette liste doit être définie comme une énumération dans une feuille de paquet d'énumérations, puis doit être attribuée aux propriétés d'une feuille de paquet de propriétés.

L'attribut MDC_P044 (Liste des codes d'énumération) du paquet d'énumérations permet de formuler une liste de valeurs. Chaque valeur de cette liste doit correspondre au type de données ou à son sous-type. Par exemple, si une liste de valeurs est destinée à des propriétés définies avec un type réel, la liste doit contenir des valeurs réelles.

S'il est requis de définir les significations des valeurs de la liste, chaque valeur doit être définie sous la forme d'un terme dans une feuille de paquet de termes, puis doit être attribuée à des énumérations dans la feuille de paquet d'énumérations.

L'attribut MDC_P025_1 (Symbole littéral préférentiel en texte) du paquet de termes permet de décrire une valeur qui peut être attribuée en tant que valeur réelle des propriétés.

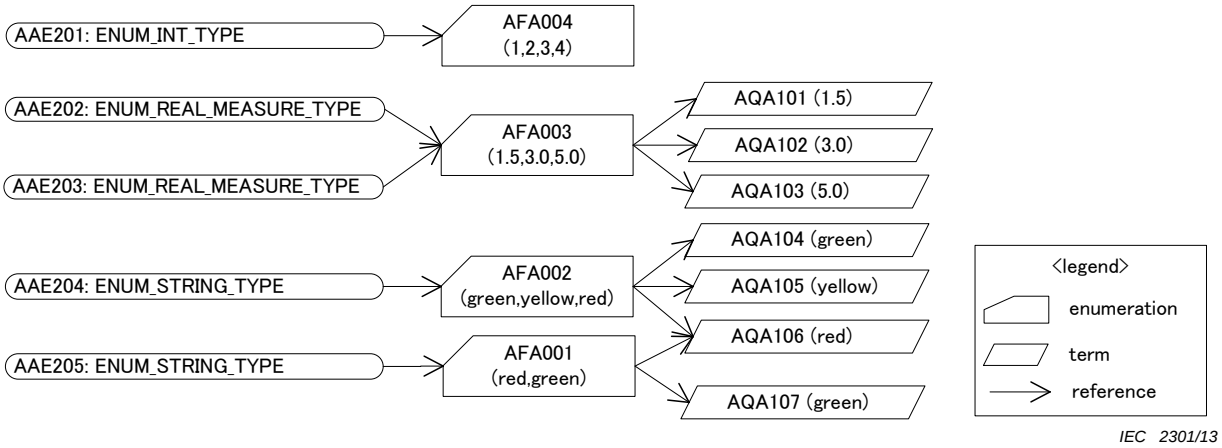
L'attribut MDC_P043 (Liste énumérée de termes) du paquet d'énumérations spécifier une liste d'identificateurs des termes qui peuvent être appliqués aux propriétés comme étant leurs valeurs candidates. Si des identificateurs de termes sont spécifiés, les valeurs décrites dans l'attribut MDC_P044 doivent être une liste de valeurs de l'attribut MDC_P025_1 des termes.

Si les valeurs des attributs MDC_P043 et MDC_P044 sont spécifiées, le nombre d'éléments de MDC_P043 doit être identique à celui des éléments de MDC_P044.

NOTE L'ISO 13584-42 ne précise aucun moyen de définir la signification d'une valeur. Par conséquent, les termes sont ignorés dans le cas des systèmes conformes à l'ISO 13584-42.

La Figure 14 présente un exemple de dictionnaire de données contenant des énumérations. Il existe cinq propriétés, quatre énumérations et sept termes. Chaque propriété est définie en tant que sous-type de ENUM_TYPE qui spécifie une énumération.

Les énumérations AFA001 à AFA003 spécifient des termes définis dans la feuille de paquet de termes. AFA001 spécifie les deux termes AQA106 et AQA107. L'énumération AFA002 spécifie les trois termes AQA104 à AQA106, dans lesquels des valeurs de chaîne sont spécifiées. L'énumération AFA003 spécifie les trois termes AQA101 à AQA103, dans lesquels des valeurs réelles sont spécifiées. L'énumération AFA004 spécifie simplement une liste de quatre valeurs entières, aucun terme n'étant donc spécifié pour chaque valeur. Le terme AQA106 est spécifié par les deux énumérations AFA001 et AFA002, car la signification du terme peut être partagée par ces énumérations. A l'inverse, les deux termes AQA104 et AQA107 ont la même valeur "verte", mais sont définis séparément, car ils indiquent deux sortes de "vert" différentes. L'énumération AFA003 est partagée par les propriétés AAE202 et AAE203.



Légende

Anglais	Français
legend	légende
enumeration	énumération
term	terme
reference	référence
(green,yellow,red)	(vert,jaune,rouge)
green	vert
yellow	jaune
red	Rouge
(red,green)	(rouge,vert)

Figure 14 – Exemple d'un cas d'utilisation d'énumération

La Figure 15 présente des feuilles de paquets conjonctifs permettant de décrire le dictionnaire de données présenté à la Figure 14.

Dans la feuille de paquet de propriétés, présentée comme étant la première feuille, cinq propriétés sont décrites. La colonne de l'attribut MDC_P022 (Type de données) spécifie le type de données de chaque propriété comme étant un sous-type de ENUM_TYPE qui spécifie l'identificateur d'une énumération et sa possible liste de valeurs.

Dans la feuille de paquet d'énumérations, présentée comme étant la deuxième feuille, quatre énumérations sont décrites. L'attribut MDC_P043 (Liste énumérée de termes) spécifie les termes utilisés comme des valeurs candidates pour les propriétés, alors que l'attribut MDC_P044 (Liste des codes d'énumération) spécifie les valeurs réelles des propriétés. L'énumération AFA004 ne contient aucune valeur pour l'attribut MDC_P043. Seules les valeurs candidates sont donc spécifiées.

Dans la feuille de paquet de termes, présentée comme étant la troisième feuille, sept termes sont décrits. Chaque valeur qui peut être sélectionnée comme valeur candidate pour les propriétés est décrite dans l'attribut MDC_P025_1 (Symbole littéral préférentiel en texte).

#CLASS_ID:= **MDC_C003**			
#CLASS_NAME.en:= **Property meta-class**			
#PROPERTY_ID	MDC_P001_6	MDC_P023_1	MDC_P022
#PROPERTY_ NAME.en	Code	Unit in text	Data type
	AAE201		**ENUM_INT_TYPE(AFA004(1,2,3,4))**
	AAE202	m	**ENUM_REAL_MEASURE_TYPE(AFA003(1.5,3.0,5.0))**
	AAE203	m	**ENUM_REAL_MEASURE_TYPE(AFA003(1.5,3.0,5.0))**
	AAE204		**ENUM_STRING_TYPE(AFA002(green,yellow,red))**
	AAE205		**ENUM_STRING_TYPE(AFA001(red,green))**

#CLASS_ID:= **MDC_C005**			
#CLASS_NAME.en:= **Enumeration meta-class**			
#PROPERTY_ID	MDC_P001_12	MDC_P043	MDC_P044
#PROPERTY_ NAME.en	Code	Enumerated list of terms	Enumeration code list
	AFA001	**(AQA106,AQA107)**	**(red,green)**
	AFA002	**(AQA104,AQA105,AQA106)**	**(green,yellow,red)**
	AFA003	**(AQA101,AQA102,AQA103)**	**(1.5,3.0,5.0)**
	AFA004		**(1,2,3,4)**

value candidate

enumeration

#CLASS_ID:= **MDC_C010**		
#CLASS_NAME.en:= **Term meta-class**		
#PROPERTY_ID	MDC_P001_11	MDC_P025_1
#PROPERTY_ NAME.en	Code	Preferred letter symbol
	AQA101	**1.5**
	AQA102	**3.0**
	AQA103	**5.0**
	AQA104	**green**
	AQA105	**yellow**
	AQA106	**red**
	AQA107	**green**

IEC 2302/13

Légende

Anglais	Français
#CLASS_NAME.en:=Property meta-class	#CLASS_NAME.en:= méta-classe de propriétés
Code	Code
Unit in text	Unité en texte
Data type	Type de données
#CLASS_NAME.en:=Enumeration meta-class	#CLASS_NAME.en:= méta-classe d'énumérations
Enumerated list of terms	Liste énumérée de termes
Enumerated code list	Liste énumérée de codes
(red,green)	(rouge,vert)
(gree,yellow,red)	(vert,jaune,rouge)
#CLASS_NAME.en:=Term meta-class	#CLASS_NAME.en:= méta-classe de termes
Preferred letter symbol	Symbole littéral préférentiel
enumeration	énumération

Anglais	Français
green	vert
yellow	jaune
red	Rouge
value candidate	valeur candidate

Figure 15 – Mise en œuvre de paquet pour les énumérations

7.2 Définition des types de données nommés

La CEI 62656-1 contient un certain nombre de types de données, dont la plupart repose sur les types de données définis dans la CEI 61360-2, certains d'entre eux étant cependant étendus. En règle générale, ces types de données prédéfinis suffisent à la modélisation des données spécifiques au domaine. Mais parfois, il est nécessaire de définir de nouveaux types de données portant leurs propres noms, afin, par exemple, de renommer un type de données à partager entre plusieurs propriétés ou de le distinguer explicitement des autres types de données ou du type de données d'origine, avant d'attribuer le nouveau nom. Ces types de données sont appelés "type de données nommé" et doivent être définis en tant qu'instances du paquet de types de données identifié par MDC_C006 (méta-classe de type de données).

Dans la feuille de paquet de types de données, chaque type nommé doit être décrit dans chaque ligne de la section de données. Le type de données de base de chaque type de données nommé doit être un type de données prédéfini de la CEI 62656-1 ou un autre type de données nommé. Il est supposé qu'une référence entre des types de données nommés et une référence imbriquée parmi des types nommés ne doit pas créer une structure en boucle.

A l'instar de la définition d'une propriété, un type de données nommé doit être applicable à une classe dans laquelle il est requis pour la définition d'une propriété ou d'un autre type de données nommé. Ces informations doivent être décrites dans l'attribut MDC_P015 (Types applicables) d'une feuille de paquet de classes. Il est également possible d'importer un type de données nommé à partir d'une classe appartenant à une autre branche. Dans ce cas, l'attribut MDC_P091 (Types importés) de la feuille de paquet de classes doit être utilisé pour spécifier un ensemble de types de données nommés à utiliser, l'attribut MDC_P013 (Is case of) doit l'être pour spécifier les classes à partir desquelles ces types de données nommés sont importés.

Les propriétés et types de données nommés définis dans une classe peuvent utiliser l'un des types de données nommés applicables à la classe au moyen du type de données prédéfini NAMED_TYPE dans leurs zones de l'attribut MDC_P022 (Type de données).

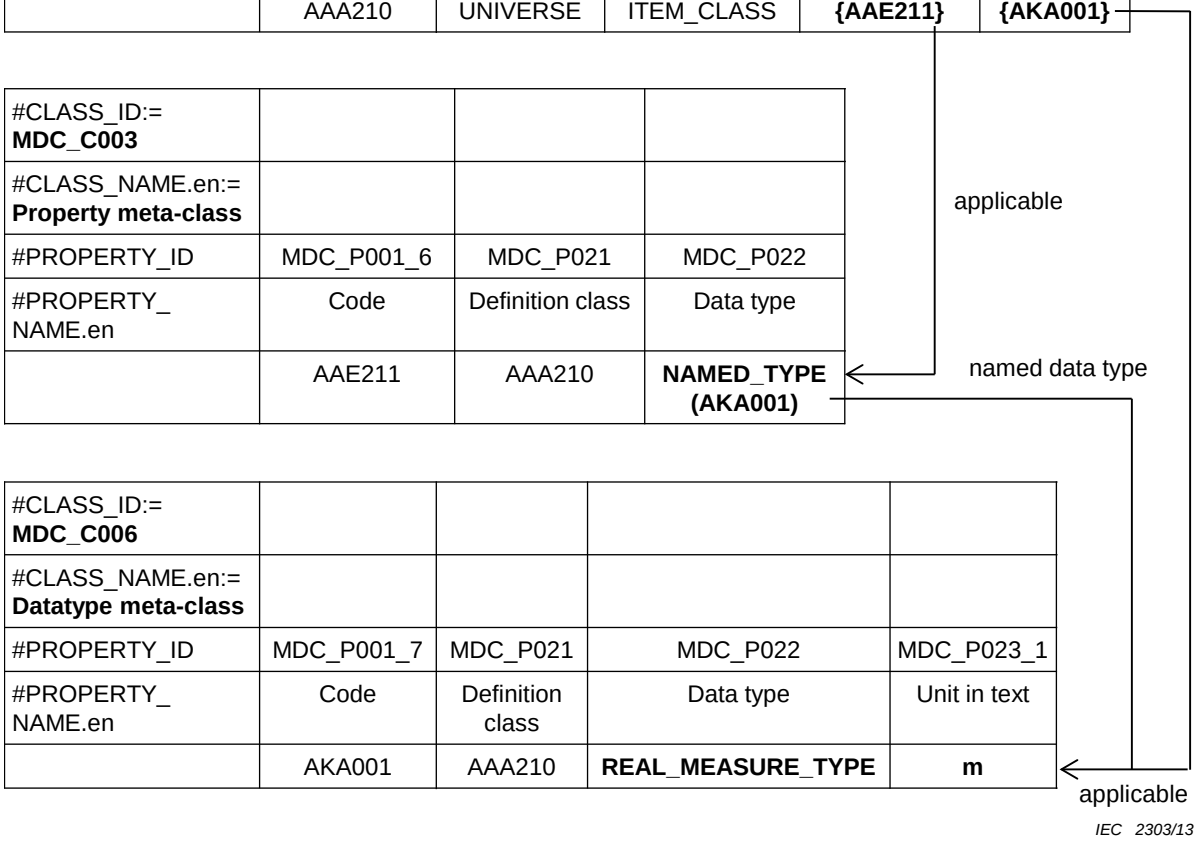
Noter que si un type de données nommé est l'un des types de mesure ou des types de monnaie, il doit respectivement comporter une unité de mesure ou de monnaie. D'autre part, une propriété et un autre type de données nommé défini comme un type de données nommé doivent comporter la même unité de mesure ou de monnaie que le type de données nommé. Si ces informations ne sont pas spécifiées dans une propriété ou un autre type de données nommé, les informations du type de données nommé référencé sont implicitement appliquées à la propriété ou au type de données nommé.

