Technical Specification 110

ECLASS Serialization as RDF

PART 1

ECLASS e.V.

April 22, 2024

Authors and contributors

CRD adhoc group - W3C / WoT / TD & BCON² GmbH

Dr. Oliver Drumm, Siemens AG

Dr. Christian Block, ECLASS Head Office / BCON² GmbH

Dr. Sebastian Käbisch, Siemens AG

Dr. Darko Anicic, Siemens AG

Stefan Willms, morphe* Information Science

Andreas Neumann, Siemens Energy Global GmbH & Co. KG

Andreas Maisenbacher, Festo SE & Co. KG

Dr. Samuel Suhas Singapogu, Schneider Electric SE

Markus Füsi, Paradine GmbH

Nikolaus Ondracek, Paradine GmbH

Artur Bondza, Pepperl+Fuchs SE

Please send remarks to info@eclass.eu

Revision History

Date	Version	Description	Who?
April 22, 2024	1.0	Published	Block

Foreword

ECLASS is a formal semantic dictionary which is used for product description and product classification. In this document the RDF representation of ECLASS is described. It describes the basic mapping of ECLASS into triples using RDF. A CRD ad hoc group has evaluated different concepts over several years. For this purpose, external advice was acquired from the companies OntoText and Derivo. Special thanks go to Siemens AG for their support of this project.

This specification is structured as series containing different parts:

- Part 1: ECLASS Base Model & ECLASS Content
- Part 2: ECLASS Validation with SHACL & SPARQL
- Part 3: ECLASS RDF Usage
  - Part 3.a: RDF product description based on ECLASS
  - Part 3.b: Usage of ECLASS RDF in W3C Web of Things (WoT) Thing Description
  - Part 3.c: Usage of ECLASS RDF in AAS

In this document Part 1 is described.

Table of contents

List of Figures

List of Tables

Abbreviations

AAS .....Asset Administration Shell

AC .....Application Class

API .....Application Programming Interface

CDP .....ContentDevelopmentPlatform

CSV .....Comma-Separated-Values

IRDI .....International Registration Data Identifier

JSON .....JavaScript Object Notation

OPC-UA.....Open Platform Communications United Architecture

OWL.....Web Ontology Language

RDF .....Resource Description Framework

URI .....Uniform Resource Identifier

URL .....Uniform Resource Locator

W3C.....World Wide Web Consortium

WoT .....Web of Things

XML .....Extensible Markup Language

XSD .....XML Schema Definition

1 Introduction

The objective of ECLASS is to simplify the electronic, cross-industry data exchange through the classification of standardized product descriptions. The central unique characteristic of ECLASS is the possibility of providing an unambiguous, language-neutral, machine-readable, and industry-independent description of products and services and their characteristics. Currently, there are about 45,000 product classes, that are organized in four levels. On the fourth level a set of Properties is assigned. The set of Properties are generated as a set of the 19,000 well-defined Properties of the ECLASS Dictionary. A large part of the goods and services traded worldwide is represented in ECLASS. In addition to classes and Properties, further element types and modelling features exist in the ECLASS Dictionary. The Dictionary, which is exchanged in different ways like CSV, XML or JSON is available in two representations ECLASS Basic and ECLASS Advanced.

ECLASS uses globally unique identifiers for every Structure Element included in the ECLASS Standard. This globally unique identifier is called IRDI (International Registration Data Identifier). These identifiers are used to ensure that the semantic of an element is unique in the overall system. The IRDI is based on the international standards ISO/IEC 11179-6, ISO 29002, and ISO 6532.

Technically, ECLASS consists of classes for classification as well as for product description. Globally available Properties are attached to the latter, to which in turn Value Lists can be attached. ECLASS is therefore already a graph of structural elements (nodes) and relationships (edges). To exchange such graphs, the Resource Description Framework (RDF) is a superior technology compared to XML because XML captures syntax by semantics only. RDF was developed by the World Wide Web Consortium (W3C) and is a concept to represent information in subject-predicate-object triples.

While in most XML or JSON serialized models relations are only mapped in a proprietary way, RDF defines the basis for mapping as well as interpreting such relations. Thus, not only information can be represented and exchanged, but also knowledge. The data representation of RDF captures both syntax (structure) and semantics (meaning).

More information about RDF can be found at https://www.w3.org/TR/rdf11-concepts/ and https://www.w3.org/TR/rdf-schema/.

However, ECLASS is only exchanged as CSV, XML or JSON in specific schemas. A directly machine-interpretable representation of the logical statements in the ECLASS Dictionary as e.g. RDF is still missing today.

W3C standards (and recommendations) have created a robust technology to create and link RDF data models that can be incorporated into a wide range of technology stacks. The Semantic Web contains a formal knowledge representation that enables interoperability by linked data. Therefore, RDF is a main building block of the semantic web and for ontologies in general.

