PD IEC/TS 62656-2:2013

BSI Standards Publication

Standardized product ontology register and transfer by spreadsheets

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

National foreword

This Published Document is the UK implementation of IEC/TS 62656-2:2013.

The UK participation in its preparation was entrusted to Technical Committee GEL/3, Documentation and graphical symbols.

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

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

Published by BSI Standards Limited 2013

ISBN 978 0 580 75061 8
ICS 01.040.01; 01.110

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

This Published Document was published under the authority of the Standards Policy and Strategy Committee on 31 October 2013.

Amendments/corrigenda issued since publication

Date	Text affected

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

Warning! Make sure that you obtained this publication from an authorized distributor.
Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.

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.

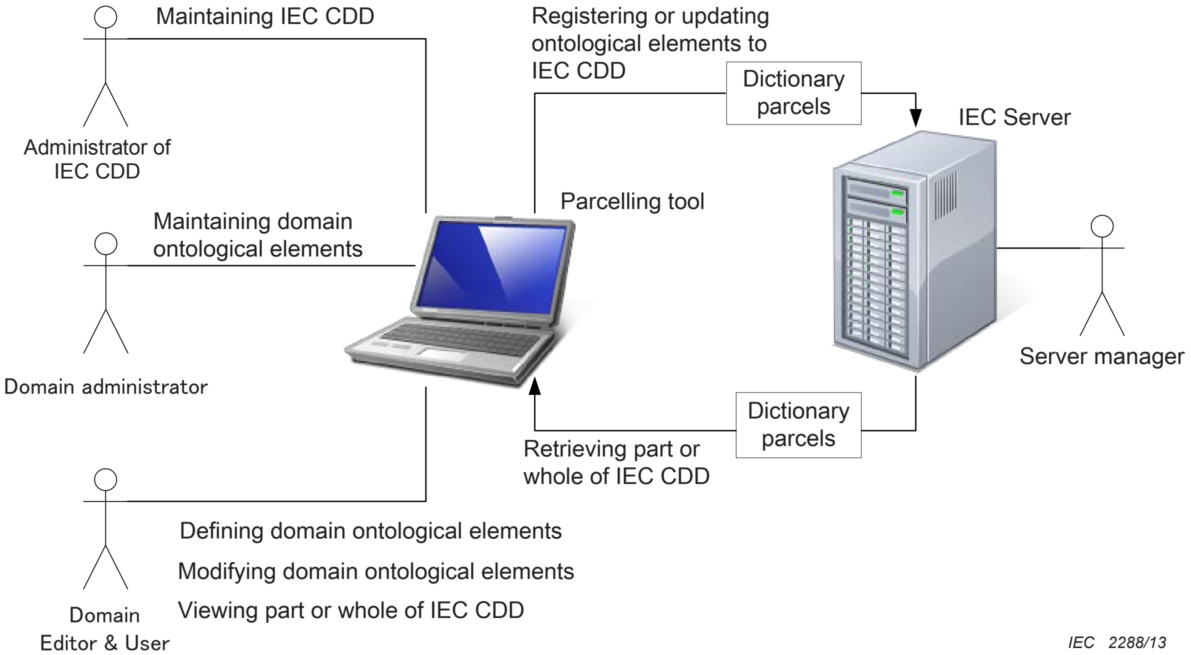

Maintaining IEC CDD
Administrator of IEC CDD
Registering or updating ontological elements to IEC CDD
Dictionary parcels
IEC Server
Parcelling tool
Maintaining domain ontological elements
Domain administrator
Server manager
Dictionary parcels
Retrieving part or whole of IEC CDD
Defining domain ontological elements
Modifying domain ontological elements
Viewing part or whole of IEC CDD
Domain Editor & User
IEC 2288/13

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.

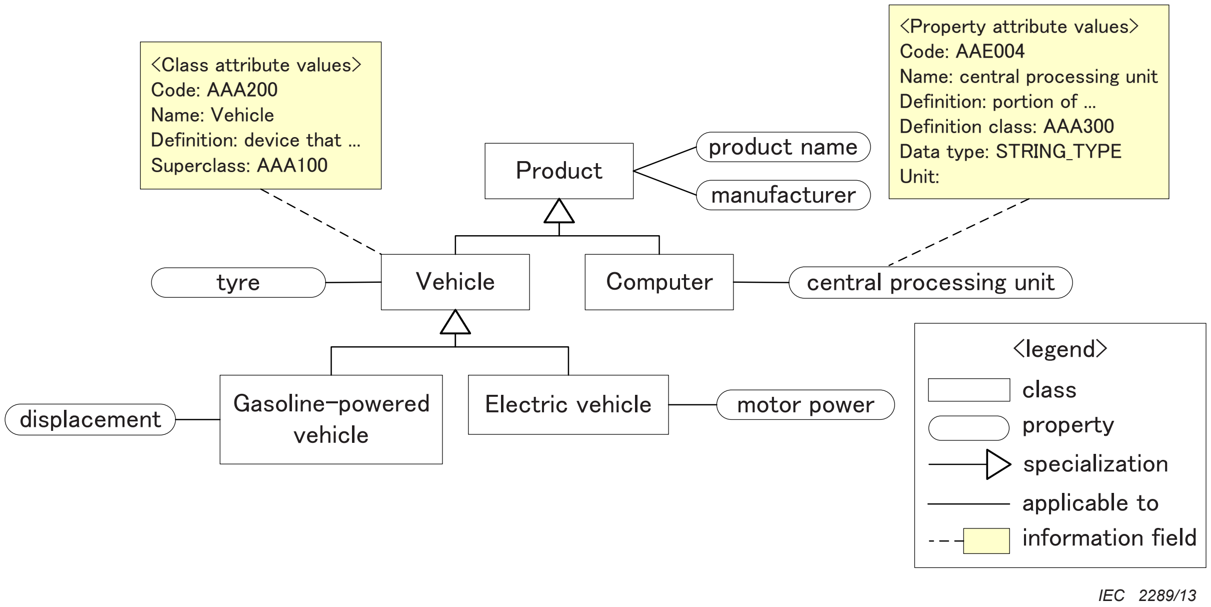


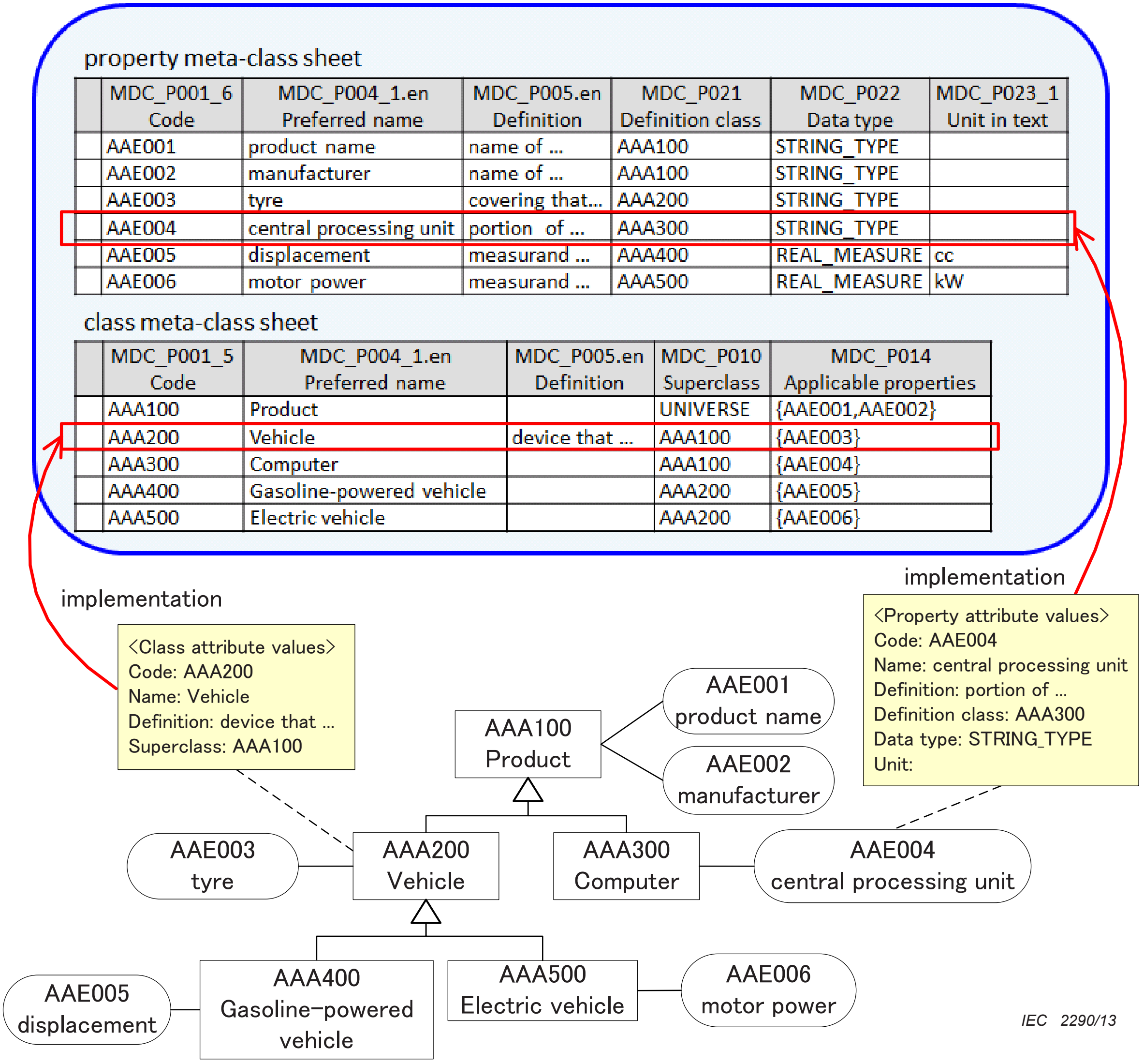
Figure 2 – Data dictionary

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).