La Figure 16 illustre un exemple de feuilles de paquets conjonctifs pour décrire un dictionnaire de données, contenant une propriété définie en tant que type de données nommé. Dans cet exemple, une propriété AAE211 est définie comme un type de données nommé AKA001, qui repose sur le type de données prédéfini REAL_MEASURE_TYPE. Par conséquent, le type de données nommé est défini dans la feuille de paquet de types de données en bas de la figure, et l'identificateur du type de données nommé est spécifié dans le champ de type de données de la propriété grâce au mot-clé "NAMED_TYPE". Bien que le type de données nommé soit défini en tant que REAL_MEASURE_TYPE, aucune unité de mesure n'est spécifiée pour la propriété AAE211. Dans ce cas, l'unité de mesure "m" du type de données nommé AKA001 est appliquée à la propriété AAE211 comme unité de mesure.

#CLASS_ID:= **MDC_C002**					
#CLASS_NAME.en:= **Class meta-class**					
#PROPERTY_ID	MDC_P001_5	MDC_P010	MDC_P011	MDC_P014	MDC_P015
#PROPERTY_ NAME.en	Code	Superclass	Class type	Applicable properties	Applicable types
	AAA210	UNIVERSE	ITEM_CLASS	**{AAE211}**	**{AKA001}**

#CLASS_ID:= **MDC_C003**			
#CLASS_NAME.en:= **Property meta-class**			
#PROPERTY_ID	MDC_P001_6	MDC_P021	MDC_P022
#PROPERTY_ NAME.en	Code	Definition class	Data type
	AAE211	AAA210	**NAMED_TYPE (AKA001)**

#CLASS_ID:= **MDC_C006**				
#CLASS_NAME.en:= **Datatype meta-class**				
#PROPERTY_ID	MDC_P001_7	MDC_P021	MDC_P022	MDC_P023_1
#PROPERTY_ NAME.en	Code	Definition class	Data type	Unit in text
	AKA001	AAA210	**REAL_MEASURE_TYPE**	**m**



Légende

Anglais	Français
#CLASS_NAME.en:=Class meta-class	#CLASS_NAME.en:= méta-classe de classes
Code	Code
Superclass	Superclasse
Class type	Type de classe
Applicable properties	Propriétés applicable
Applicable types	Types applicables
#CLASS_NAME.en:=Property meta-class	#CLASS_NAME.en:= méta-classe de propriétés
applicable	applicable
Definition class	Classe de définition
Data type	Type de données
named data type	type de données nommé
#CLASS_NAME.en:=Datatype meta-class	#CLASS_NAME.en:= méta-classe de types de données
Unit in text	Unité en texte

Figure 16 – Mise en œuvre de paquet pour les types de données nommés

7.3 Définition des informations des ressources externes

Il existe des ressources à l'extérieur des paquets de données (documents binaires, fichiers images, audio et vidéo, par exemple). Pour définir et spécifier ces ressources, le paquet de documents identifié sous la forme MDC_C007 (méta-classe document) est utilisé.

Dans une feuille de paquet de documents, les informations relatives à chaque ressource externe sont décrites dans chaque ligne. Ces informations incluent les caractéristiques de l'organisme ayant la responsabilité d'un document, l'accès à un document, etc.

A l'instar de la définition de propriété, les informations de ressources doivent s'appliquer à une classe dans laquelle elles sont requises. Ces informations doivent être décrites dans l'attribut MDC_P094 (Documents applicables) d'une feuille de paquet de classes. Il est également possible d'importer des documents à partir d'une classe dans une autre branche. Dans ce cas, l'attribut MDC_P093 (Documents importés) de la feuille de paquet de classes doit être utilisé pour spécifier un ensemble de documents à utiliser, l'attribut MDC_P013 (Is case of) devant l'être pour spécifier les classes à partir desquelles ces documents sont importés.

Dans d'autres feuilles de paquet, les attributs prédéfinis MDC_P004_4 (Icône de nom), MDC_P008_1 (Dessin simplifié) et MDC_P008_2 (Graphisme) sont censés comporter l'identificateur d'un document défini dans la feuille de paquet de documents.

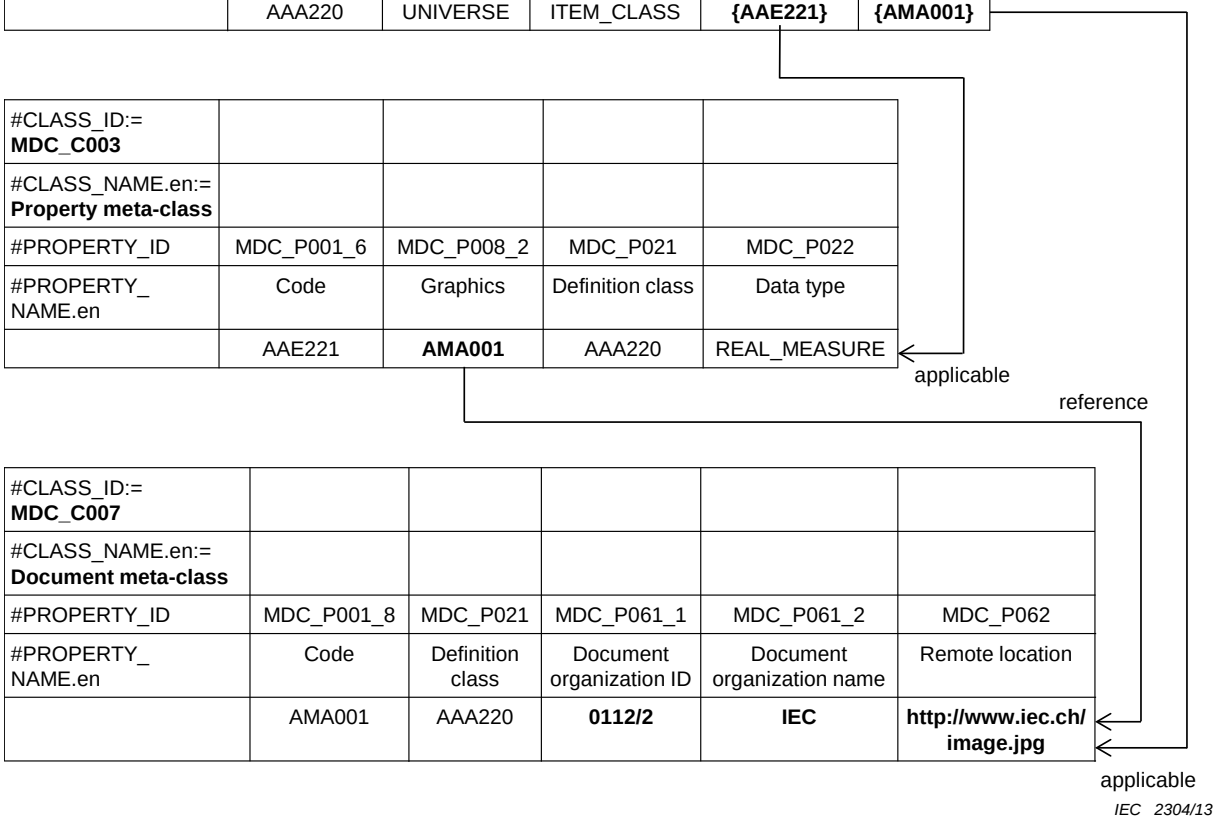
La Figure 17 illustre un exemple de feuilles de paquets conjonctifs pour la description d'un dictionnaire de données, contenant les informations d'un fichier externe référencé à partir d'une propriété. Les informations du fichier externe accessibles à l'adresse "http://iec.ch/image.jpg" sont identifiées en tant que document AMA001 et décrites dans la feuille de paquet de documents en bas de la figure. Dans la feuille de paquet de propriétés au milieu de la figure, la propriété AAE221 dont le graphisme est déduit de l'URL spécifiée est décrite. Dans la feuille de paquet de classes en haut de la figure, la propriété AAE221 et le document AMA001 deviennent applicables à la classe AAA220

NOTE <http://iec.ch/image.jpg> est utilisé comme un exemple et est fictif.

#CLASS_ID:= **MDC_C002**					
#CLASS_NAME.en:= **Class meta-class**					
#PROPERTY_ID	MDC_P001_5	MDC_P010	MDC_P011	MDC_P014	MDC_P094
#PROPERTY_ NAME.en	Code	Superclass	Class type	Applicable properties	Applicable documents
	AAA220	UNIVERSE	ITEM_CLASS	**{AAE221}**	**{AMA001}**

#CLASS_ID:= **MDC_C003**				
#CLASS_NAME.en:= **Property meta-class**				
#PROPERTY_ID	MDC_P001_6	MDC_P008_2	MDC_P021	MDC_P022
#PROPERTY_ NAME.en	Code	Graphics	Definition class	Data type
	AAE221	**AMA001**	AAA220	REAL_MEASURE

#CLASS_ID:= **MDC_C007**					
#CLASS_NAME.en:= **Document meta-class**					
#PROPERTY_ID	MDC_P001_8	MDC_P021	MDC_P061_1	MDC_P061_2	MDC_P062
#PROPERTY_ NAME.en	Code	Definition class	Document organization ID	Document organization name	Remote location
	AMA001	AAA220	**0112/2**	**IEC**	**http://www.iec.ch/ image.jpg**



Légende

Anglais	Français
#CLASS_NAME.en:=Class meta-class	#CLASS_NAME.en:= méta-classe de classes
Code	Code
Superclass	Superclasse
Class type	Type de classe
Applicable properties	Propriétés applicables
Applicable documents	Documents applicables
#CLASS_NAME.en:=Property meta-class	#CLASS_NAME.en:= méta-classe de propriétés
Graphics	Graphisme
Definition class	Classe de définition
Data type	Type de données
applicable	applicable
reference	référence
#CLASS_NAME.en:=Document meta-class	#CLASS_NAME.en:= méta-classe de documents
Document organization ID	Identificateur du document d'organisation
Document organization name	Nom du document d'organisation
IEC	CEI
Remote location	Emplacement distant

Figure 17 – Mise en œuvre de paquet pour les références de document

7.4 Définition des unités de mesure

Le moyen le plus simple de spécifier l'unité d'une propriété consiste à la décrire dans les attributs MDC_P023 (Structure d'unité), MDC_P023_1 (Unité en texte) et MDC_P023_2 (Unité en SGML). L'attribut MDC_P023 est destiné à une expression STEP d'une unité [6]. Ensuite, l'attribut MDC_P023_1 concerne une expression textuelle d'une unité. Enfin, l'attribut MDC_P023_2 concerne une expression SGML [7] d'une unité, y compris l'expression MathML [8].

Outre le simple moyen d'expression d'unité ci-dessus, il existe des cas plus complexes dans lesquels il convient d'identifier les unités, par exemple:

– définition non seulement des unités SI ou non SI simples, mais également de leur combinaison,
– distinction entre les unités si elles sont décrites à l'aide de la même unité de référence ou du même ensemble d'unités, mais elles peuvent avoir des significations différentes selon les domaines dans lesquels elles peuvent être appliquées,
– conversion informatique de la valeur d'une unité vers sa valeur alternative.

La CEI 62720, qui contient un dictionnaire d'unités pour l'ensemble des domaines électrique et électronique, est un bon exemple de l'exigence présentée ci-dessus.

Dans la CEI 62656-1, le paquet d'unités de mesure (paquet UoM pour faire court), identifié sous la forme MDC_C009 (méta-classe UoM) permet de définir des unités de mesure. La feuille de paquet UoM contient les trois attributs MDC_P023, MDC_P023_1 et MDC_P023_2 de description des informations d'une unité de mesure réellement utilisée par les propriétés et les types de données nommés. Ce type d'unité est identifié par l'attribut MDC_P001_10 (Code), qui spécifie un identificateur de l'unité définie.

Une unité définie en tant qu'instance d'un paquet UoM peut être utilisée par des propriétés ou des types de données nommés grâce aux attributs MDC_P041 (Code pour l'unité) et MDC_P042 (Codes pour les unités alternatives) de ces paquets. L'attribut MDC_P042 peut comporter un ensemble d'identificateurs d'unités qui sont des alternatives à une unité principale décrite dans l'attribut MDC_P041. Cela signifie que l'attribut MDC_P041 doit avoir une valeur à chaque fois que l'attribut MDC_P042 en a une.

La Figure 18 présente des feuilles de paquets conjonctifs composées d'un paquet de propriétés et d'un paquet UoM. Dans la feuille de paquet UoM sont décrites quatre unités de mesure qui reposent sur le contenu défini dans la CEI 62720. La propriété AAE231 spécifie "m" comme unité représentée sous forme de texte. Une propriété AAE232 spécifie une unité de mesure UAA726 définie dans la feuille de paquet UoM. Une propriété AAE233 spécifie une unité en texte et une unité de mesure définies dans la feuille de paquet UoM. Par conséquent, l'unité en représentation texte l'emporte sur la propriété AAE233. La propriété AAE233 spécifie également un ensemble d'unités alternatives UAB197 et UAA917 pour son unité de mesure principale UAA594.

Dans la feuille de paquet UoM, chaque unité de mesure définit son unité en représentation texte dans l'attribut MDC_P023_1 et la description MathML dans l'attribut MDC_P023_2.

#CLASS_ID:= **MDC_C003**					
#CLASS_NAME.en:= **Property meta-class**					
#PROPERTY_ID	MDC_P001_6	MDC_P022	MDC_P023_1	MDC_P041	MDC_P042
#PROPERTY_ NAME.en	Code	Data type	Unit in text	Code for unit	Codes for alternative units
	AAE231	**REAL_MEASURE**	m		
	AAE232	**REAL_MEASURE**		UAA726	
	AAE233	**LEVEL(MIN,MAX) OF REAL_MEASURE**	kg	UAA594	{UAB197,UAA917}

#CLASS_ID:= **MDC_C009**				
#CLASS_NAME.en:= **UoM meta-class**				
#PROPERTY_ID	MDC_P001_10	MDC_P004_1.en	MDC_P023_1	MDC_P023_2
#PROPERTY_ NAME.en	Code	Preferred name	Unit in text	Unit in SGML
	UAA726	metre	**m**	**<math xmlns="http://www.w3.org/1998/Math/MathML" display="block"><mrow><mrow><mi>m</mi></mrow></mrow></math>**
	UAA594	kilogram	**kg**	**<math xmlns="http://www.w3.org/1998/Math/MathML" display="block"><mrow><mrow><mi>k</mi><mi>g</mi></mrow></mrow></math>**
	UAB197	Pound (US)	**lb**	**<math xmlns="http://www.w3.org/1998/Math/MathML" display="block"><mrow><mrow><mi>lb</mi><mspace width="0.3em"/> <mfenced><mrow><mi>US</mi></mrow></mfenced></mrow></mrow></math>**
	UAA917	Ounce	**oz**	**<math xmlns="http://www.w3.org/1998/Math/MathML" display="block"><mrow><mi>oz</mi></mrow></math>**

primary unit

alternative units

IEC 2305/13

Légende

Anglais	Français
#CLASS_NAME.en:=Property meta-class	#CLASS_NAME.en:= méta-classe de propriétés
Code	Code
Data type	Type de données
Unit in text	Unité en texte
Code for unit	Code pour l'unité
Codes for alternative units	Codes pour les unités alternatives
#CLASS_NAME.en:=UoM meta-class	#CLASS_NAME.en:= méta-classe d'unités de mesure
Preferred name	Nom préférentiel
Unit in SGML	Unité en SGML
metre	mètre
kilogram	kilogramme
Pound (US)	Livre (États-Unis)
Ounce	Once

Anglais	Français
primary unit	unité primaire
alternative units	unités alternatives

Figure 18 – Mise en œuvre de paquet pour l'unité de mesure

7.5 Définition des relations entre les éléments ontologiques

Dans la CEI 62656-1, différents types d'attribut sont spécifiés pour définir les relations entre des éléments ontologiques. Par exemple, MDC_P010 (Superclasse) est l'attribut de description de relations is-a entre des classes. Un autre exemple est l'attribut MDC_P014 (Propriétés applicables) qui permet de décrire une relation entre une classe et une propriété qui devient applicable à la classe. Outre les attributs prédéfinis, il est nécessaire de définir un type de relation entre les éléments ontologiques. Un cas d'utilisation classique est un groupement logique de propriétés qui peuvent être largement utilisées pour différentes applications. Dans la CEI 62656-1, ce type de relation peut être mis en œuvre par les relations définies dans une feuille de paquet de relations.

Chaque relation est de type "prédicat" ou "fonctionnel" selon l'objectif de la relation. Ce type est spécifié dans l'attribut MDC_P200 (Type de relation) d'une feuille de paquet de relations. Si une relation est définie comme étant de type fonctionnel, le mot-clé "FUNCTION" ou son abréviation "FUNC" doit être spécifié dans l'attribut MDC_P200, ses domaine et co-domaine doivent être respectivement spécifiés dans les attributs MDC_P202 (Domaine de la fonction) et MDC_P203 (Co-domaine de la fonction). Sinon, le mot-clé "PREDICATION" ou son abréviation "PRED" doit être spécifié dans l'attribut MDC_P200, son domaine doit être spécifié dans l'attribut MDC_P201 (Domaine de la relation).

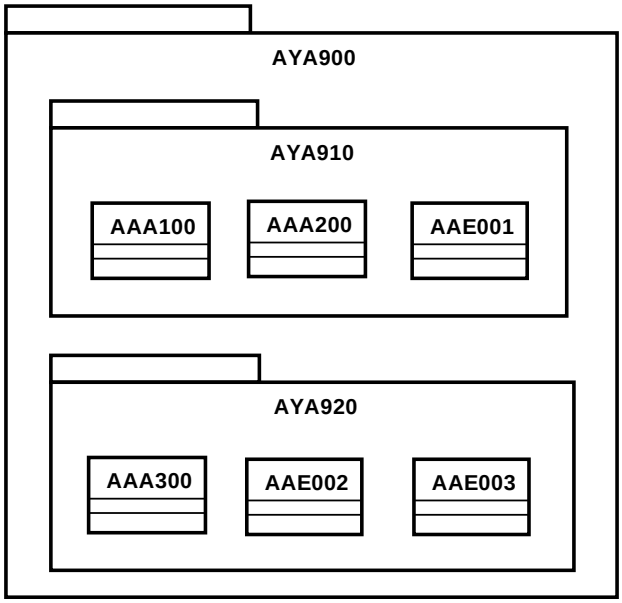
L'attribut MDC_P208 (Type de domaine) est également disponible, si le type de tous les éléments ontologiques contenus dans le domaine d'une relation est identique à celui du paquet. Dans ce cas, l'identificateur de ce type de paquet doit être spécifié en tant que valeur de l'attribut. Sinon, la valeur de l'attribut MDC_P208 doit être vierge.

NOTE 1 Chaque relation peut spécifier une autre relation comme étant (un membre de) son domaine.

NOTE 2 L'utilisation de la relation et les éléments ontologiques spécifiés dépendent de chaque application.

Le langage de modélisation unifié (UML, "Unified Modeling Language") [9], qui est un langage de modélisation *de facto* dans les domaines de la programmation orientée objet définie par l'OMG (Object Management Group), offre un mécanisme de paquetage permettant de regrouper les objets de manière logique (les classes et cas d'utilisation, par exemple) dans un paquetage. Un paquetage peut en contenir un autre (sous-paquetage) afin de concevoir une structure hiérarchique imbriquée.

La Figure 19 présente un exemple de schéma de paquetage UML composé de trois paquetages. Le paquetage AYA900 contient les autres paquetages AYA910 et AYA920. Le paquetage AYA910 contient les deux classes AAA100 et AAA200 et la propriété AAE001, le paquetage AYA920 contenant la classe AAA300 et les deux propriétés AAE002 et AAE003.



IEC 2306/13

Figure 19 – Diagramme de paquetage UML par relations

La Figure 20 présente une feuille de paquet de relations sous la forme d'un exemple de mise en œuvre du paquetage UML décrit à la Figure 19 par des relations de prédicat. Dans cet exemple, les identificateurs des classes et propriétés des paquetages AYA910 et AYA920 sont simplement spécifiés dans l'attribut MDC_P201 de chaque paquetage. Les paquetages AYA910 et AYA920 sont spécifiés dans l'attribut MDC_P201 du paquetage AYA900, afin de représenter la relation imbriquée entre les paquetages.

#CLASS_ID:= **MDC_C011**					
#CLASS_NAME.en:= **Relation meta-class**					
#PROPERTY_ID	MDC_P001_13	MDC_P004_1.en	**MDC_P200**	**MDC_P201**	**MDC_P210**
#PROPERTY_ NAME.en	Code	Preferred name	Relation type	Domain of the relation	Role of the relation
	AYA900	Package 1	**PRED**	**{AYA910,AYA920}**	**package**
	AYA910	Package 2	**PRED**	**{AAA100,AAA200,AAE001}**	**package**
	AYA920	Package 3	**PRED**	**{AAA300,AAE002,AAE003}**	**package**

IEC 2307/13

Légende

Anglais	Français
#CLASS_NAME.en:=Relation meta-class	#CLASS_NAME.en:= méta-classe de relations
Code	Code
Preferred name	Nom préférentiel
Relation type	Type de relation
Domain of the relation	Domaine de la relation
Role of the relation	Rôle de la relation
Package 1	Paquetage 1
Package 2	Paquetage 2
Package 3	Paquetage 3

Figure 20 – Mise en œuvre d'un paquet de paquetages UML par des relations de prédicat

La Figure 21 présente un autre exemple de schéma de paquetage UML représentant une structure de données analogue à celle de la Figure 19, mais selon une approche différente. La différence avec la Figure 19 est que les quatre paquetages supplémentaires AYA911, AYA912, AYA921 et AYA922 sont définis pour catégoriser les éléments ontologiques correspondant à la relation pour chaque paquet. Par exemple, le paquetage AYA911 contient les deux classes AAA100 et AAA200 qui représentent le type de paquet de classes. La propriété AAE001 est le type de paquet de propriétés. Par conséquent, la propriété AAE001 est contenue dans l'autre relation, AYA912.

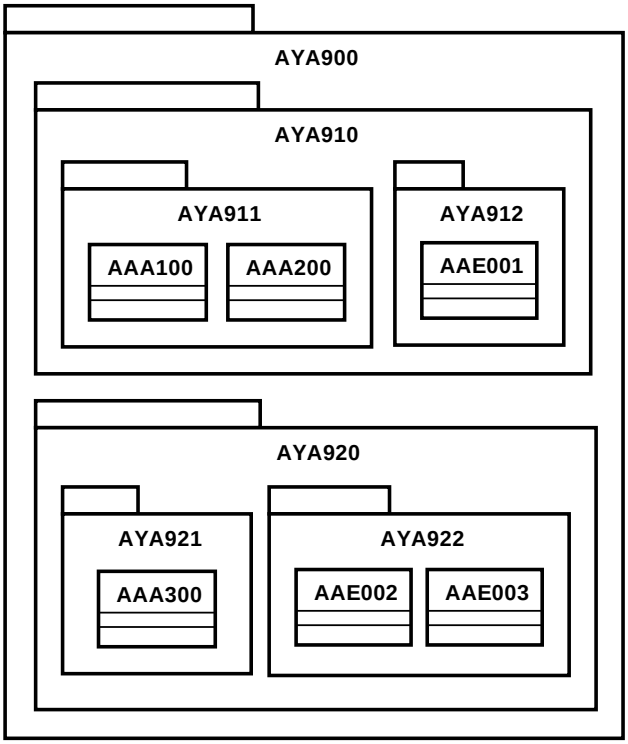


Figure 21 – Diagramme de paquetage UML par fonctions

La Figure 22 présente une feuille de paquet de relations proposant un exemple de mise en œuvre du paquetage UML décrit à la Figure 21. Dans cet exemple, les sept relations AYA900 à AYA922 sont définies pour contenir des instances de chaque paquet. Par exemple, la relation AYA911 spécifie effectivement les deux classes AAA100 et AAA200, qui sont des membres de la relation AYA910. Chaque relation spécifie le type de domaine dans sa valeur de l'attribut MDC_P208 pour faciliter le calcul. Par exemple, la relation AYA911 contient uniquement des classes dans son domaine. Par conséquent, “MDC_C002”, qui est l'identificateur du paquet de classes, est spécifié dans sa valeur de l'attribut MDC_P208.

Un paquetage contenu dans un autre paquetage est spécifié par l'attribut MDC_P203 de ce dernier. Par exemple, la relation AYA900 est spécifiée par les deux relations AYA910 et AYA920 grâce à cet attribut.

#CLASS_ID:= **MDC_C011**							
#CLASS_NAME.en:= **Relation meta-class**							
#PROPERTY_ID	MDC_ P001_13	MDC_P004_1.en	**MDC_P200**	**MDC_P202**	**MDC_P203**	**MDC_P208**	**MDC_P210**
#PROPERTY_ NAME.en	Code	Preferred name	Relation type	Domain of the function	Codomain of the function	Type of the domain	Role of the relation
	AYA900	Package 1	**FUNC**	**{AYA910,AYA920}**	**UNIVERSE**	**MDC_C011**	**package**
	AYA910	Package 2	**FUNC**	**{AYA911,AYA912}**	**AYA900**	**MDC_C011**	**package**
	AYA920	Package 3	**FUNC**	**{AYA921,AYA922}**	**AYA900**	**MDC_C011**	**package**
	AYA911	classes contained in Package 2	**FUNC**	**{AAA100,AAA200}**	**AYA910**	**MDC_C002**	**package**
	AYA912	properties contained in Package 2	**FUNC**	**{AAE001}**	**AYA910**	**MDC_C003**	**package**
	AYA921	classes contained in Package 3	**FUNC**	**{AAA300}**	**AYA920**	**MDC_C002**	**package**
	AYA922	properties contained in Package 3	**FUNC**	**{AAE002,AAE003}**	**AYA920**	**MDC_C003**	**package**

IEC 2309/13

Légende

Anglais	Français
#CLASS_NAME.en:=Relation meta-class	#CLASS_NAME.en:= méta-classe de relations
Code	Code
Preferred name	Nom préférentiel
Relation type	Type de relation
Domain of the function	Domaine de la fonction
Codomain of the function	Co-domaine de la fonction
Type of the domain	Type du domaine
Role of the relation	Rôle de la relation
Package 1	Paquetage 1
Package 2	Paquetage 2
Package 3	Paquetage 3
classes contained in Package 2	classes contenues dans le Paquetage 2
properties contained in Package 2	propriétés contenues dans le Paquetage 2
classes contained in Package 3	classes contenues dans le Paquetage 3
properties contained in Package 3	propriétés contenues dans le Paquetage 3

Figure 22 – Mise en œuvre de paquet des paquetages UML par fonctions

La première approche décrite à la Figure 19 est simple, mais chaque paquetage peut contenir des instances de paquets hétérogènes. En d'autres termes, aucune information fournie dans la feuille de paquet de relations ne permet de savoir à quel paquet appartient chaque instance. Par conséquent, pour reconstituer les informations détaillées de chaque élément ontologique d'un paquetage, il est nécessaire de rechercher chaque paquet. Dans ce cas, au fur et à mesure de l'augmentation du nombre d'éléments ontologiques, les exigences de traitement sont de plus en plus importantes. À l'inverse, la dernière approche décrite à la Figure 21 est complexe, mais permet de déterminer aisément la structure de paquetage et de savoir à quel paquet appartient chaque élément ontologique. Du point de vue des performances d'une application logicielle, la CEI 62656-3 introduit la dernière mise en œuvre pour décrire la structure de paquetage du modèle CIM ("Common Information Model" – "Modèle d'informations commun") définie dans la CEI 61970-301.

Noter que la mise en œuvre de ce type de structure ne se limite pas aux exemples présentés à la Figure 20 et à la Figure 22. Si une autre utilisation du paquet de relations est possible et nécessaire pour une application particulière, elle devrait être improvisée dans l'application.

8 Concepts avancés

8.1 Mise en œuvre de la condition

Une propriété de condition est une propriété qui peut affecter la décision de valeur d'une autre propriété (voir CEI 61360-1).

Un cas d'utilisation classique de ce type de propriété consiste à définir un environnement extérieur (température et humidité, par exemple) pour les valeurs de mesure. Un autre cas d'utilisation classique consiste à offrir un moyen de configuration explicite du nombre d'emplacements d'un automate programmable au niveau de la couche Bibliothèques de domaines (voir 8.2 à 8.4).