property meta-class sheet

MDC_P001_6 Code	MDC_P004_1.en Preferred name	MDC_P005.en Definition	MDC_P021 Definition class	MDC_P022 Data type	MDC_P023_1 Unit in text
AAE001	product name	name of …	AAA100	STRING_TYPE	
AAE002	manufacturer	name of …	AAA100	STRING_TYPE	
AAE003	tyre	covering that…	AAA200	STRING_TYPE	
AAE004	central processing unit	portion of …	AAA300	STRING_TYPE	
AAE005	displacement	measurand …	AAA400	REAL_MEASURE	cc
AAE006	motor power	measurand …	AAA500	REAL_MEASURE	kW

class meta-class sheet

MDC_P001_5 Code	MDC_P004_1.en Preferred name	MDC_P005.en Definition	MDC_P010 Superclass	MDC_P014 Applicable properties
AAA100	Product		UNIVERSE	{AAE001,AAE002}
AAA200	Vehicle	device that …	AAA100	{AAE003}
AAA300	Computer		AAA100	{AAE004}
AAA400	Gasoline-powered vehicle		AAA200	{AAE005}
AAA500	Electric vehicle		AAA200	{AAE006}

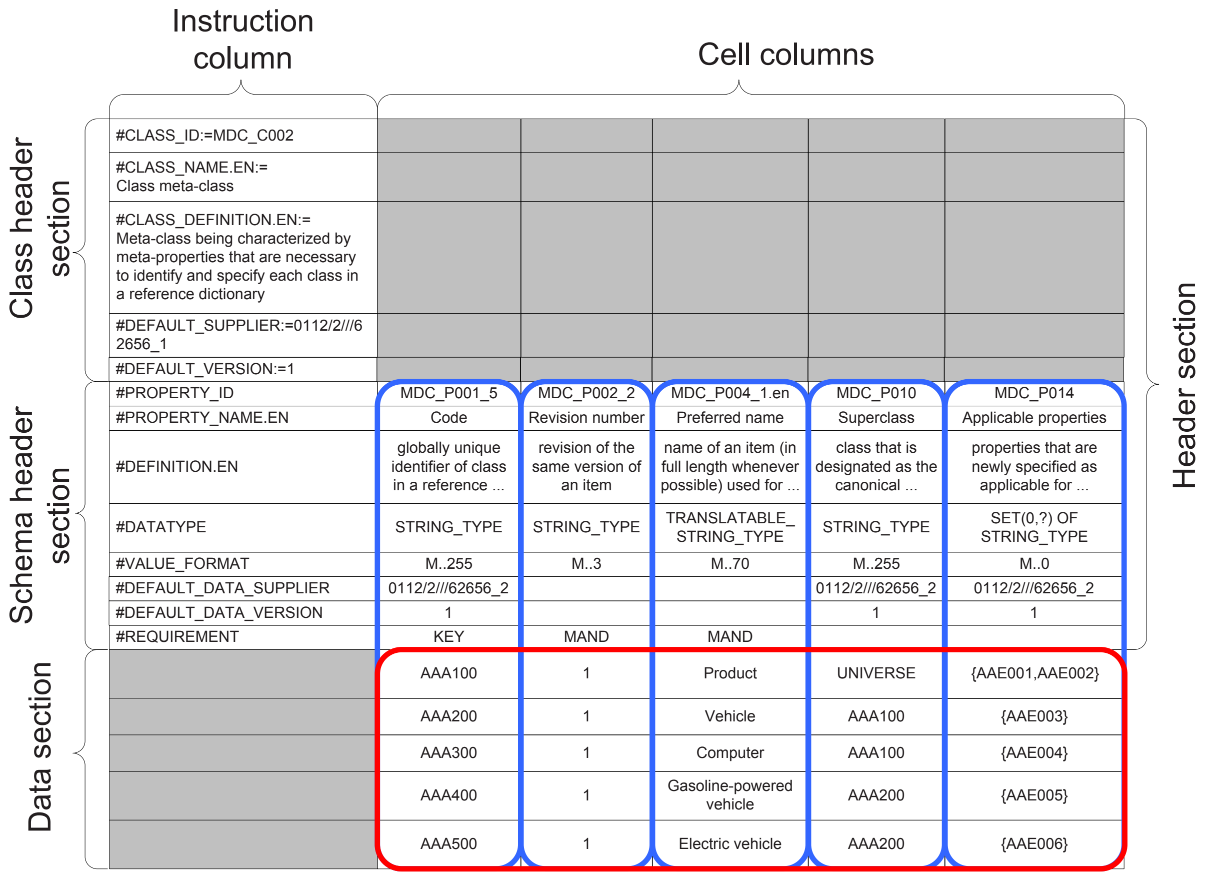
IEC 2290/13

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.



#CLASS_ID:=MDC_C002					
#CLASS_NAME.EN:= Class meta-class					
#CLASS_DEFINITION.EN:= Meta-class being characterized by meta-properties that are necessary to identify and specify each class in a reference dictionary					
#DEFAULT_SUPPLIER:=0112/2///62656_1					
#DEFAULT_VERSION:=1					
#PROPERTY_ID	MDC_P001_5	MDC_P002_2	MDC_P004_1.en	MDC_P010	MDC_P014
#PROPERTY_NAME.EN	Code	Revision number	Preferred name	Superclass	Applicable properties
#DEFINITION.EN	globally unique identifier of class in a reference ...	revision of the same version of an item	name of an item (in full length whenever possible) used for ...	class that is designated as the canonical ...	properties that are newly specified as applicable for ...
#DATATYPE	STRING_TYPE	STRING_TYPE	TRANSLATABLE_STRING_TYPE	STRING_TYPE	SET(0,?) OF STRING_TYPE
#VALUE_FORMAT	M..255	M..3	M..70	M..255	M..0
#DEFAULT_DATA_SUPPLIER	0112/2///62656_2			0112/2///62656_2	0112/2///62656_2
#DEFAULT_DATA_VERSION	1			1	1
#REQUIREMENT	KEY	MAND	MAND		
	AAA100	1	Product	UNIVERSE	{AAE001,AAE002}
	AAA200	1	Vehicle	AAA100	{AAE003}
	AAA300	1	Computer	AAA100	{AAE004}
	AAA400	1	Gasoline-powered vehicle	AAA200	{AAE005}
	AAA500	1	Electric vehicle	AAA200	{AAE006}

IEC 2291/13

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,

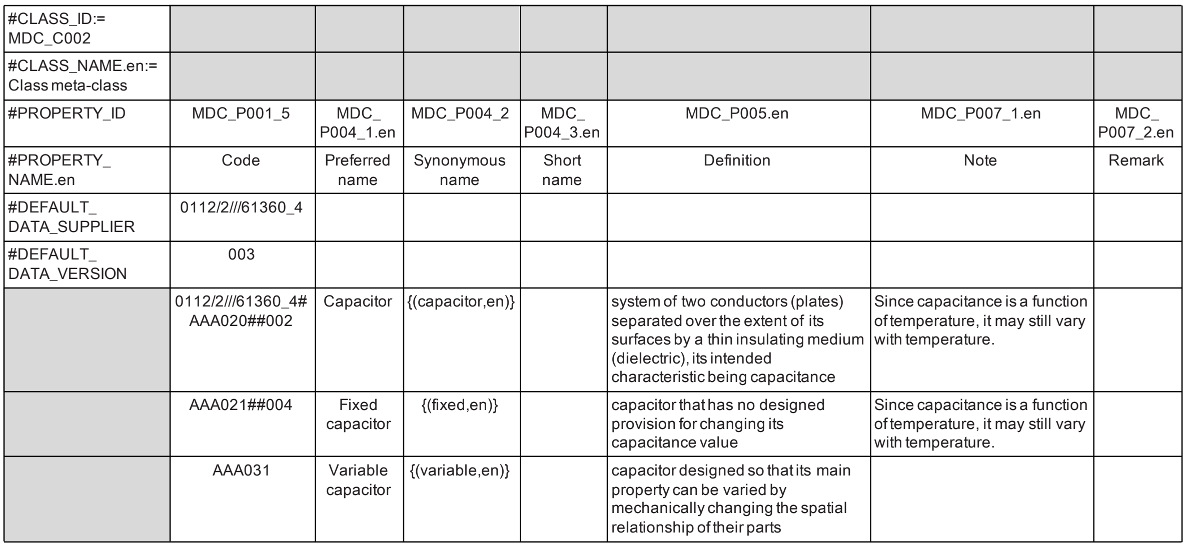
– reducing data size,

– 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.

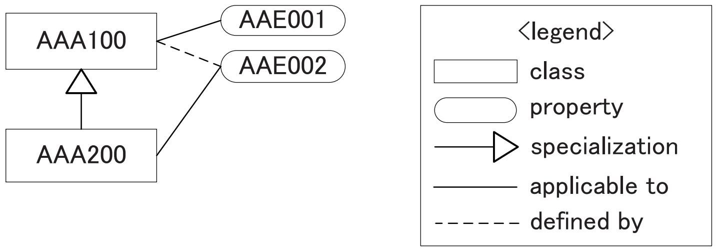
To avoid replication of the same property, data type and document in several branches, each of such ontological elements which may be commonly used by several classes should be defined within their common general class.