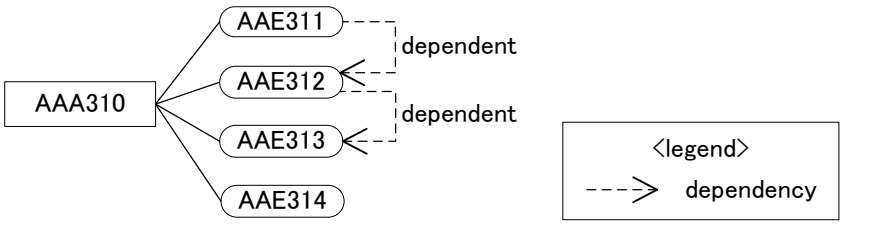
L'attribut MDC_P020 (Type d'élément de données de propriétés) du paquet de propriétés permet de spécifier la dépendance d'une propriété, à savoir i) si cette propriété est une propriété de condition pour une autre propriété, et ii) si cette propriété dépend d'une autre propriété. Le Tableau 1 présente la valeur qui doit être sélectionnée pour chaque propriété dans chacune des combinaisons des cas i) et ii) ci-dessus.

Tableau 1 – Type d'élément de données de propriétés pour la condition

condition pour une autre propriété	dépend d'une autre propriété	type à attribuer
		NON_DEPENDENT_P_DET
✓		CONDITION_DET
	✓	DEPENDENT_P_DET
✓	✓	DEPENDENT_C_DET

Si une propriété dépend d'une ou de plusieurs autres propriétés, DEPENDENT_P_DET ou DEPENDENT_C_DET est attribué à la première. Les identificateurs des dernières propriétés doivent être spécifiés dans l'attribut MDC_P028 (Condition) du paquet de propriétés.

La Figure 23 présente un exemple de dictionnaire de données contenant des propriétés de condition dont la classe AAA310 contient quatre propriétés. La propriété AAE311 dépend de la propriété AAE312, laquelle dépend également de la propriété AAE313.



IEC 2310/13

Légende

Anglais	Français
dependent	dépendant
legend	légende
dependency	dépendance

Figure 23 – Exemple de condition

La Figure 24 présente un exemple de feuille de paquet de propriétés décrivant les propriétés présentées à la Figure 23. La propriété AAE311 dépend de la propriété AAE312. Par conséquent, la propriété AAE311 est définie comme DEPENDENT_P_DET et sa propriété de condition est énumérée dans l'attribut MDC_P028. La propriété AAE312 est la propriété de condition de la propriété AAE311 et dépend de la propriété AAE313. Par conséquent, la propriété AAE312 est définie comme DEPENDENT_C_DET. Ensuite, la propriété AAE313 ne comporte aucune propriété de condition. Elle est donc définie comme CONDITION_DET. Enfin, la propriété AAE314 n'est pas une propriété de condition pour les autres et ne dépend pas des autres propriétés, elle est donc définie comme NON_DEPENDENT_P_DET.

#CLASS_ID:= **MDC_C003**					
#CLASS_NAME.en:= **Property meta-class**					
#PROPERTY_ID	MDC_P001_6	MDC_P020	MDC_P022	MDC_P023_1	MDC_P028
#PROPERTY_ NAME.en	Code	Property data element type	Data type	Unit in text	Condition
	AAE311	**DEPENDENT_P_DET**	REAL_MEASURE	M	**{AAE312}**
	AAE312	**DEPENDENT_C_DET**	REAL_MEASURE	°C	**{AAE313}**
	AAE313	**CONDITION_DET**	REAL_MEASURE	1	
	AAE314	**NON_DEPENDENT_P_DET**	INT		

dependent

dependent

IEC 2311/13

Légende

Anglais	Français
#CLASS_NAME.en:=Property meta-class	#CLASS_NAME.en:= méta-classe de propriétés
Code	Code
Property data element type	Type d'élément de données de propriétés
Data type	Type de données
Unit in text	Unité en texte
Condition	Condition
dependent	dépendant

Figure 24 – Mise en œuvre de paquet pour la condition

8.2 Mise en œuvre de la cardinalité

La cardinalité consiste à limiter le nombre d'occurrences d'une valeur de propriété, qui inclut des composants, des pièces et des matériaux. Notamment, elle définit le nombre minimal et maximal de la collection. Une fonction de cette propriété est que les éléments hétérogènes peuvent apparaître dans la collection.

Une bicyclette est un exemple classique. Elle est dotée de deux pneus, un à l'avant et l'autre à l'arrière d'un cadre. Un autre exemple est celui d'un ordinateur doté de trois ports USB intégrés qui peuvent prendre en charge différents appareils (une souris optique, un disque à mémoire flash USB et une webcam, connectés en même temps, par exemple).

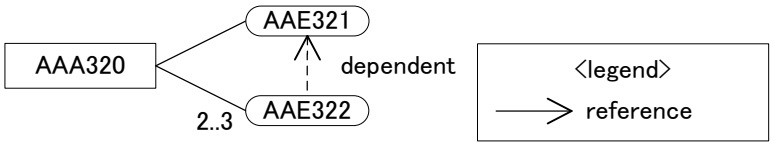
Pour décrire la cardinalité, LIST est le type de données le plus recommandé, car, dans la plupart des cas, l'ordre des éléments est reconnu de manière implicite ou explicite. Si l'ordre des éléments n'est pas réellement nécessaire, BAG peut également être utilisé.

Les nombres minimum et maximum doivent être spécifiés comme étant les arguments de ces types de données. Par exemple, si ces nombres pour une propriété sont respectivement 2 et 3, la propriété est définie en tant que "LIST(2,3)".

En utilisant une propriété de condition (propriété d'interface) pour ce type de propriété, qui indique explicitement le nombre d'emplacements de cette dernière, il est possible de contrôler son interface. Si ce type de propriété n'est pas requis, cette propriété peut être omise.

NOTE Si les paramètres “minimum” et “maximum” sont omis, ils sont implicitement reconnus comme "0" et "?", ce dernier indiquant un nombre illimité.

La Figure 25 présente un exemple de mise en œuvre de cardinalité en incluant une propriété d'interface. La propriété AAE322 de la classe AAA320 est censée contenir deux ou trois valeurs. Dans cet exemple, la propriété AAE321 est également attribuée à la classe AAA320, pour indiquer le nombre réel d'emplacements pour la propriété AAE322.



IEC 2312/13

Légende

Anglais	Français
dependent	dépendant
<legend>	<légende>
reference	référence

Figure 25 – Exemple de cardinalité

La Figure 26 présente un exemple de feuille de paquet de propriétés décrivant les deux propriétés présentées à la Figure 25. Pour la propriété AAE322, les nombres minimal et maximal d'occurrences sont spécifiés comme des arguments de LIST dans la valeur de l'attribut MDC_P022 ("LIST(2,3) OF REAL_MEASURE_TYPE", par exemple). La propriété AAE321 est définie comme INT_TYPE pour spécifier le nombre d'occurrences des valeurs de la propriété AAE322 au niveau de la couche Bibliothèques de domaines. La relation entre ces deux propriétés est définie dans la valeur de l'attribut MDC_P028 pour la propriété AAE322, qui dépend de la propriété AAE321.

#CLASS_ID:= **MDC_C003**					
#CLASS_NAME.en:= **Property meta-class**					
#PROPERTY_ID	MDC_P001_6	MDC_P020	MDC_P022	MDC_P023_1	MDC_P028
#PROPERTY_ NAME.en	Code	Property data element type	Data type	Unit in text	Condition
	AAE321	**CONDITION_DET**	**INT_TYPE**		
	AAE322	**DEPENDENT_P_DET**	**LIST(2,3) OF REAL_MEASURE_TYPE**	**m**	**{AAE321}**

IEC 2313/13

Légende

Anglais	Français
#CLASS_NAME.en:=Property meta-class	#CLASS_NAME.en:= méta-classe de propriétés
Code	Code
Property data element type	Type d'élément de données d’une propriété
Data type	Type de donnée
Unit in text	Unité en texte
Condition	Condition

Figure 26 – Mise en œuvre de paquet pour la cardinalité

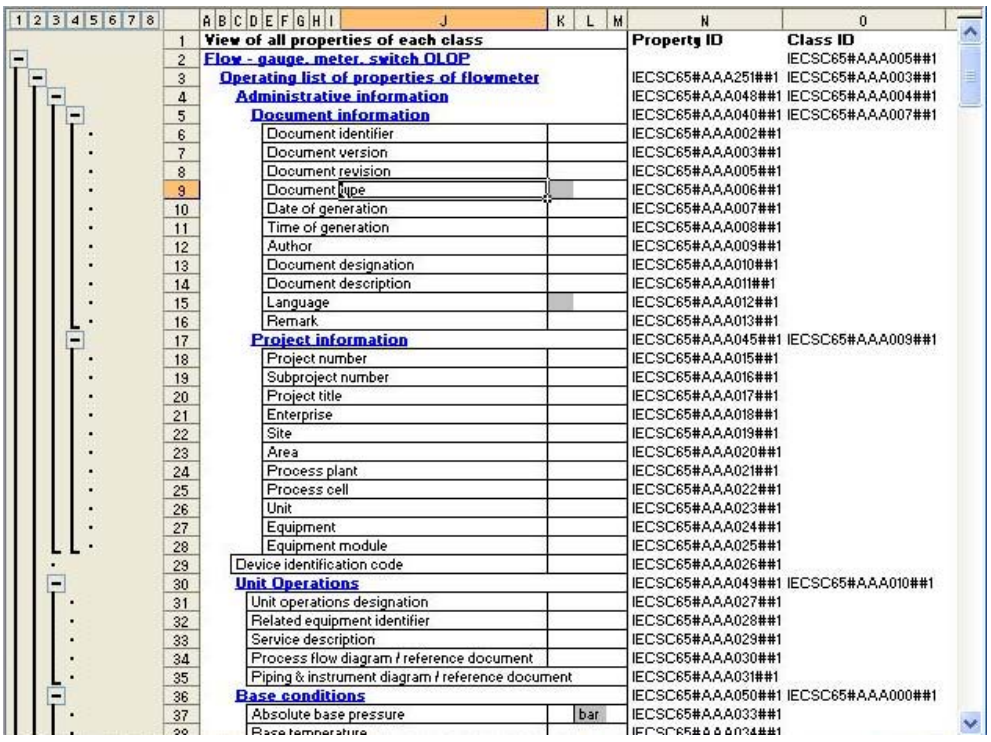
8.3 Mise en œuvre de blocs et de listes de propriétés (LOP)

Le bloc et la liste de propriétés (LOP – List Of Properties) font partie d'un mécanisme défini à l'origine dans la CEI 61987-10 et permettant de regrouper librement les propriétés. Dans le modèle de dictionnaire commun ISO 13584-42/CEI 61360-2, chaque ensemble bloc et LOP

est mis en œuvre comme une classe de caractéristiques, à laquelle ses groupes de propriétés sont applicables. Chaque ensemble bloc et LOP peut contenir un ou plusieurs blocs par la propriété CLASS_REFERENCE_TYPE ou CLASS_INSTANCE_TYPE, conformément à la relation "has-a" expliquée en 6.4, c'est-à-dire que la structure imbriquée de blocs et de LOP est autorisée.

La cardinalité, présentée en 8.2, est également utilisée pour indiquer la répétition des blocs de propriétés ou des LOP. Cela signifie que la valeur d'une propriété de cardinalité dans les données de transaction définit le nombre de fois qu'il convient de répéter un bloc ou une liste de propriétés. Dans ce cas, "LIST OF CLASS_REFERENCE_TYPE" ou "LIST OF CLASS_INSTANCE_TYPE" doit être utilisé pour les propriétés afin de définir les nombres minimal et maximal de blocs.

La Figure 27 présente un exemple de vue d'une présentation classique d'une LOP et de ses blocs imbriqués dans une application tableur. Dans cet exemple, la ligne 2 présente la LOP, les lignes 3 à 5, 17, 30 et 36 présentent les blocs, alors que les autres lignes présentent les propriétés. Les colonnes A à D présentent le niveau de chaque bloc. Par exemple, la LOP "Flow – gauge, meter, switch OLOP", décrite dans la ligne 2, est la racine de la structure, et elle est donc décrite dans la première colonne "A". Le bloc "Operating list of properties of flowmeter", décrit dans la ligne 3, est contenu dans la LOP. Par conséquent, le bloc est décrit dans la deuxième colonne "B". Dans cet exemple, la hiérarchie, qui est composée des LOP et de leurs blocs, s'affiche de manière hiérarchique.



IEC 2314/13

Légende

Anglais	Français
View of all properties of each class	Vue de toutes les propriétés de chaque classe
Property ID	Identificateur de propriété
Class ID	Identificateur de classe
Flow-gauge, meter, switch OLOP	Débitmètre, compteur, commutateur OLOP
Operating list of properties of flowmeter	(OLOP) Liste de propriétés de fonctionnement du débitmètre
Administrative information	Informations d'administration

Anglais	Français
Document information	Informations relatives au document
Document identifier	Identificateur du document
Document version	Version du document
Document revision	Révision du document
Document type	Type du document
Date of generation	Date de création
Time of generation	Heure de création
Author	Auteur
Document designation	Désignation du document
Document description	Description du document
Language	Langue
Remark	Remarque
Project information	Informations relatives au projet
Project number	Numéro du projet
Subproject number	Numéro de sous-projet
Project title	Titre de projet
Entreprise	Entreprise
Site	Site
Area	Zone
Process plant	Installation de transformation
Process cell	Cellule de processus
Unit	Unité
Equipment	Équipement
Equipment module	Module d'équipement
Device identification code	Code d'identification de dispositif
Unit Operations	Exploitations d'unité
Unit operations designation	Désignation d'exploitations d'unité
Related equipment identifier	Identificateur d'équipement connexe
Service description	Description de service
Process flow diagram/ reference document	Diagramme de flux de processus / document de référence
Piping & instrument diagram / reference document	Schéma de tuyauterie et d'instrumentation, P&ID
Base conditions	Conditions de base
Absolute base pressure	Pression absolue de base
Base temperature	Température de base

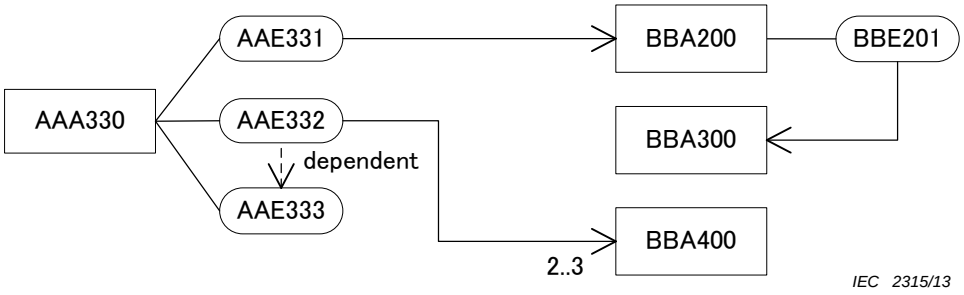
Figure 27 – Exemple de vue d'une LOP et de blocs imbriqués

Pour éviter les définitions redondantes d'éléments identiques, il convient de ne définir qu'une seule fois les propriétés qui peuvent être partagées par plusieurs LOP ou blocs et de les référencer à partir des autres endroits où elles sont utilisées. Si aucune branche générale de l'arbre de classification ne spécifie à quel endroit peuvent se trouver ces propriétés couramment utilisées, il convient de les importer à partir d'une LOP ou d'un bloc d'une autre branche à l'aide du mécanisme "case_of" présenté en 6.3.

La Figure 28 présente un exemple de cas d'utilisation classique d'un bloc, y compris le bloc multiple et le bloc imbriqué. L'exemple présente deux branches, l'une étant uniquement

composée de la classe AAA330 (à gauche de la Figure) et l'autre des trois classes de bloc BBA200, BBA300 et BBA400 (à droite de la Figure). La classe AAA330 comporte les trois propriétés AAE331 à AAE333. La propriété AAE331 est définie comme CLASS_REFERENCE_TYPE spécifiant la classe de bloc BBA200, alors que la propriété AAE332 est définie comme LIST(2,3) OF CLASS_REFERENCE_TYPE spécifiant la classe de bloc BBA400. La dernière propriété spécifie également la propriété AAE333 comme étant sa propriété de contrôle. La classe de bloc BBA200 contient la propriété BBE201, qui est définie comme CLASS_REFERENCE_TYPE spécifiant la classe de bloc BBA300 (c'est-à-dire le bloc imbriqué).

NOTE Les classes AAA330, BBA200, BBA300 et BBA400 font partie de la hiérarchie globale de classes. Pour de raisons de clarté, la superclasse est omise et n'est pas présentée dans la figure.

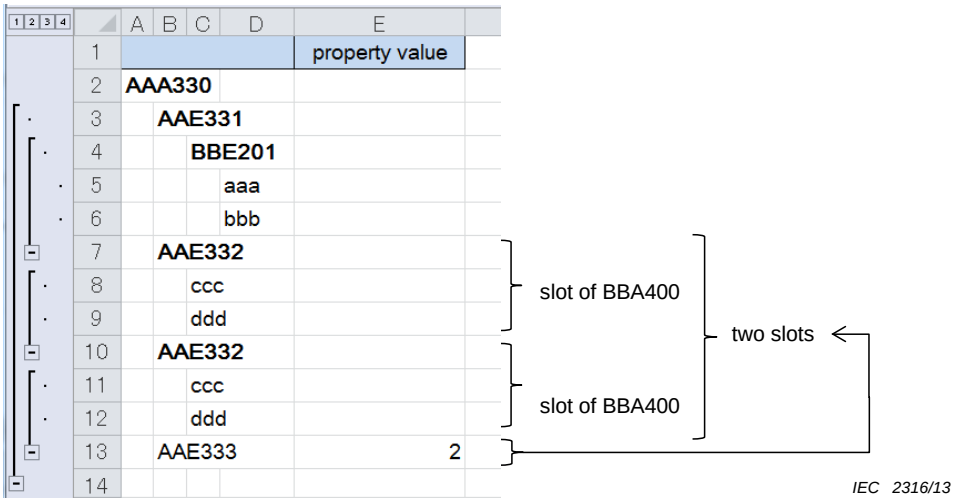


Légende

Anglais	Français
dependent	dépendant

Figure 28 – Exemple de cas d'utilisation de blocs

La Figure 29 présente un exemple de la vue de composition de AAA330 au niveau de la couche Bibliothèque de domaine représentant la structure illustrée à la Figure 28. Elle illustre la manière d'utiliser la propriété de contrôle AAE333 au niveau de la couche Bibliothèque de domaine. Chaque ligne décrite par des lettres en gras est un bloc, les autres lignes étant des propriétés normales. Dans cette figure, deux emplacements sont prévus pour la propriété AAE332 en fonction de la valeur de sa propriété de contrôle AAE333. Noter que les propriétés décrites en lettres minuscules (aaa à ddd) sont les propriétés des blocs, mais ces propriétés sont omises à la Figure 28 pour des raisons de simplicité.



Légende

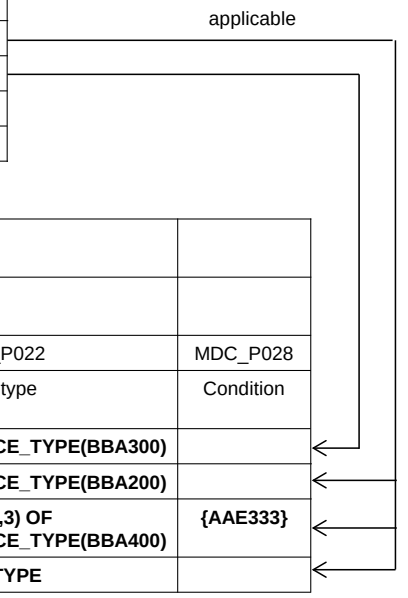
Anglais	Français
property value	valeur de propriété
slot of BBA400	emplacement de BBA400
two slots	deux emplacements

Figure 29 – Exemple d'une vue de composition d'une LOP

La Figure 30 présente un exemple de paquets conjonctifs permettant de décrire le dictionnaire de données présenté à la Figure 28. La classe et chaque classe de blocs sont décrites dans la feuille de paquet de classes en haut de la figure, chaque propriété étant décrite dans la feuille de paquet de propriétés en bas de la figure.

#CLASS_ID:= **MDC_C002**			
#CLASS_NAME.en:= **Class meta-class**			
#PROPERTY_ID	MDC_P001_5	MDC_P011	MDC_P014
#PROPERTY_ NAME.en	Code	Class type	Applicable properties
	AAA330	ITEM_CLASS	**{AAE331,AAE332,AAE333}**
	BBA200	ITEM_CLASS	**{BBE201}**
	BBA300	ITEM_CLASS	
	BBA400	ITEM_CLASS	

#CLASS_ID:= **MDC_C003**				
#CLASS_NAME.en:= **Property meta-class**				
#PROPERTY_ID	MDC_P001_6	MDC_P020	MDC_P022	MDC_P028
#PROPERTY_ NAME.en	Code	Property data element type	Data type	Condition
	BBE201	**NON_DEPENDENT_P_DET**	**CLASS_REFERENCE_TYPE(BBA300)**	
	AAE331	**NON_DEPENDENT_P_DET**	**CLASS_REFERENCE_TYPE(BBA200)**	
	AAE332	**DEPENDENT_P_DET**	**LIST(2,3) OF CLASS_REFERENCE_TYPE(BBA400)**	**{AAE333}**
	AAE333	**CONDITION_DET**	**INT_TYPE**	



IEC 2317/13

Légende

Anglais	Français
#CLASS_NAME.en:=Class meta-class	#CLASS_NAME.en:= méta-classe de classes
Code	Code
Class type	Type de classe
Applicable properties	Propriétés applicables
applicable	applicable
#CLASS_NAME.en:=Property meta-class	#CLASS_NAME.en:= méta-classe de propriétés
Code	Code
Property data element type	Type d'élément de données d'une propriété
Data type	Type de données
Condition	Condition

Figure 30 – Mise en œuvre de paquet pour les blocs

8.4 Mise en œuvre de polymorphisme

Le polymorphisme (voir CEI 61987-10) est un mécanisme permettant de choisir parmi les blocs au niveau de la couche Bibliothèque de domaines. Dans la CEI 62656-1, le polymorphisme peut être simplement exprimé à l'aide de ENUM_REFERENCE_TYPE ou de ENUM_INSTANCE_TYPE. Ce type de données spécifie une énumération qui représente les identificateurs des classes à sélectionner comme choix polymorphiques. Une liste de ces identificateurs est spécifiée comme étant une valeur de l'attribut MDC_P025_1 (Symbole littéral préférentiel en texte) du paquet de termes.

Pour spécifier explicitement un choix polymorphique au niveau de la Bibliothèque de domaines, une propriété de condition peut faire office de propriété de contrôle (voir 8.1). Cette propriété de condition est définie en tant que ENUM_STRING_TYPE, qui spécifie la même énumération de la propriété polymorphique.

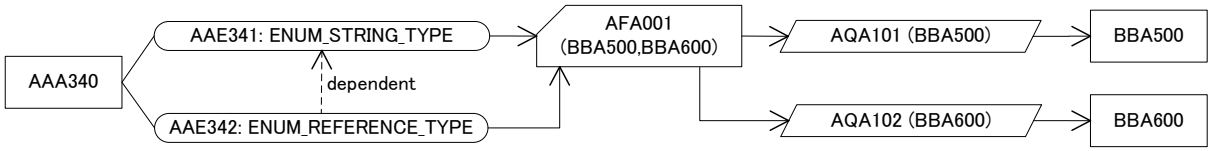
NOTE Traditionnellement, dans le CEI CDD, les propriétés sont directement assignées à une classe, chacune étant de CLASS_REFERENCE_TYPE ou CLASS_INSTANCE_TYPE et pointant vers une classe à sélectionner comme choix polymorphique. Dans ce cas, pour grouper ces propriétés et donner une fonction permettant d sélectionner l'un des choix au niveau de la couche Bibliothèque de domaine, une propriété de type énumération (type non quantitatif) comme condition pour contrôler la sélection est également assignée dans la même classe et ces propriétés dépendent de cette propriété de contrôle.

Chaque propriété polymorphique peut permettre plusieurs utilisations du bloc associé, et ce, conformément au mécanisme de cardinalité expliqué en 8.2. Dans ce cas, il convient de définir ce type de propriété en tant que LIST OF ENUM_REFERENCE_TYPE ou LIST OF ENUM_INSTANCE_TYPE. Si une propriété de contrôle est utilisée pour énumérer des choix polymorphiques possibles, il convient de définir ce type de propriété en tant que LIST OF ENUM_STRING_TYPE.

La Figure 31 est un exemple de cas d'utilisation de polymorphisme. La propriété AAE342 agit comme un pointeur vers les choix polymorphiques disponibles. Les choix polymorphiques pour la propriété AAE342 sont proposés par l'énumération AFA001 spécifiant les deux termes AQA101 et AQA102, qui à leur tour se réfèrent aux classes de blocs BBA500 et BBA600 comme étant les choix polymorphiques. Par ces relations, la propriété AAE342 offre les choix BBA500 et BBA600 comme ses valeurs. Dans cet exemple, la propriété AAE341 est fournie en plus pour permettre de spécifier un choix de manière explicite au niveau de la couche Bibliothèque de domaine. Par conséquent, la propriété AAE341 est une propriété de condition pour la propriété AAE342 et elle est définie en tant que ENUM_STRING_TYPE utilisant la même énumération que celle utilisée par la propriété AAE342.

NOTE 1 Les classes AAA340, BBA500 et BBA600 font partie de la hiérarchie globale de classes. Pour de raisons de clarté, la superclasse est omise et n'est pas présentée dans la figure.

NOTE 2 Traditionnellement, dans l'exemple ci-dessus, deux propriétés sont assignées à AAA340 et elles sont de CLASS_REFERENCE_TYPE et pointent respectivement vers BBA500 et BBA600 comme choix polymorphiques. Dans ce cas, ces propriétés dépendent de AAE341 en tant que leur propriété de contrôle.



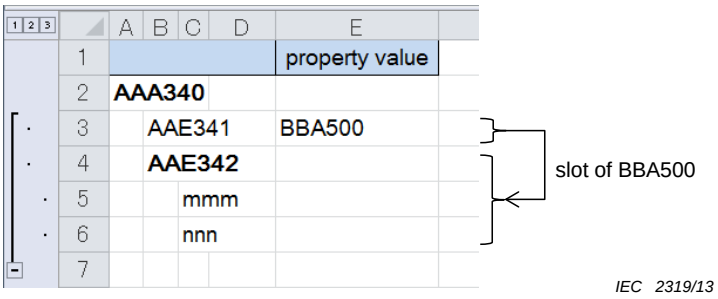
IEC 2318/13

Légende

Anglais	Français
dependent	dépendant

Figure 31 – Exemple de cas d'utilisation de polymorphisme

La Figure 32 présente un exemple de la vue de composition de AAA340 au niveau de la couche Bibliothèque de domaine représentant la structure illustrée à la Figure 31. Chaque ligne utilisant des lettres en gras représente un bloc, alors que chacune des autres lignes représente une propriété. Dans cette figure, la propriété AAE341 spécifie "BBA500", le bloc BBA500 est donc sélectionné comme étant le choix réel de la propriété AAE342. Noter que les propriétés décrites en lettres minuscules (mmm à nnn) sont des propriétés pour le bloc BBA500. Les propriétés mmm et nnn sont omises à la Figure 31, pour des raisons de simplicité.