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

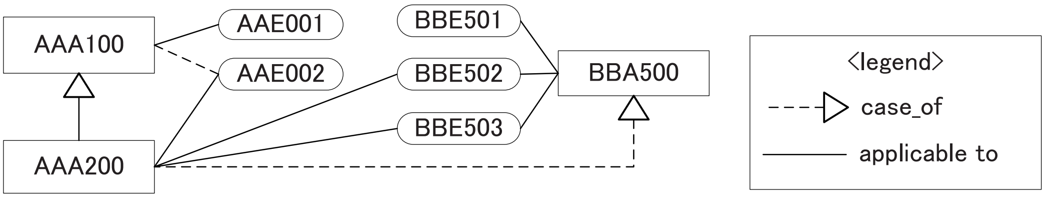
Figure 8 – Parcel implementation for simple classification trees

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.

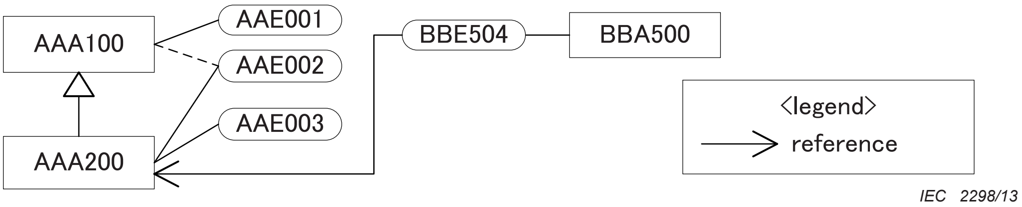


Figure 11 – Composition relationship between two branches

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.

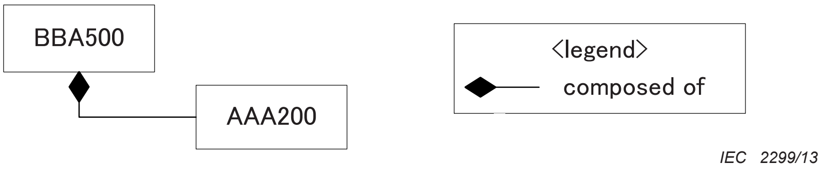
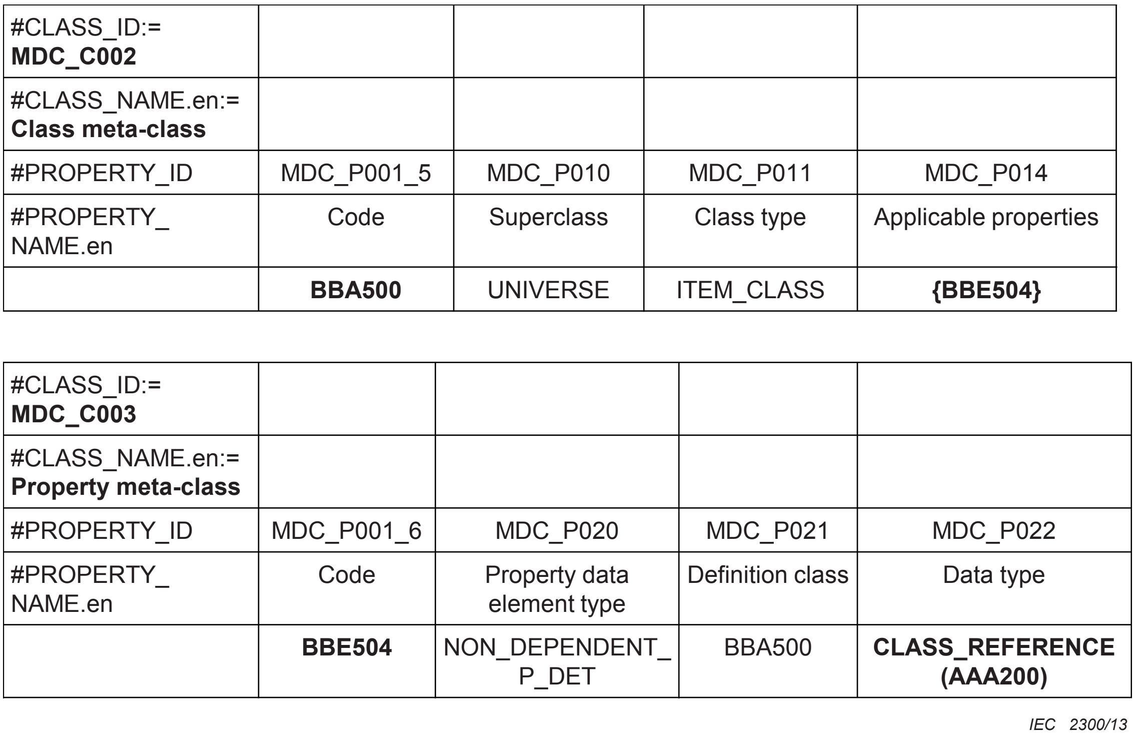


Figure 12 – Example of a composition tree

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.

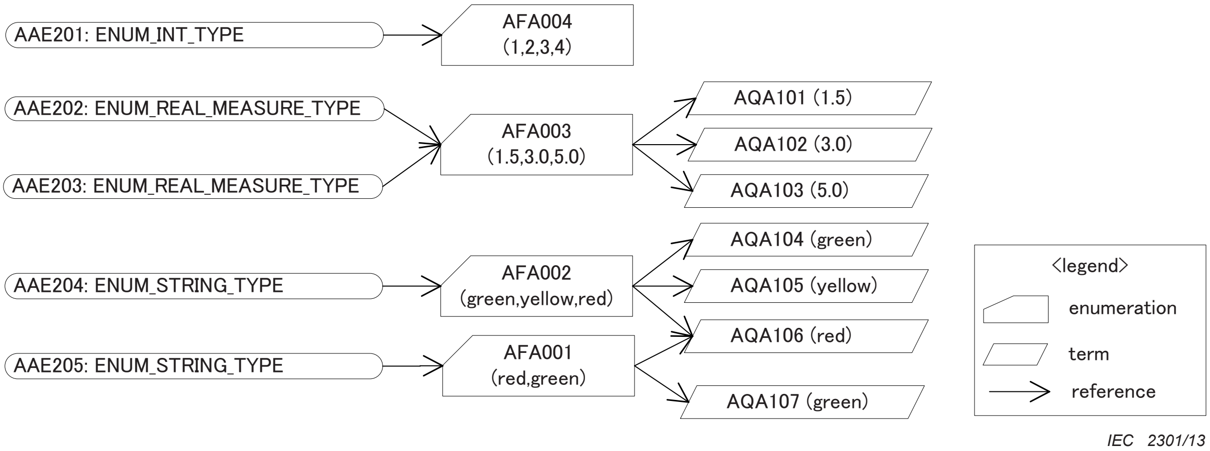
The attribute MDC_P043 (Enumerated list of terms) of the enumeration parcel specifies a list of the identifiers of terms which may be applied to properties as their value candidates. If identifiers of terms are specified, values which are described in the attribute MDC_P044 shall be a list of values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



IEC 2301/13

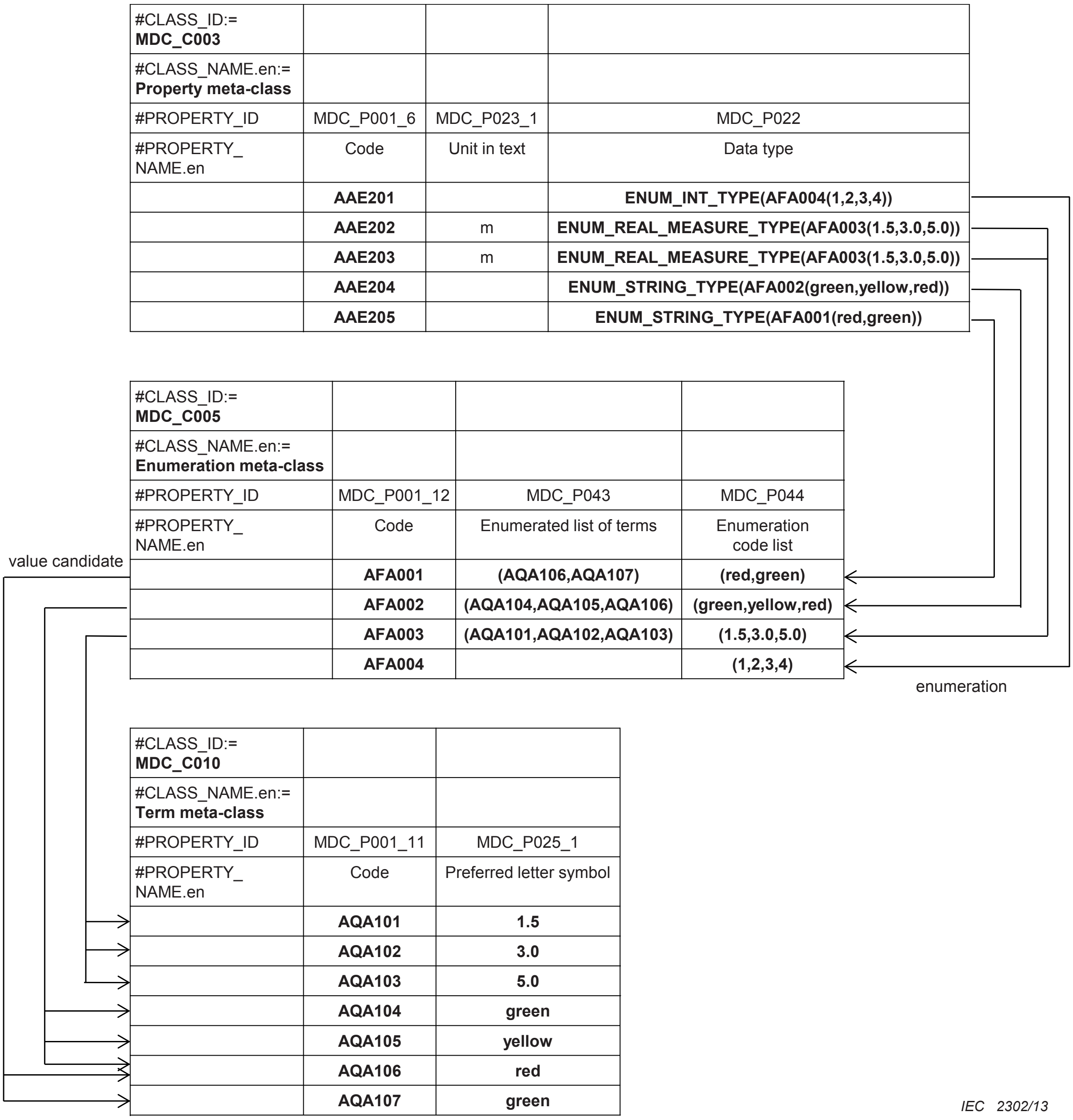
Figure 14 – Example of a use case of enumeration

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).