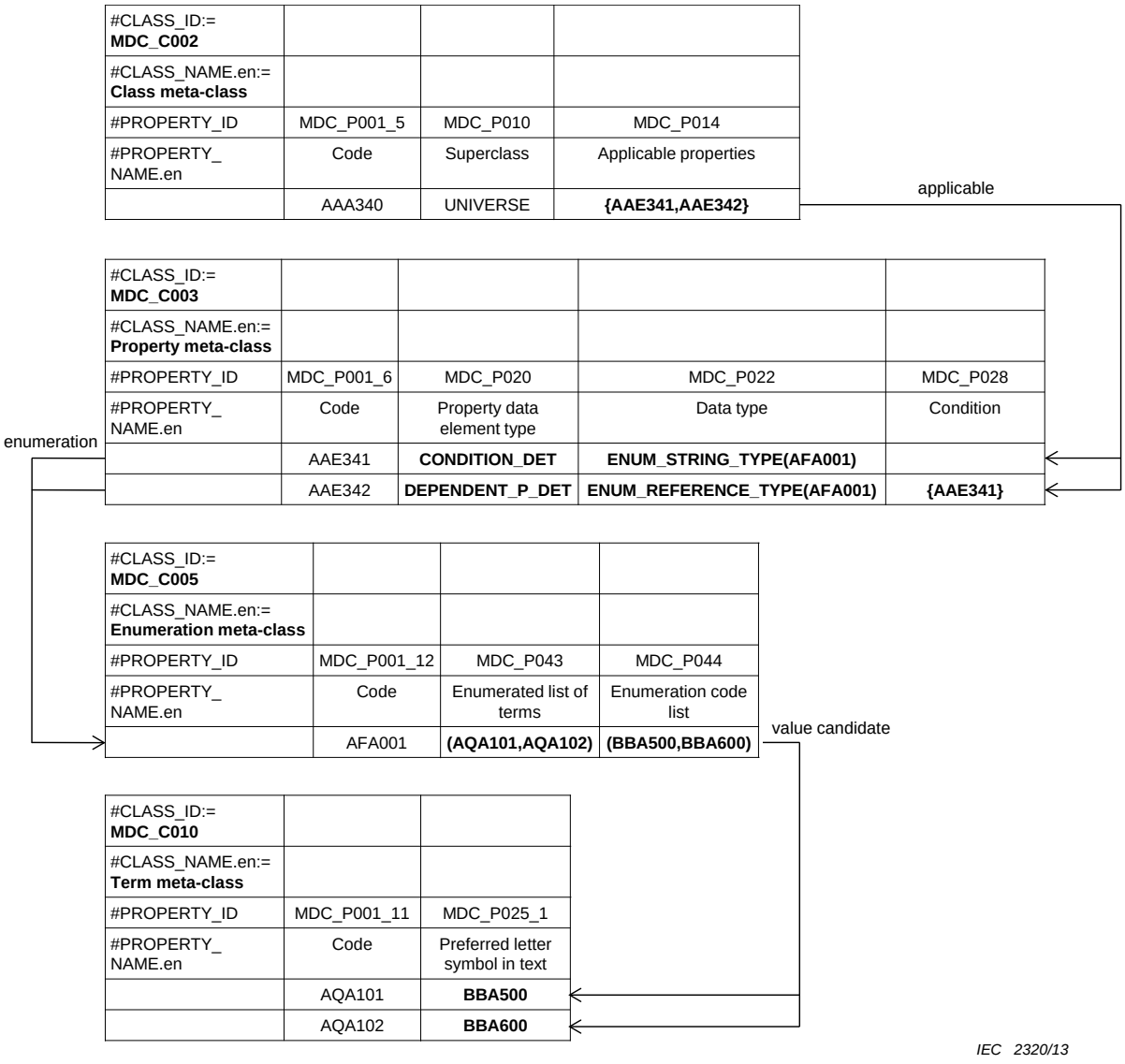


Légende

Anglais	Français
property value	valeur de propriété
slot of BBA500	emplacement de BBA500

Figure 32 – Exemple de vue de composition pour le polymorphisme

La Figure 33 présente des feuilles de paquets conjonctifs pour le dictionnaire de données présenté à la Figure 31. Les propriétés AAE341 et AAE342 sont respectivement définies en tant que ENUM_STRING_TYPE et ENUM_REFERENCE_TYPE, chacune d'elles spécifiant l'identificateur de l'énumération AFA001 entre ces parenthèses. Les blocs BBA500 et BBA600, qui sont les choix polymorphiques pour ces propriétés, sont spécifiés dans les attributs MDC_P025_1 (Symbole littéral préférentiel en texte) des termes AQA101 et AQA102, respectivement.



Légende

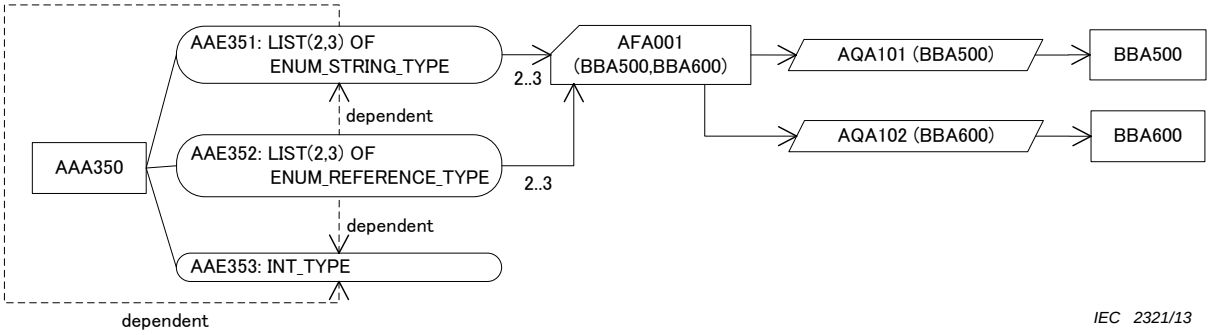
Anglais	Français
#CLASS_NAME.en:=Class meta-class	#CLASS_NAME.en:= méta-classe de classes
Code	Code
Superclass	Superclasse
Applicable properties	Propriétés applicables
applicable	applicable
#CLASS_NAME.en:=Property meta-class	#CLASS_NAME.en:= méta-classe de propriétés
Property data element type	Type d'élément de données d'une propriété
Data type	Type de données
Condition	Condition
enumeration	énumération
#CLASS_NAME.en:=Enumeration meta-class	#CLASS_NAME.en:= méta-classe d'énumérations
Enumerated list of terms	Liste énumérée de termes
Enumeration code list	Liste des codes d'énumération
value candidates	valeurs candidates
#CLASS_NAME.en:=Term meta-class	#CLASS_NAME.en:= méta-classe de termes
Preferred letter symbol in text	Symbole littéral préférentiel en texte

Figure 33 – Mise en œuvre de paquet pour le polymorphisme

La Figure 34 est une extension de l'exemple présenté à la Figure 31 permettant plusieurs choix de polymorphisme. Dans cet exemple, les propriétés AAE351 et AAE352 sont définies en tant que type LIST, ce qui permet d'identifier les choix multiples qu'elles proposent. La

propriété AAE353 est également ajoutée en tant que propriété de contrôle pour les propriétés AAE351 et AAE352, indiquant la répétition de ces propriétés.

NOTE Les classes AAA350, BBA500 et BBA600 font partie de la hiérarchie globale de classes. Pour de raisons de clarté, la superclasse est omise et n'est pas présentée dans la figure.

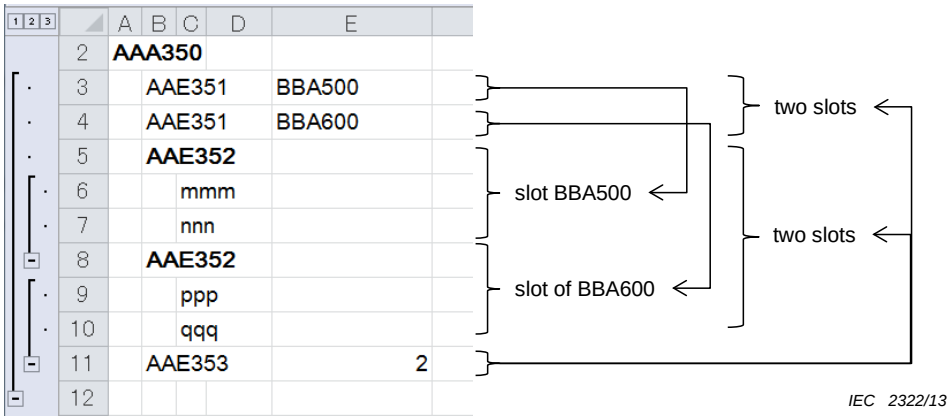


Légende

Anglais	Français
dependent	dépendant

Figure 34 – Exemple de cas d'utilisation de polymorphisme à choix multiples

La Figure 35 présente un exemple de la vue de composition de AAA350 au niveau de la couche Bibliothèque de domaine afin de représenter la structure illustrée à la Figure 34. Elle illustre également la manière d'utiliser la propriété de contrôle AAE353 au niveau de la couche Bibliothèque de domaine. Chaque ligne décrite par des lettres en gras est un bloc, les autres lignes étant des propriétés normales. Dans cette figure, deux emplacements sont prévus pour la propriété AAE351 et la propriété polymorphique AAE352 en fonction de la valeur de sa propriété de contrôle AAE353. La première ligne de la propriété AAE351 spécifie "BBA500", le bloc BBA500 est donc sélectionné comme étant le premier emplacement de la propriété AAE352. La deuxième ligne de la propriété AAE351 spécifie "BBA600", le bloc BBA600 est donc sélectionné comme étant le deuxième emplacement de la propriété AAE352. Noter que les propriétés décrites en lettres minuscules (mmm à qqq) sont des propriétés pour les blocs, mais elles ne sont pas présentées à la Figure 34 pour des raisons de simplicité.

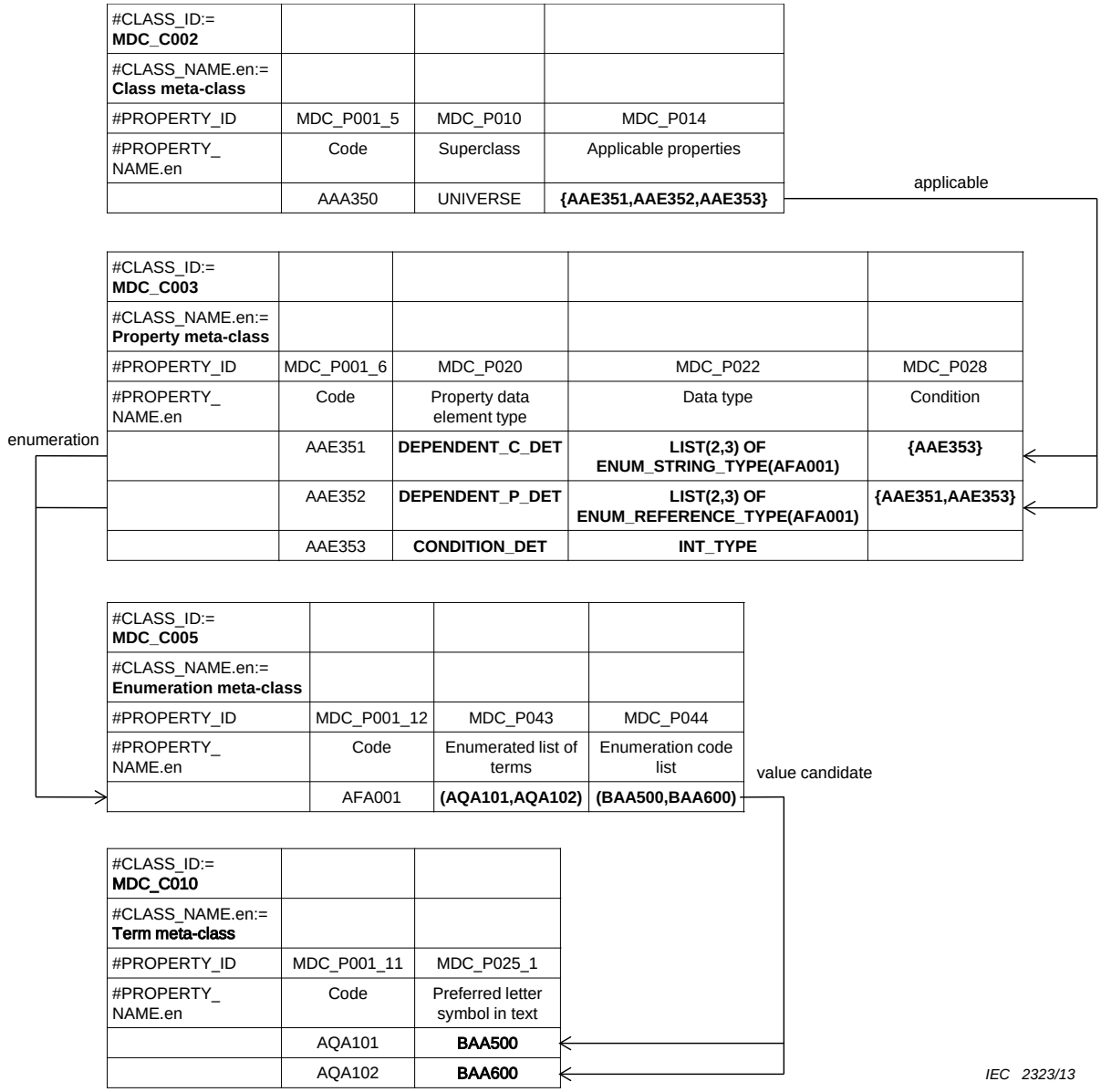


Légende

Anglais	Français
slot BBA500	emplacement BBA500
slot of BBA600	emplacement de BBA600
two slots	deux emplacements

Figure 35 – Exemple de vue de composition pour le polymorphisme à choix multiples

La Figure 36 présente des feuilles de paquets conjonctifs pour le dictionnaire de données présenté à la Figure 34. Pour les choix multiples, les propriétés AAE351 et AAE352 sont respectivement définies en tant que LIST(2,3) OF ENUM_STRING_TYPE et LIST(2,3) OF ENUM_REFERENCE_TYPE, chacune d'elles spécifiant l'identificateur de l'énumération AFA001 entre ces parenthèses. La propriété AAE353 est définie en tant que INT_TYPE et elle est spécifiée en tant que propriété de condition utilisant les deux propriétés AAE351 et AAE352. La propriété AAE351 est la propriété de condition de la propriété AAE352 et dépend de la propriété AAE353. Par conséquent, la propriété AAE351 est définie en tant que DEPENDENT_C_DET de la manière expliquée en 8.1.



Légende

Anglais	Français
#CLASS_NAME.en:=Classes meta-class	#CLASS_NAME.en:= méta-classe de classes
Code	Code
Superclass	Superclasse
Applicable properties	Propriétés applicables
applicable	applicable
#CLASS_NAME.en:=Property meta-class	#CLASS_NAME.en:= méta-classe de propriétés

Anglais	Français
Property data element type	Type d'élément de données d'une propriété
Data type	Type de données
Condition	Condition
enumeration	énumération
#CLASS_NAME.en:=Enumeration meta-class	#CLASS_NAME.en:= méta-classe d'énumérations
Enumerated list of terms	Liste énumérée de termes
Enumeration code list	Liste des codes d'énumération
value candidates	valeurs candidates
#CLASS_NAME.en:=Term meta-class	#CLASS_NAME.en:= méta-classe de termes
Preferred letter symbol in text	Symbole littéral préférentiel en texte

Figure 36 – Mise en œuvre de paquet pour le polymorphisme à choix multiples

8.5 Identificateurs alternatifs

Une classe et une propriété peuvent comporter un identificateur alternatif. Le cas d'utilisation classique d'un identificateur alternatif consiste à mapper deux éléments ontologiques analogues ou identiques à partir de dictionnaires de données différents, modélisés en classes différentes pour des raisons historiques ou organisationnelles.