#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)**

#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

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).

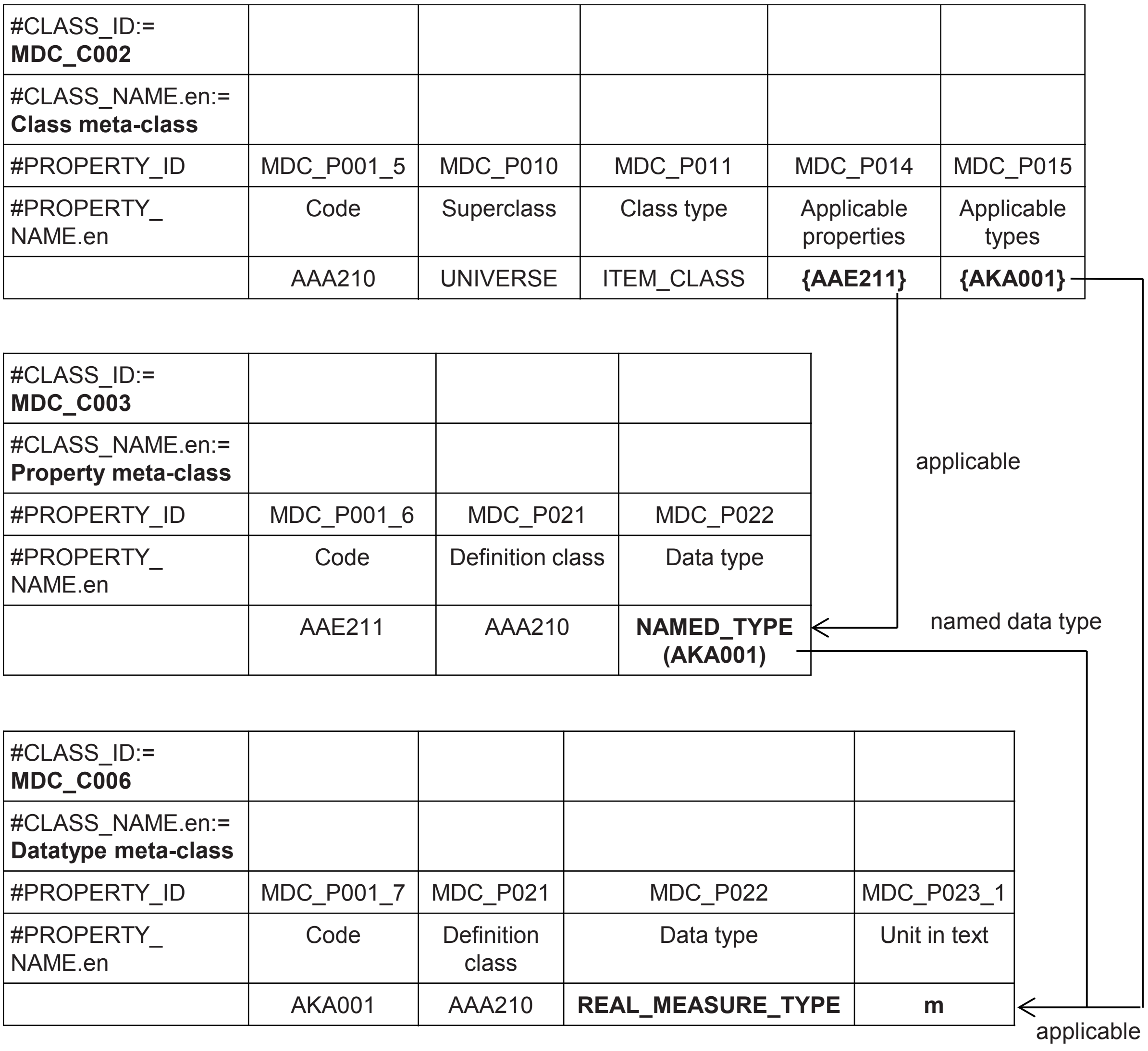
In a data type parcel sheet, each named data type shall be described in each line in the data section. Each named data type shall have its base data type which is either a predefined data type in IEC 62656-1 or another named data type. It is assumed that any reference between named data types and nested 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}**

#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**

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.

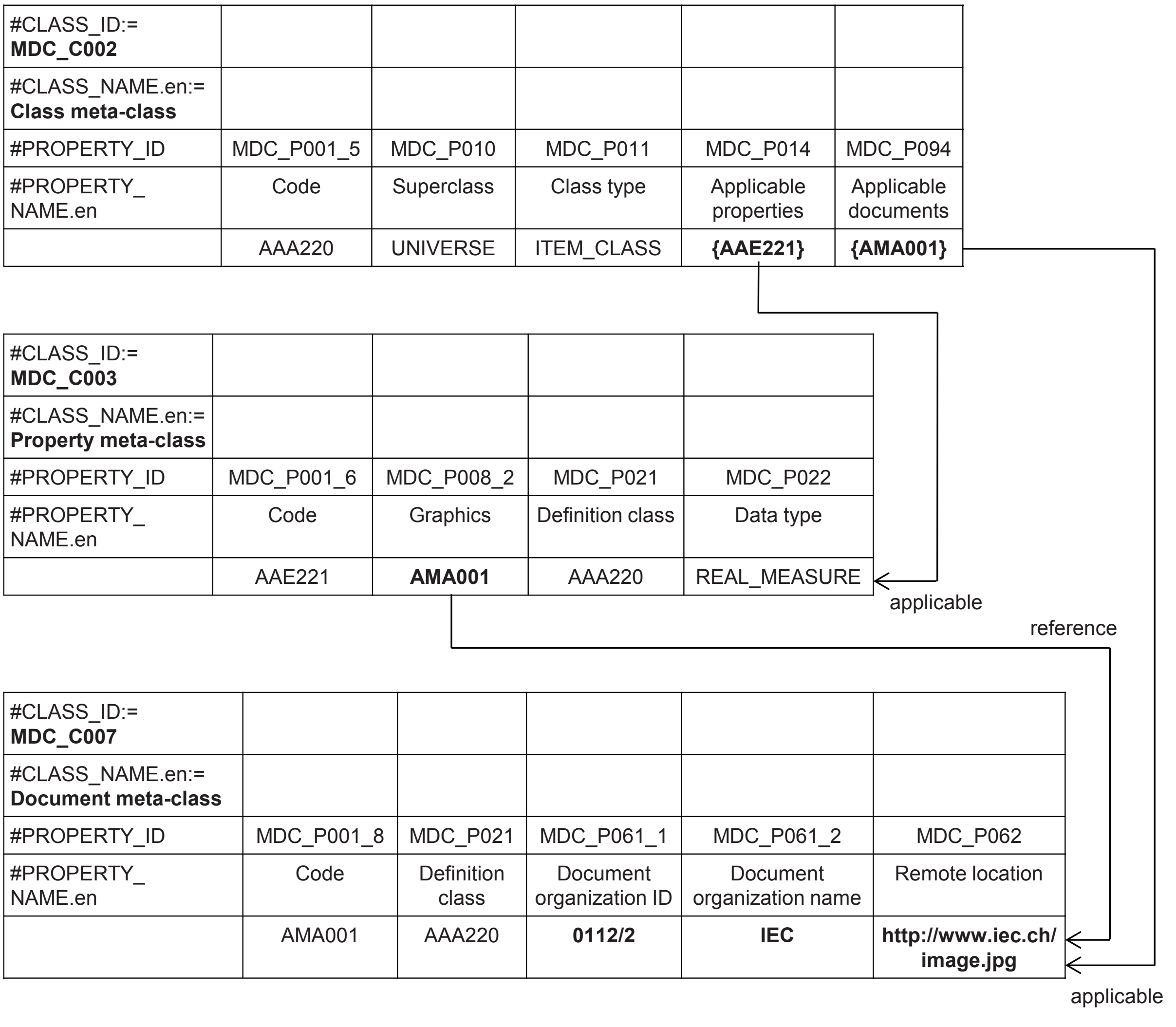
In a document parcel sheet, information regarding each outer resource is described in each line. Such information includes details of the organization that has responsibility to a document, how to access a document, etc.

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.



#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**

IEC 2304/13

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.