Pour une propriété, si des biens doivent être spécifiés comme propriétés alternatives, les identifiants des propriétés alternatives sont spécifiées sous la forme d'une valeur de l'attribut MDC_P101 (Identificateur alternatif). Pour une classe, si une classe doit être précisée comme classe alternative à l'ancienne classe, l'identificateur de la classe alternative est spécifié comme valeur d'un attribut MDC_P102 (Identificateur alternatif).

La Figure 37 présente un exemple de feuille de paquet de propriétés dans laquelle trois propriétés et leurs identificateurs alternatifs sont identifiés. La propriété AAE361 spécifie la propriété BBE001 comme étant son identificateur alternatif, qui est défini dans un autre dictionnaire de données. La propriété AAE362 spécifie également un ensemble des identificateurs de propriété BBE002 et CCE001, qui sont définis dans d'autres dictionnaires de données. Enfin, aucune propriété particulière ne fait office d'identificateur alternatif pour la propriété AAE363. Par conséquent, le champ de l'attribut MDC_P101 reste vide.

#CLASS_ID:=**MDC_C003**		
#CLASS_NAME.en:=**Property meta-class**		
#PROPERTY_ID	MDC_P001_6	MDC_P101_6
#PROPERTY_NAME.en	Code	Alternate ID
	AAE361	**BBE001**
	AAE362	**{BBE002,CCE001}**
	AAE363	

IEC 2324/13

Légende

Anglais	Français
#CLASS_NAME.en:=Property meta-class	#CLASS_NAME.en:= méta-classe de propriétés
Code	Code
Alternate ID	Identificateur alternatif

Figure 37 – Mise en œuvre de paquet pour l'identificateur alternatif

9 Représentation de fichier de données pour le stockage et l'échange

9.1 Format CSV pour la représentation des paquets de données

La CEI 62656-1 ne spécifie aucun format de fichier pour le stockage et l'échange de données, mais le format CSV (Comma Separated Values, c'est-à-dire valeurs séparées par une virgule, voir [10]) est recommandé, car ce format CSV s'apparente à la structure d'une feuille de paquet et est acceptable pour différentes applications sans aucune ou très peu de modification. Les paragraphes 9.2 à 9.5 donnent des informations pertinentes relatives à la lecture et l'écriture correctes de données dans un fichier CSV.

9.2 Délimiteur de cellules

Comme l'indique le nom CSV, une virgule est généralement utilisée en tant que délimiteur de cellule. Toutefois, plusieurs applications utilisent un autre caractère si la virgule est déjà utilisée à d'autres fins (un point décimal dans certaines langues européennes, par exemple). Ainsi, il convient que l'outil de paquetage reconnaisse le caractère faisant office de délimiteur de cellule afin de déterminer correctement la valeur de chaque cellule.

Si une valeur de cellule contient un délimiteur de cellule, cette valeur doit être placée entre des qualificatifs de texte.

EXEMPLE Si le caractère point-virgule est un délimiteur de cellules et la valeur de cellule est “yes;no;unknown”, cette valeur est supposée être placée entre des qualificatifs de texte. Dans ce cas, la virgule n'est pas un délimiteur de cellule et, donc, les valeurs contenant "," (par exemple: “yes, no or unknown”) peuvent ne pas être placées entre des qualificatifs de texte.

9.3 Caractère de saut de ligne

Tous les outils commerciaux ou non commerciaux ne permettent pas d'utiliser le caractère de saut de ligne dans une valeur de cellule. Par conséquent, le caractère de saut de ligne ne doit pas être utilisé dans une valeur de cellule, sauf si les applications concernées par l'échange de données sont en mesure de traiter correctement les données le contenant. S'il est admis d'utiliser le caractère de saut de ligne, toutes les valeurs de cellule le contenant doivent être placées entre des qualificatifs de texte.

NOTE Le nombre de qualificatifs de texte d'une ligne d'un fichier CSV est toujours pair. Par conséquent, s'il est impair, la ligne est concaténée avec le caractère de saut de ligne et la ligne suivante, dans l'ordre.

La Figure 38 présente un exemple de données CSV contenant le caractère de saut de ligne dans une valeur de cellule.

La première ligne contient des données CSV valides, représentées par une ligne et deux colonnes dans les applications de tableur. Le premier élément contient le caractère de saut de ligne dans sa valeur. La valeur est donc placée entre des qualificatifs de texte.

La deuxième ligne contient des données CSV non valides, car la valeur du premier élément n'est pas placée entre des qualificatifs de texte. Il s'ensuit que les données ne s'affichent pas correctement sur deux lignes et deux colonnes dans les applications tableur.

La troisième ligne contient également des données CSV non valides, car la valeur du premier élément commence mais ne se termine pas par un qualificatif de texte. Il s'ensuit que les données ne s'affichent pas correctement sur une ligne et une colonne dans un tableur.

CSV data	View on a text editor	View on spreadsheet	Correctness
“sentence 1.<LF>sentence 2.”,sentence 3.	“sentence 1.<LF> sentence 2. “,sentence 3.	sentence 1. sentence 2. | sentence 3.	valid
sentence 1.<LF>sentence 2.,sentence 3.	sentence 1.<LF> sentence 2.sentence 3.	sentence 1. | sentence 2. | sentence 3.	invalid
“sentence 1.<LF>sentence 2.,sentence 3.	“sentence 1.<LF> sentence 2.sentence 3.	sentence 1. sentence 2., sentence 3.	invalid

IEC 2325/13

Légende

Anglais	Français
CSV data	Données CSV
View on a text editor	Vue dans un éditeur de texte
View on spreadsheet	Vue dans le tableur
Correctness	Exactitude
valid	valide
invalid	non valide
sentence 1	phrase 1
sentence 2	phrase 2
sentence 3	phrase 3

Figure 38 – Exemple d'échappement des caractères de saut de ligne

9.4 Caractère d'espace

Un caractère d'espace inutile inséré avant ou après une valeur afin d'en faciliter la lecture ou à cause d'une erreur typographique doit être ignoré. S'il est prévu de conserver ce type de caractère d'espace, la valeur le/les contenant doit être placée entre des qualificatifs de texte. Il convient d'appliquer également cette règle à chaque valeur membre d'une agrégation.

EXEMPLE 1 Si les propriétés applicables sont décrites sous la forme "{P001, P002, P003}", les caractères d'espace qui précèdent P002 et P003 sont ignorés.

EXEMPLE 2 Si une propriété "SET OF STRING_TYPE" contient une valeur "{abc, efg,"" xyz ""}", le caractère d'espace qui précède "efg" est ignoré, mais les caractères espace qui précèdent et suivent "xyz" sont conservés.

9.5 Codage de caractères

Un paquet de données peut contenir un ou plusieurs caractères à plusieurs octets. Dans ce cas, le codage de caractère d'un fichier CSV doit prendre en charge ces caractères afin d'éviter les problèmes de stockage et d'échange de données. Par conséquent, il est recommandé d'utiliser UTF-8 ou UTF-16.

10 Conformité à la mise en œuvre du CEI CDD

La CEI 62656-1 définit les classes de conformité POM (Parcellized ontology model – "Modèle d'ontologie paquetée"). Les classes de conformité sont classées selon le numéro de niveau de la couche supérieure des paquets utilisés dans le cadre d'un échange, conformément au schéma de couche MoF (Meta-object facility [11] ("Fonction méta-objets")), puis sous-classées en fonction des couches incluses dans l'échange. La classification peut également spécifier la présence (ou l'absence) d'extension(s) dans les capacités de modélisation n'entrant pas dans le cadre de la CEI 61360. De plus, la CEI 62656-1 permet de spécifier un modèle d'ontologie par le nom, qui doit être utilisé pour traiter les paquets de données. Le

nom doit être spécifié dans les parenthèses ajoutées après le CCID (Conformance class ID, c'est-à-dire Identificateur de classe de conformité).

La présente partie de la CEI 62656 définit deux classes de conformité supplémentaires, 2(CDD) et 2X(CDD) comme étant les spécialisations des classes de conformité 2 et 2X définies dans la CEI 62656-1, pour le téléversement et le téléchargement des ontologies de domaine ou des dictionnaires de données vers et depuis le CEI CDD. Noter que si tous les attributs du CEI CDD sont fournis dans d'autres classes de conformité, qui incluent les couches M3-M2 et M4-M3 ainsi que la couche M2-M1, par exemple, dans les classes de conformité 3B, 3BX, 4C et 4CX, ces classes de conformité satisfont également aux exigences de conformité du CEI CDD.

Par conséquent, le Tableau 2 énumère toutes les classes de conformité définies dans la CEI 62656-1 et les classes de conformité supplémentaires 2(CDD) et 2X(CDD).

Tableau 2 – Classes de conformité POM

CCID	Définition	Couches MoF
1	Paquet de données spécialement pour DL (Domain Library ("Bibliothèque de domaine"))	M1-M0
1X	Paquet de données seulement pour DL avec une extension locale des propriétés et probablement de leurs valeurs d'instances	M1-M0
2	Paquet de données spécialement pour DO (Domain Ontology ("Ontologie de domaine"))	M2-M1
2X	Paquet de données spécialement pour DO avec une extension locale des propriétés et probablement de leurs valeurs d'instances	M2-M1
2(CDD)	**Paquet de données pour DO à enregistrer dans le CEI CDD**	**M2-M1**
2X(CDD)	**Paquet de données pour DO à enregistrer dans le CEI CDD avec une extension locale des propriétés**	**M2-M1**
2A	Paquets de données pour toutes les couches ci-dessous comprenant DO et DL	M2-M1, M1-M0
2AX	Paquets de données pour toutes les couches ci-dessous comprenant DO et DL avec une extension locale des propriétés et probablement de leurs valeurs d'instances	M2-M1, M1-M0
3	Paquet de données spécialement pour MO (Méta-ontologie)	M3-M2
3X	Paquet de données seulement pour MO avec une extension locale des propriétés et probablement de leurs valeurs d'instances	M3-M2
3A	Paquets de données avec toutes les couches ci-dessous, comprenant MO, DO et DL	M3-M2, M2-M1, M1-M0
3AX	Paquets de données avec toutes les couches ci-dessous, comprenant MO, DO et DL avec une extension locale des propriétés et probablement de leurs valeurs d'instances	M3-M2, M2-M1, M1-M0
3B	Paquets de données avec la couche ci-dessous comprenant MO et DO	M3-M2, M2-M1
3BX	Paquets de données avec la couche ci-dessous comprenant MO et DO avec une extension locale des propriétés et probablement de leurs valeurs d'instances	M3-M2, M2-M1
4	Paquet de données spécialement pour AO (Axiomatic ontology (Ontologie axiomatique"))	M4-M3
4X	Paquet de données seulement pour AO avec une extension locale des propriétés et probablement de leurs valeurs d'instances	M4-M3
4A	Paquets de données avec toutes les couches ci-dessous, comprenant AO, MO, DO et DL	M4-M3, M3-M2, M2-M1, M1-M0
4AX	Paquets de données avec toutes les couches ci-dessous, comprenant AO, MO, DO et DL avec une extension locale des propriétés et probablement de leurs valeurs d'instances	M4-M3, M3-M2, M2-M1, M1-M0
4B	Paquets de données avec la couche ci-dessous comprenant AO et MO	M4-M3, M3-M2
4BX	Paquets de données avec la couche ci-dessous comprenant AO et MO avec une extension locale des propriétés et probablement de leurs valeurs d'instances	M4-M3, M3-M2
4C	Paquets de données avec toutes les couches excepté DL comprenant AO, MO et DO.	M4-M3, M3-M2, M2-M1
4CX	Paquets de données avec toutes les couches excepté DL comprenant AO, MO et DO avec une extension locale des propriétés et probablement de leurs valeurs d'instances	M4-M3, M3-M2, M2-M1

Annexe A
(normative)

Enregistrement d'objet d'informations – Identification de document

Afin de fournir une identification sans ambiguïté d'un objet d'informations dans un système ouvert, l'identificateur d'objet

{iec standard 62656 part (2) version (1)}

est assigné à la présente partie de la CEI 62656. La signification de cette valeur est définie dans l'ISO/CEI 8824-1 [12].

Annexe B
(informative)

Exemples de contraintes de modèle pour les attributs

Le Tableau B.1 donne des exemples de contraintes de modèle classiques qui peuvent être appliquées à certains attributs définis dans la CEI 62656-1. Chaque contrainte de modèle peut être spécifiée dans la ligne de l'instruction préservée "#PATTERN" d'une feuille de paquet, pour limiter d'un point de vue syntaxique ou vérifier les valeurs de la section de données de la feuille. Les attributs qui n'apparaissent pas dans le tableau suivant signifient qu'il n'existe aucune règle particulière à leur appliquer.

Chaque chaîne de caractères en italique dans la troisième colonne est un modèle qui peut être communément utilisé par plusieurs attributs. Son modèle réel est décrit dans la note de bas du tableau.