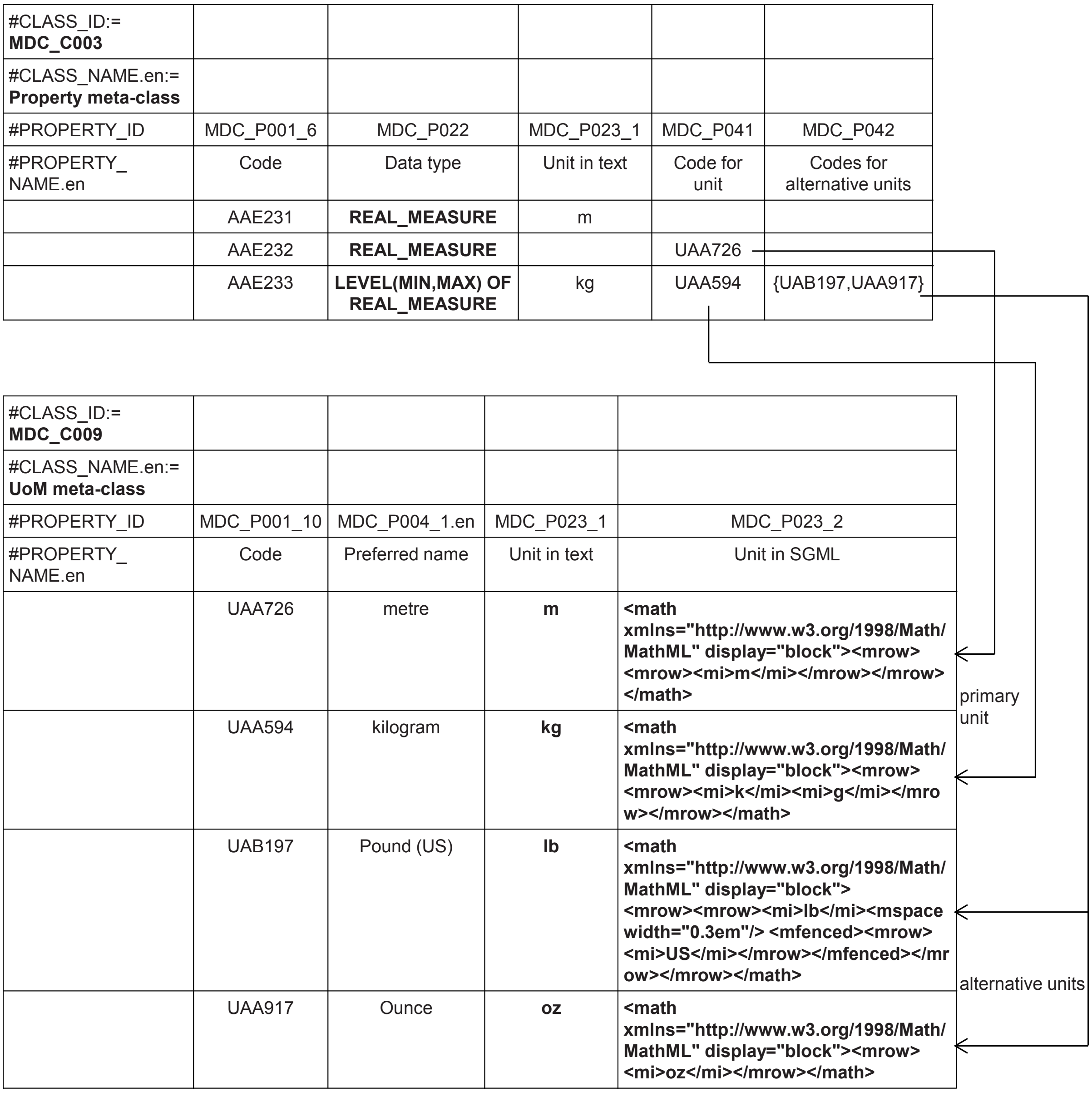
IEC 62720, which contains a unit dictionary for the whole electric and electronic domains, is a good example of the above requirement.

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>**

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).

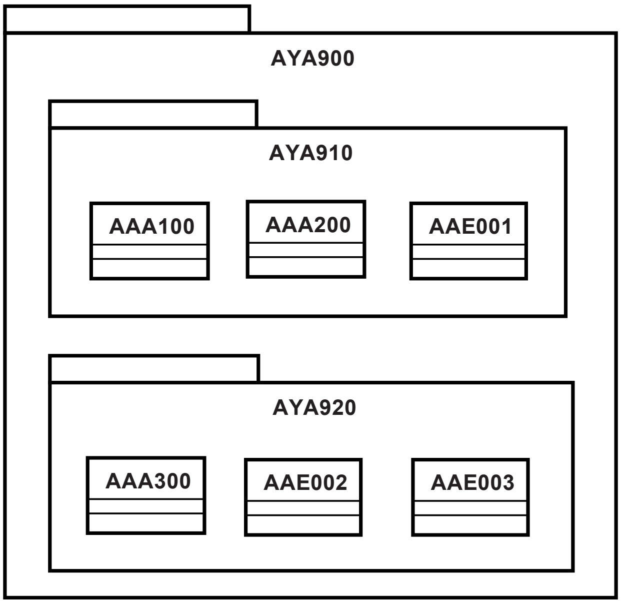
The attribute MDC_P208 (Type of the domain) is also available, if all ontological elements contained in the domain of a relation are of the same type as the parcel. In this case, the identifier of such a parcel shall be specified as a value of the attribute. Otherwise, the value of the attribute MDC_P208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



IEC 2306/13

Figure 19 – UML package diagram by relations

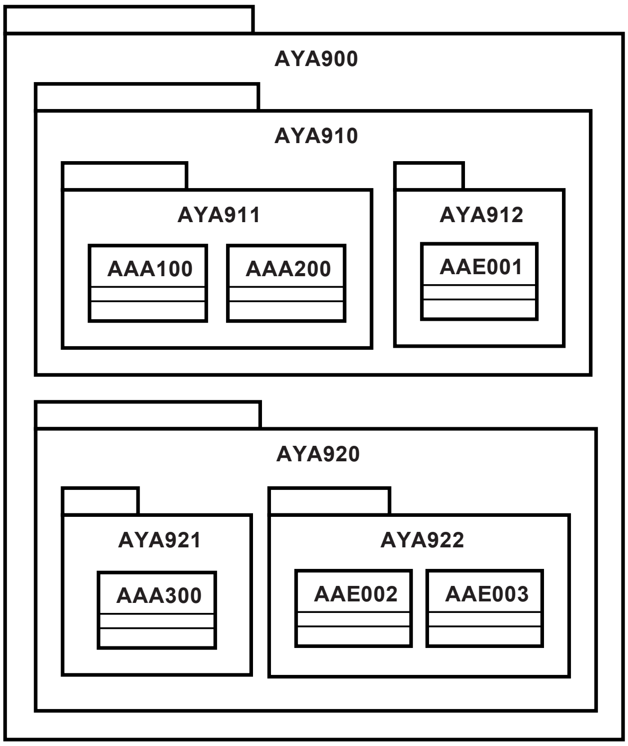
Figure 20 shows a relation parcel sheet as an implementation example for the UML package depicted in Figure 19 by predicate relations. In this example, the identifiers of the classes and properties in the packages AYA910 and AYA920 are simply specified in the attribute MDC_P201 of each package. The packages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

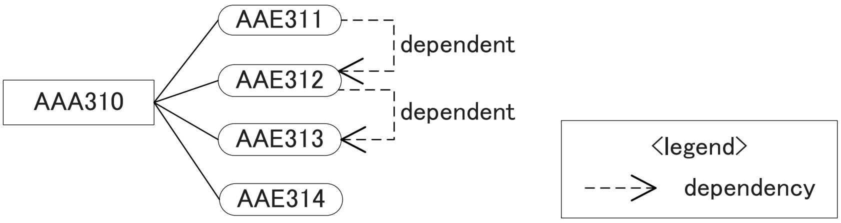
The attribute MDC_P020 (Property data element type) of the property parcel is an attribute for specifying the dependency of a property, which means i) whether that property is a condition property for another property, and ii) whether that property is dependent on another property. Table 1 shows what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

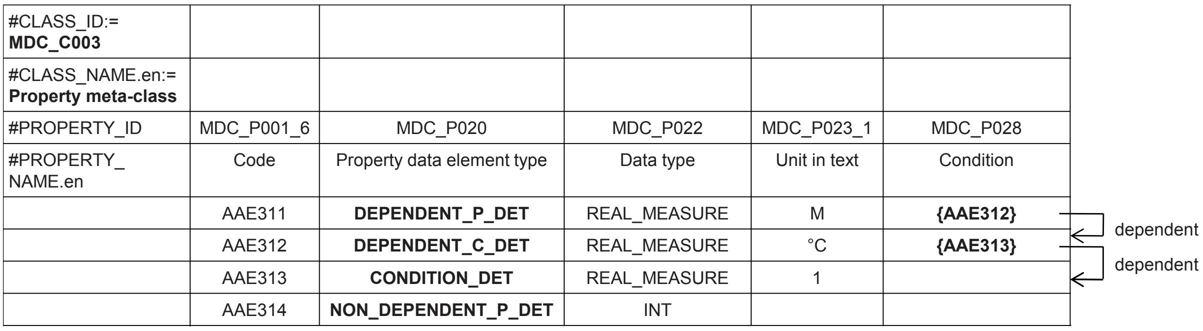
Figure 23 shows an example of a data dictionary including condition properties in which the class AAA310 has four properties. The property AAE311 depends on the property AAE312, and the property AAE312 also depends on the property AAE313.



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.

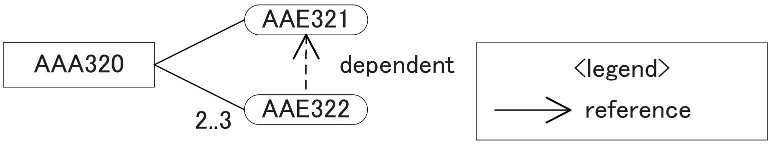
LIST is the most recommended data type for describing cardinality, because in many cases, the order of the items is recognized implicitly or explicitly. In the case that the order of the items is not actually needed, BAG may be also used.

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

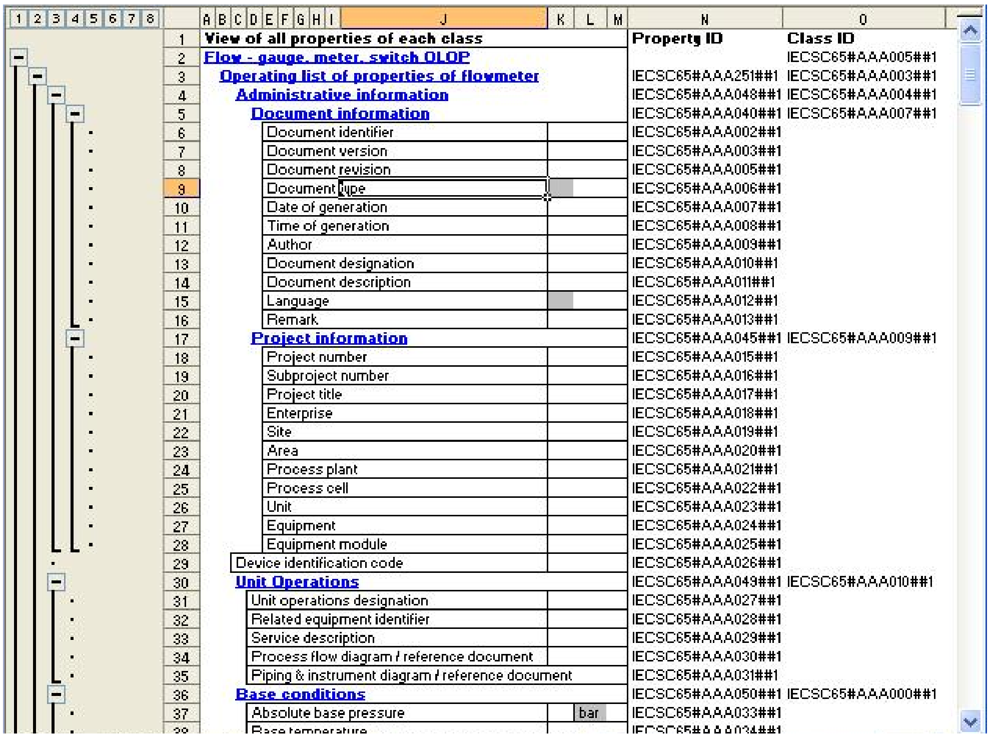
8.3 Implementation of blocks and lists of properties (LOPs)

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.



	A B C D E F G H I J	K	L	M	N	O
1	**View of all properties of each class**				**Property ID**	**Class ID**
2	Flow - gauge, meter, switch OLOP					IECSC65#AAA005##1
3	Operating list of properties of flowmeter				IECSC65#AAA251##1	IECSC65#AAA003##1
4	Administrative information				IECSC65#AAA048##1	IECSC65#AAA004##1
5	Document information				IECSC65#AAA040##1	IECSC65#AAA007##1
6	Document identifier				IECSC65#AAA002##1	
7	Document version				IECSC65#AAA003##1	
8	Document revision				IECSC65#AAA005##1	
9	Document type				IECSC65#AAA006##1	
10	Date of generation				IECSC65#AAA007##1	
11	Time of generation				IECSC65#AAA008##1	
12	Author				IECSC65#AAA009##1	
13	Document designation				IECSC65#AAA010##1	
14	Document description				IECSC65#AAA011##1	
15	Language				IECSC65#AAA012##1	
16	Remark				IECSC65#AAA013##1	
17	Project information				IECSC65#AAA045##1	IECSC65#AAA009##1
18	Project number				IECSC65#AAA015##1	
19	Subproject number				IECSC65#AAA016##1	
20	Project title				IECSC65#AAA017##1	
21	Enterprise				IECSC65#AAA018##1	
22	Site				IECSC65#AAA019##1	
23	Area				IECSC65#AAA020##1	
24	Process plant				IECSC65#AAA021##1	
25	Process cell				IECSC65#AAA022##1	
26	Unit				IECSC65#AAA023##1	
27	Equipment				IECSC65#AAA024##1	
28	Equipment module				IECSC65#AAA025##1	
29	Device identification code				IECSC65#AAA026##1	
30	Unit Operations				IECSC65#AAA049##1	IECSC65#AAA010##1
31	Unit operations designation				IECSC65#AAA027##1	
32	Related equipment identifier				IECSC65#AAA028##1	
33	Service description				IECSC65#AAA029##1	
34	Process flow diagram / reference document				IECSC65#AAA030##1	
35	Piping & instrument diagram / reference document				IECSC65#AAA031##1	
36	Base conditions				IECSC65#AAA050##1	IECSC65#AAA000##1
37	Absolute base pressure		bar		IECSC65#AAA033##1	
38	Base temperature				IECSC65#AAA034##1	

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.

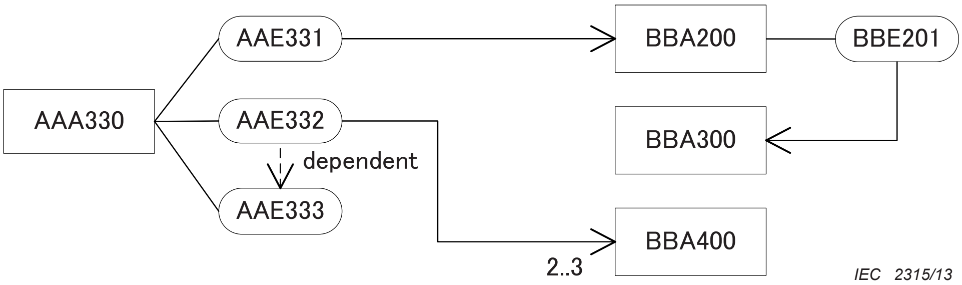
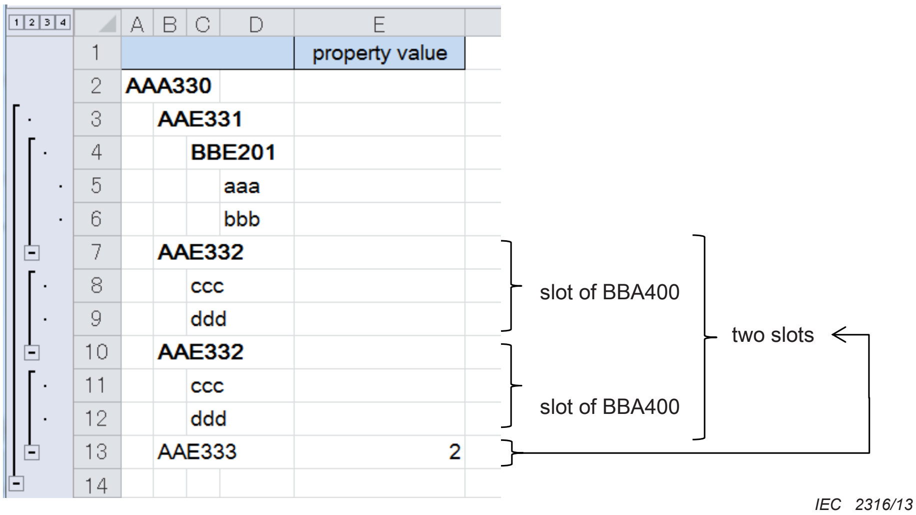


Figure 28 – Example of use case of blocks

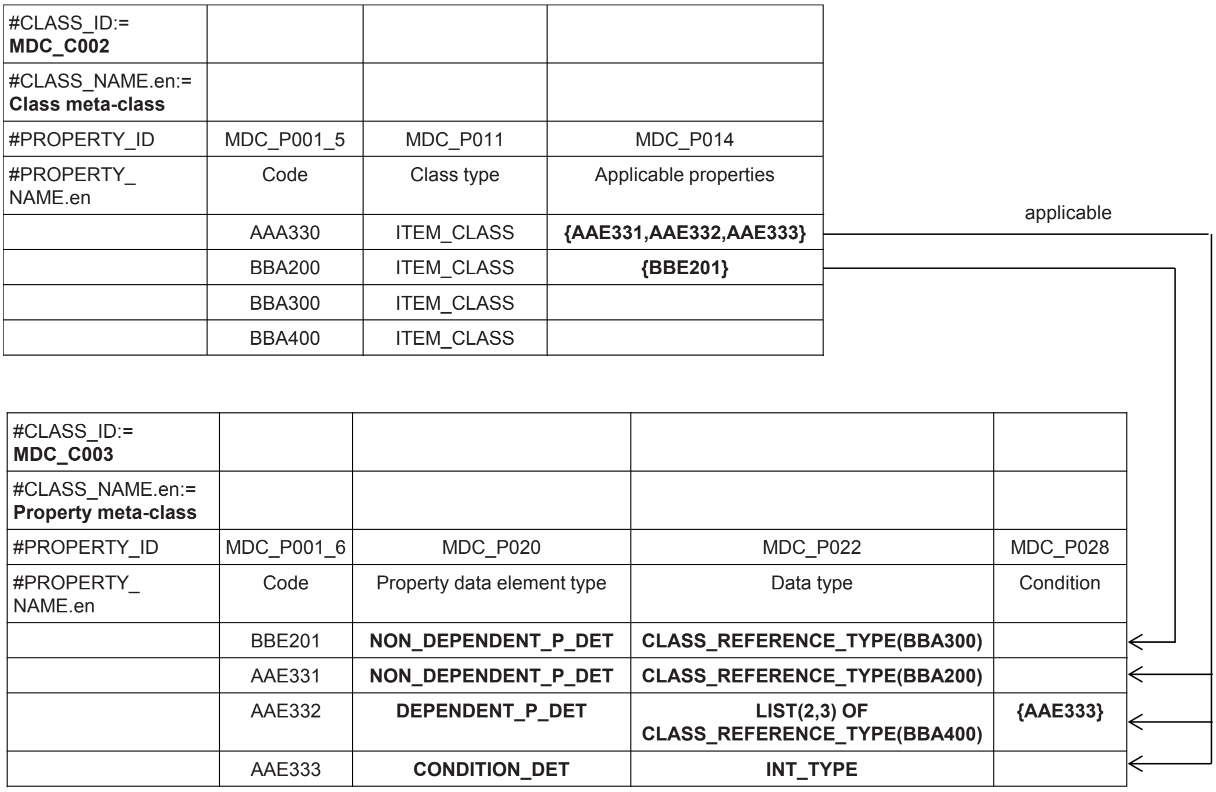
Figure 29 shows an example of the composition view of AAA330 at Domain Library layer representing the structure within Figure 28. It illustrates how to use the control property AAE333 at Domain Library layer. Each line depicted by bold letters is a block, while each line depicted in plain letters is 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.