Tableau B.1 – Exemples de contraintes de modèle pour les attributs *(1 de 3)*

Code	Nom préférentiel	Contrainte de modèle
MDC_P001_X	Code	id_pattern [a]
MDC_P002_1	Numéro de version	[0-9]{1,10}
MDC_P002_2	Numéro de révision	[0-9]{1,3}
MDC_P002_3	Révision du contenu	[0-9]{1,3}
MDC_P003_1	Date de la définition originale	date_pattern [b]
MDC_P003_2	Date de la version actuelle	date_pattern
MDC_P003_3	Date de la révision actuelle	date_pattern
MDC_P004_2	Nom synonyme	\{\(element_pattern,lang_pattern\)(,\(element_pattern,lang_pattern\))*\} [c d]
MDC_P004_4	Icône de nom	id_pattern
MDC_P008_1	Dessin simplifié	id_pattern
MDC_P008_2	Graphisme	id_pattern
MDC_P010	Superclasse	id_pattern
MDC_P011	Type de classe	ITEM_CLASS|COMPONENT_CLASS|MATERIAL_CLASS| FEATURE_CLASS|ITEM_CLASS_CASE_OF| COMPONENT_CLASS_CASE_OF|MATERIAL_CLASS_CASE_OF| FEATURE_CLASS_CASE_OF
MDC_P012	Fournisseur	[a-zA-Z0-9_/]+
MDC_P013	Is case of	id_set_pattern [e]
MDC_P014	Propriétés applicables	id_set_pattern
MDC_P015	Types applicables	id_set_pattern
MDC_P016	propriétés de sélection de sous-classes	id_set_pattern
MDC_P017	Affectation de valeurs de classe	\{\(id_pattern,element_pattern\)(,\(id_pattern,element_pattern\))*\}

Tableau B.1 *(2 de 3)*

Code	Nom préférentiel	Contrainte de modèle
MDC_P020	Type d'élément de données d'une propriété	NON_DEPENDENT_P_DET|DEPENDENT_P_DET|CONDITION_DET|DEPENDENT_C_DET
MDC_P021	Classe de définition	id_pattern
MDC_P028	Condition	id_set_pattern
MDC_P040	Classification DET	[A-Z][0-9]{2}
MDC_P041	Code pour l'unité	id_pattern
MDC_P042	Codes pour les unités alternatives	id_set_pattern
MDC_P043	Liste énumérée de termes	id_list_pattern [f]
MDC_P044	Liste des codes d'énumération	\(element_pattern(,element_pattern)*\)
MDC_P051_11	Adresse courriel	([a-zA-Z0-9][a-zA-Z0-9_\.+\\-]*)@(([a-zA-Z0-9][a-zA-Z0-9_\\-]*\.)+[a-zA-Z]{2,6})
MDC_P062	Emplacement distant	url_pattern [g]
MDC_P065_3	Codage du contenu principal	7bit|8bit|binary|quoted-printable|base64
MDC_P065_9	Accès distant du contenu principal	url_pattern
MDC_P068	Contrainte de propriété	\{property_constraint_pattern(,property_constraint_pattern)*\} [h]
MDC_P072	État LIIM	WD|CD|DIS|FDIS|IS|TS|PAS|ITA
MDC_P074	Date LIIM	[0-9]{1,4}
MDC_P075	Application LIIM	[1-6]E?
MDC_P080	Langue globale	lang_pattern
MDC_P081	Langue source	lang_pattern
MDC_P090	Propriétés importées	id_set_pattern
MDC_P091	Types importés	id_set_pattern
MDC_P093	Documents importés	id_set_pattern
MDC_P094	Documents applicables	id_set_pattern

Tableau B.1 *(3 de 3)*

Code	Nom préférentiel	Contrainte de modèle
MDC_P096	Classification de la propriété	\{\(id_pattern,[1-9]*[0-9],element_pattern\)(,\(id_pattern,[1-9]*[0-9],element_pattern\))*\}
MDC_P097	Exigence des propriétés	\{\(id_pattern,requisity_pattern\)(,\(id_pattern,requisity_pattern\))*\} [i]
MDC_P110	Super propriété	id_pattern
MDC_P200	Type de relation	FUNCTION|FUNC|PREDICATION|PRED
MDC_P201	Domaine de la relation	id_list_pattern
MDC_P202	Domaine de la fonction	id_list_pattern
MDC_P203	Co-domaine de la fonction	id_pattern
MDC_P205	Langage pour l'interprétation de la formule	lang_pattern
MDC_P207	Déclencheur d'événement	\(id_pattern(,id_pattern)+\)

[a] id_pattern:= ([a-zA-Z0-9_/]+#)?[a-zA-Z0-9]+(##[0-9]{1,10})?(###.*)?

[b] date_pattern:= ([1-9]{0,3}[0-9]\-((0[1-9])|(1[012]))\-((0[1-9])|([12][0-9])|(3[01])))

[c] element_pattern:= ([^\(\)\{\},"]+|"(([^"])|(""))*")

[d] lang_pattern:= [a-z]{2,3}

[e] id_set_pattern:= \{id_pattern(,id_pattern)*\}

[f] id_list_pattern:= \(id_pattern(,id_pattern)*\)

[g] url_pattern:= ((https?|ftp)(:\/\/[-_.!~*\'()a-zA-Z0-9;\/?:\@&=+\$,%#]+))

[h] property_constraint_pattern:= ((SUBCLASS_CONSTRAINT\(id_set_pattern\))
|(ENTITY_SUBTYPE_CONSTRAINT\(element_pattern(,element_pattern)*\))
|(ENUMERATION_CONSTRAINT\(element_pattern(,element_pattern)*\))
|(RANGE_CONSTRAINT\(real_pattern,(real_pattern|\?)(,(true)|(false)){2}\))
|(STRING_SIZE_CONSTRAINT\((0|([1-9][0-9]*)),(0|\?|([1-9][0-9]*))\))
|(STRING_PATTERN_CONSTRAINT\(element_pattern\))
|(CARDINALITY_CONSTRAINT\((0|[1-9][0-9]*),(0|\?|[1-9][0-9]*)\))) [j]

[i] requisity_pattern:= (KEY|MANDATORY|MAND|NOT NULL|OPTIONAL|OPT|DER)

[j] real_pattern:= (-?[1-9][0-9]*(\.[0-9]+)?|-0\.[0-9]*[1-9]|0\.[0-9]+|0)

Annexe C
(informative)

Exemples de valeurs d'attribut

Le Tableau C.1 donne un ensemble d'exemples de valeurs valides et non valides pour certains attributs définis dans la CEI 62656-1.

Tableau C.1 – Exemples de valeurs d'attribut *(1 de 3)*

Valeur prévue	Attribut	Exemple de valeur valide	Cas de valeurs non valides
ICID	MDC_P001_X (Code) MDC_P004_4 (Icône de nom) MDC_P008_1 (Dessin simplifié) MDC_P008_2 (Graphisme) MDC_P010 (Superclasse) MDC_P021 (Classe de définition) MDC_P041 (Code pour l'unité) MDC_P110 (Super propriété) MDC_P203 (Co-domaine de la fonction)	– 0112/2///62656_2#BBB001##1 – 0112/2///62656_2#BBB001##1###comments	a) violation de format d'ID – 0112/2///62656_2#BBB001#1 b) RAI ou VI manquant – BBB001##1 – 0112/2///62656_2#BBB001 – BBB001
ensemble d'ICID	MDC_P013 (Is case of) MDC_P014 (Propriétés applicables) MDC_P015 (Types applicables) MDC_P016 (Propriétés de sélection de sous-classes) MDC_P028 (Condition) MDC_P042 (Codes pour les unités alternatives) MDC_P090 (Propriétés importées) MDC_P091 (Types importés) MDC_P093 (Documents importés) MDC_P094 (Documents applicables) MDC_P207 (Événement déclencheur)	– {0112/2///62656_2#BBB001##1, 0112/2///62656_2#BBB002##1} – {0112/2///62656_2#BBB001##1###comment, 0112/2///62656_2#BBB002##1###comment}	a) violation de format d'ID b) les identificateurs ne sont pas placés entre accolades, mais entre parenthèses c) RAI ou VI est manquant au niveau d'un élément d'un ensemble d'identificateurs – {BBB001_X##1, BBB002##1} – {0112/2///62656_2#BBB001_X, 0112/2///62656_2#BBB002} – {BBB001_X, BBB002}

Tableau C.1 *(2 de 3)*

Valeur prévue	Attribut	Exemple de valeur valide	Cas de valeurs non valides
liste d'ICID	MDC_P043 (Liste énumérée de termes) MDC_P201 (Domaine de la relation) MDC_P202 (Domaine de la fonction)	– (0112/2///62656_2#BBB001##1, 0112/2///62656_2#BBB002##1) – (0112/2///62656_2#BBB001##1###comment, 0112/2///62656_2#BBB002##1###comment)	a) violation de format d'ID b) les identificateurs ne sont pas placés entre accolades, mais entre parenthèses c) RAI ou VI est manquant au niveau d'un élément d'un ensemble d'identificateurs – {BBB001_X##1, BBB002##1} – {0112/2///62656_2#BBB001, 0112/2///62656_2#BBB002} – {BBB001, BBB002}
valeur mixte ICID	MDC_P017 (Affectation de valeurs de classe) MDC_P096 (Classification de propriété) MDC_P097 (Exigence des propriétés)	– Affectation de valeurs de classe – {(0112/2///62656_2#BBB001##1, abc), (0112/2///62656_2#BBB002##1, efj)} – Classification de la propriété – {(0112/2///62656_2#BBB001##1,1,abc), (0112/2///62656_2#BBB002##1,2,efj)} – Exigence des propriétés – {(0112/2///62656_2#BBB001##1,KEY), (0112/2///62656_2#BBB001##1,NOT NULL)}	a) violation de format d'ID b) les identificateurs ne sont pas placés entre accolades, mais entre parenthèses c) RAI ou VI est manquant au niveau d'un élément d'un ensemble d'identificateurs – {(BBB001##1, abc),(BBB002##1, efj)} – {(0112/2///62656_2#BBB001,1,abc), (0112/2///62656_2#BBB002,2,efj)} – {(BBB001,KEY),(BBB001,NOT NULL)}

Tableau C.1 *(3 de 3)*

Valeur prévue	Attribut	Exemple de valeur valide	Cas de valeurs non valides

Type de données	MDC_P022 (Type de données)	– CLASS_REFERENCE et CLASS_INSTANCE
		  – CLASS_REFERENCE_TYPE(0112/2///62656_2#BBB001##1)
		– ENUM_TYPE
		  – ENUM_CODE_TYPE(0112/2///62656_2#BBB001##1)
		  – ENUM_CODE_TYPE(0112/2///62656_2#BBB001##1(a,b,c))
		– Type nommé
		  – NAMED_TYPE(0112/2///62656_2#BBB001##1)
		– Agrégation
		  – SET(0,?) OF LIST(2,2) OF STRING_TYPE
			a) violation de format d'ID
			b) RAI ou VI est manquant au niveau d'un identificateur spécifié
			  – CLASS_REFERENCE_TYPE(BBB001##1)
			  – ENUM_CODE_TYPE(BBB001(a,b,c))
			  – NAMED_TYPE(0112/2///62656_2#BBB001)
			c) caractère non valide pour la cardinalité d'agrégation
			  – SET[0,?] OF LIST[2,2] OF STRING_TYPE

date	MDC_P003_1 (Date de la définition originale)
	MDC_P003_2 (Date de la version actuelle)
	MDC_P003_3 (Date de la révision actuelle)
		– 2010-05-01	a) erreur de format
			  – 2010-5-1
			  – 2010/05/01

synonyme	MDC_P004_2 (Nom synonyme)	– {(motor,en)}
		– {(“display (computer)”,en), (“étalage (ordinateur)”,fr)}
		– {(“””computer””, display”,en), (“””ordinateur””, étalage”,fr)}
			a) violation de format d'agrégation
			  – ABC
			b) syntaxe de caractère d'échappement non valide
			  – {(display (computer),en)}
			  – {(“computer” display,en)}

Annexe D
(informative)

Échantillons de données

Pendant la période de développement de la norme, les paquets d'échantillons de données pour les dictionnaires de données décrits à l'Article 5 sont disponibles à l'URL suivante: <http://std.iec.ch/iec61360>.

Annexe E
(informative)

Outils de paquetage

Pour faciliter la lecture, les outils de paquetage réputés conformes à la présente partie de la CEI 62656 à la date de publication de la présente Spécification technique sont récapitulés dans le tableau ci-dessous. Pour connaître les conditions d'utilisation spécifiques, contacter l'URI pertinent du tableau.

Nom	Description	Contact
ParcelMaker™ [5]	La version communautaire est disponible gratuitement auprès de Toshiba Corporation pour les experts enregistrés de la CEI et de l'ISO.	parcel@sel.rdc.toshiba.co.jp

5 ParcelMaker™ est un nom de produit de Toshiba Corporation. Cette information est donnée à l'intention des utilisateurs du présent document et ne signifie nullement que la CEI approuve ou recommande le produit désigné. Des produits équivalents peuvent être utilisés s'il est démontré qu'ils conduisent aux mêmes résultats.

Bibliographie

[1] CEI 61360-4:1997, *Types normalisés d'éléments de données avec plan de classification pour composants électriques – Partie 4: Collection de référence CEI des types normalisés d'éléments de données, des classes de composants et des termes* Disponible à l'adresse: <http://std.iec.ch/iec61360>

[2] CEI 61970-301:2011*, Interface de programmation d'application pour système de gestion d'énergie (EMS-API) – Partie 301: Base de modèle d'information commun (CIM)*

[3] IEC 62656-3:—[6], *Standardized product ontology register and transfer by spreadsheets – Part 3: Interface for common information model* (disponible en anglais seulement)

[4] IEC 61360-6:—[7], *Quality guide* (disponible en anglais seulement)

[5] ISO/CEI 6523 (toutes les parties), *Technologies de l'information – Structure pour l'identification des organisations et des parties d'organisations*

[6] ISO 10303-41:2005, *Systèmes d'automatisation industrielle et intégration – Représentation et échange de données de produits – Partie 41: Ressources génériques intégrées: Principes de description et de support de produits*

[7] ISO 8879:1986, *Traitement de l'information – Systèmes bureautiques – Langage normalisé de balisage généralisé (SGML)*

[8] Mathematical Markup Language (MathML). Third Edition. World Wide Web Consortium Recommendation 21 October 2010.
Disponible à l'adresse: <http://www.w3.org/TR/MathML3/>

[9] OMG. *OMG Unified Modeling Language (OMG UML), Infrastructure, V2.3*. Needham, MA: OMG (Object Management Group Inc.), 3 May 2010.
Disponible à l'adresse: <http://www.omg.org/spec/UML/2.3/Infrastructure/PDF>

[10] NETWORK WORKING GROUP. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. Network Working Group, 2005-10.
Disponible à l'adresse: <http://www.ietf.org/rfc/rfc4180.txt>

[11] OMG. Meta Object Facility (MOF) Core Specification. OMG Available Specification, version 2.0. Needham, MA: OMG (Object Management Group Inc.), 1 January 2006. Disponible à l'adresse: <http://www.omg.org/spec/MOF/2.0>

[12] ISO/IEC 8824-1:2002, *Technologies de l'information – Notation de syntaxe abstraite numéro un (ASN.1): Spécification de la notation de base*

_____

6 À l'étude.

7 À l'étude.

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

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