	A	B	C	D	E
1					property value
2	**AAA330**				
3		**AAE331**			
4			**BBE201**		
5				aaa	
6				bbb	
7		**AAE332**			
8			ccc		
9			ddd		
10		**AAE332**			
11			ccc		
12			ddd		
13		AAE333			2
14					

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	

#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.

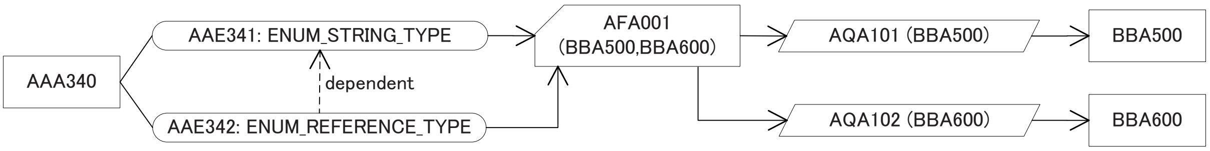
NOTE Traditionally, in the IEC CDD, properties are directly assigned to a class, each of which is of CLASS_REFERENCE_TYPE or of CLASS_INSTANCE_TYPE and points to a class to be selected as a polymorphic choice. In this case, for grouping those properties and for giving a function to select one of the 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.

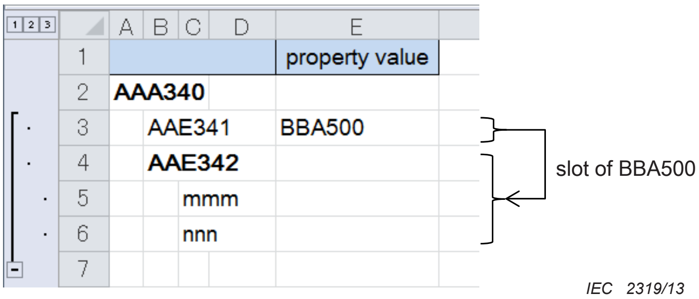
NOTE2 Traditionally, in the above example, two properties are assigned to AAA340, which are of CLASS_REFERENCE_TYPE and point to BBA500 and BBA600, respectively, as polymorphic choices. In this case, those properties depend on AAE341 as their control property.



IEC 2318/13

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.



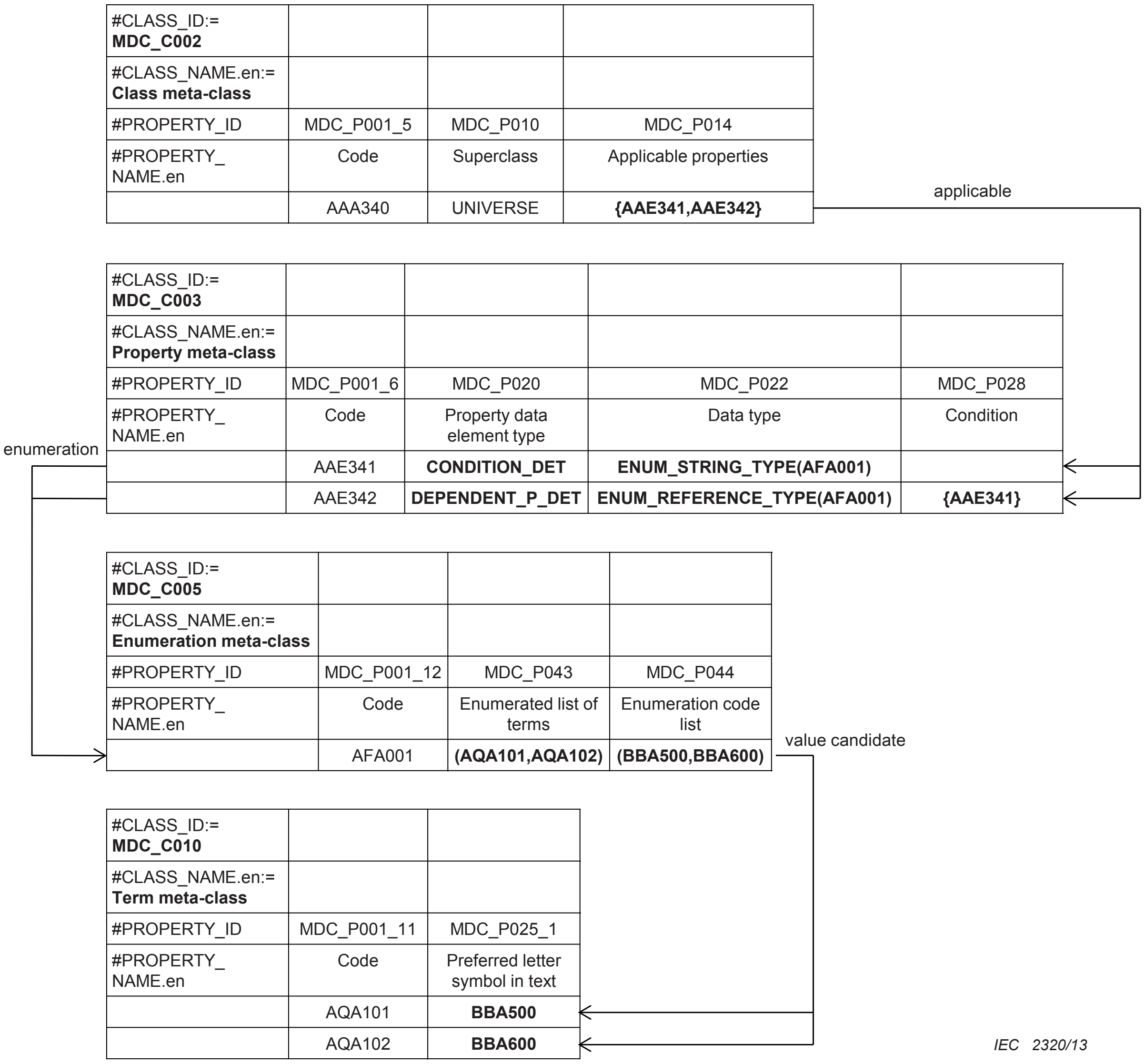
	A	B	C	D	E
1					property value
2	**AAA340**				
3		AAE341			BBA500
4		**AAE342**			
5			mmm		
6			nnn		
7					

slot of BBA500

IEC 2319/13

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.



#CLASS_ID:= **MDC_C002**			
#CLASS_NAME.en:= **Class meta-class**			
#PROPERTY_ID	MDC_P001_5	MDC_P010	MDC_P014
#PROPERTY_ NAME.en	Code	Superclass	Applicable properties
	AAA340	UNIVERSE	**{AAE341,AAE342}**

#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
	AAE341	**CONDITION_DET**	**ENUM_STRING_TYPE(AFA001)**	
	AAE342	**DEPENDENT_P_DET**	**ENUM_REFERENCE_TYPE(AFA001)**	**{AAE341}**

#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	**(AQA101,AQA102)**	**(BBA500,BBA600)**

#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 in text
	AQA101	**BBA500**
	AQA102	**BBA600**

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.

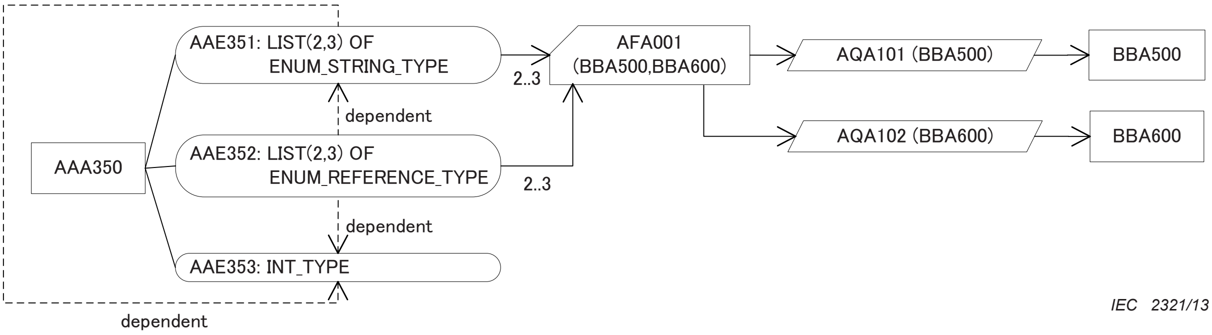
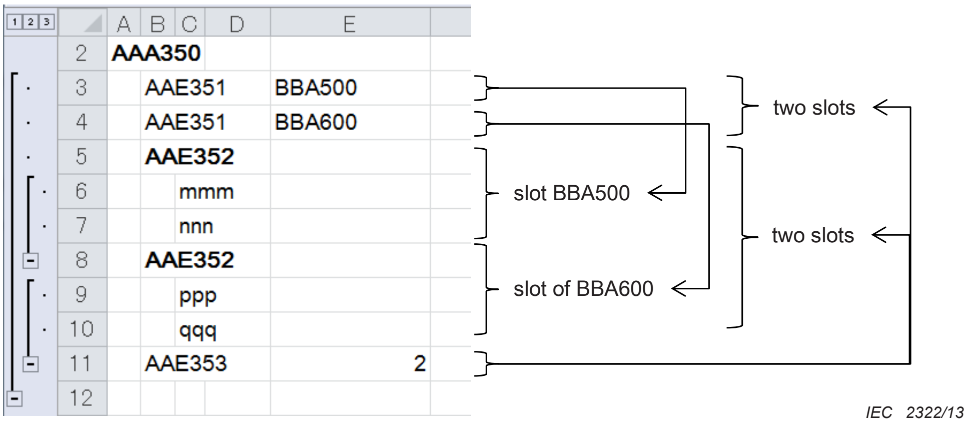


Figure 34 – Example of a use case of polymorphism with multiple choices

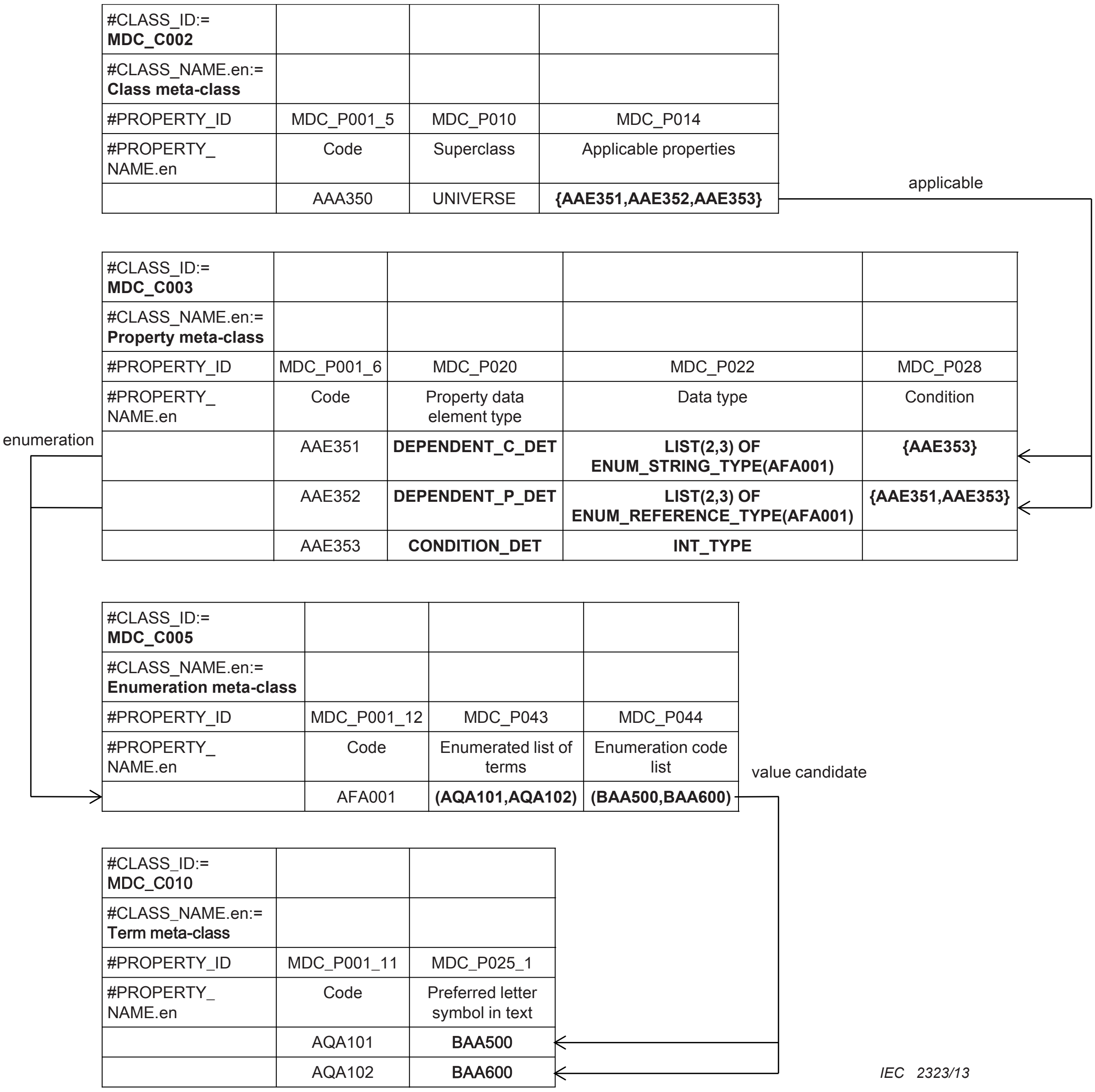
Figure 35 shows an example of the composition view of AAA350 at Domain Library layer to represent the structure shown in Figure 34. It also illustrates how to utilise the control property AAE353 at Domain Library layer. Each line depicted by bold letters is a block, while each line depicted by plain 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.



	A	B	C	D	E
2	**AAA350**				
3		AAE351			BBA500
4		AAE351			BBA600
5		**AAE352**			
6			mmm		
7			nnn		
8		**AAE352**			
9			ppp		
10			qqq		
11		AAE353			2
12					

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.



#CLASS_ID:= **MDC_C002**			
#CLASS_NAME.en:= **Class meta-class**			
#PROPERTY_ID	MDC_P001_5	MDC_P010	MDC_P014
#PROPERTY_ NAME.en	Code	Superclass	Applicable properties
	AAA350	UNIVERSE	**{AAE351,AAE352,AAE353}**

#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
	AAE351	**DEPENDENT_C_DET**	**LIST(2,3) OF ENUM_STRING_TYPE(AFA001)**	**{AAE353}**
	AAE352	**DEPENDENT_P_DET**	**LIST(2,3) OF ENUM_REFERENCE_TYPE(AFA001)**	**{AAE351,AAE353}**
	AAE353	**CONDITION_DET**	**INT_TYPE**	

#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	**(AQA101,AQA102)**	**(BAA500,BAA600)**

#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 in text
	AQA101	**BAA500**
	AQA102	**BAA600**

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.

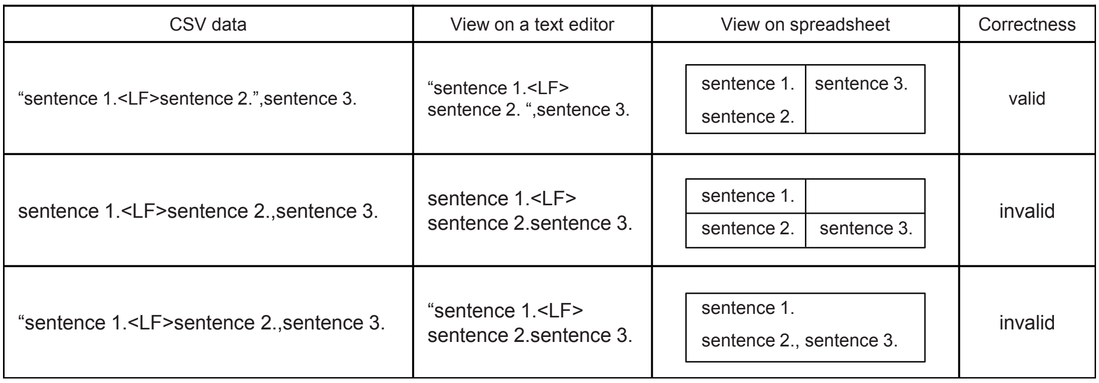
NOTE The number of text qualifiers in a line are always even in a CSV file. Thus, if the number of text qualifiers in a line is odd, the line is concatenated with the line feed character and next line in order.

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)*

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

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_CON STRAINT\(element_pattern\))
                              |(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.

This page deliberately left blank

This page deliberately left blank

British Standards Institution (BSI)

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

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

About us

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

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

Information on standards

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

Buying standards

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

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

Subscriptions

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

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

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

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

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

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

Revisions

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

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

Copyright

All the data, software and documentation set out in all British Standards and other BSI publications are the property of and copyrighted by BSI, or some person or entity that owns copyright in the information used (such as the international standardization bodies) and has formally licensed such information to BSI for commercial publication and use. Except as permitted under the Copyright, Designs and Patents Act 1988 no extract may be reproduced, stored in a retrieval system or transmitted in any form or by any means – electronic, photocopying, recording or otherwise – without prior written permission from BSI. Details and advice can be obtained from the Copyright & Licensing Department.

Useful Contacts:

Customer Services
Tel: +44 845 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 845 086 9001
Email: subscriptions@bsigroup.com

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

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

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

bsi.

...making excellence a habit.™