Therefore, the ECLASS association aims to provide ECLASS with an RDF derivation of the ECLASS content in addition to the existing CSV, XML and JSON representations in order to be able to provide structured data on the web, e.g. in the W3C WoT Thing Description.

The objective of this document is the definition of an RDF representation of ECLASS. Technical rules are defined on how to represent the structural elements and modeling features of ECLASS together with the different relations in logical statements as subject-predicate-object triples. This serves both the technical implementation of an RDF derivation, e.g. from the ECLASS ContentDevelopmentPlatform (CDP), and the interpretation of the RDF representation.

1.1 Goals

- Technology point of view
  - Support semantic technologies
  - Support web technologies
  - Use well known W3C standard
  - Connectivity to other ontologies
  - Reduction of overhead caused by OntoML schema
  - Enable query technology
  - Allow cross-references by adding new relations
  - Validation of constraints by using SHACL
- Use/Business cases
  - Product Configurations / Composite device – assembly of articles
  - Knowledge Graph of Product Data
  - Check lists linked to product description

- PIM/PLM
  - Procurement – requirement vs. product selection
  - Check lists linked to product description.
  - Logistic aspects – assembly of parts
  - Documentation – combining description, terminology and translations
- Asset Administration Shell
- OPC-UA
  - Sematic Validation
- W3C WoT Thing Description
- Schema.org
  - Annotation on websites
- IoT-Schema (http://iotschema.org/)
- SAREF

1.2 Scope

The scope of this document is the mapping of all ECLASS contents from the ECLASS Dictionary into pure RDF. Contain the full information which is also available in XML or JSON.

1.3 Out of Scope

- Validation (See Part 2)
- Constraints of Value Lists (Optional for Part 2)
- Range (new in Release 14.0)
- Translation Quality Level (new in Release 14.0)
- Deprecation (Optional for Part 2)
  - Each element cannot be deleted in the Dictionary. It will be set to deprecated and included in future releases. Now deprecated information is not included in general.
  - Each reference can be deprecated:
    - A Property reference can be revoked on Class level
    - An Aspect can be revoked on Application Class level
    - A Value can be revoked on Value List level

1.4 Structure

The document is structured as follows. After a list of normative references in chapter 2 and a short state of the art in chapter 3, the generic concept is described in chapter 4. In chapter 5 the necessary data types are introduced. Afterwards chapter 6 describes the ECLASS RDF representation in detail based on simple dummy data. Chapter 7 sums this up an represent real examples of ECLASS structures in RDF.

Note: The dummy data contains IRDIs starting with 0173-X***.

2 Normative References

- RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation 25 February 2014
  https://www.w3.org/TR/rdf11-concepts/
- RDF Schema 1.1, W3C Recommendation 25 February 2014
  https://www.w3.org/TR/rdf11-schema/
- IEC 61360-2:2012, Standard data element types with associated classification scheme for electric components – Part 2: EXPRESS dictionary schema
- ISO TS 29002-5:2009, Industrial automation systems and integration — Exchange of characteristic data – Part 5: Identification scheme
- ECLASS Technical Specification 11 - Conceptual Data Model, Version 1.0
  https://eclass.eu/fileadmin/Redaktion/pdf-Dateien/Wiki/ECLASS_Technical-Specification_11_Conceptual-Data-Model_v_1.0.pdf
- ECLASS Download API JSON on SwaggerHub
  https://app.swaggerhub.com/apis/ECLASS_Standard/ECLASS_Download_JSON/1.3.8

3 State of the Art

Several approaches existed prior to this approach, but they were not official versions by ECLASS e.V.. In addition, neither all Structure Elements nor advanced features are supported in the approaches. The approaches are only briefly listed here:

- eClassOWL (2005)

eClassOWL was a project aimed at creating an ontology for the ECLASS Standard. The ontology is designed to provide a standardized way of representing ECLASS data using Semantic Web technologies, particularly OWL (Web Ontology Language). By doing so, it facilitates the integration and interoperability of ECLASS data with other systems and applications that utilize Semantic Web standards. The eClassOWL project offers resources and tools for utilizing and extending this ontology to support various use cases in industry and commerce.

http://www.heppnetz.de/projects/eclassowl/

- GenTax (2007)

The paper titled "GenTax: A Generic Methodology for Deriving OWL and RDF-S Ontologies from Hierarchical Classifications, Thesauri, and Inconsistent Taxonomies" presented at ESWC 2007 by Hepp and De Bruijn discusses a method for automatically generating an ontology representing the ECLASS Standard. It proposes an approach using Semantic Web technologies to map eClassOWL concepts. This allows the automatic creation, enhancing the interoperability and accessibility of ECLASS data within the Semantic Web framework.

http://www.heppnetz.de/files/hepp-de-bruijn-ESWC2007-gentax-CRC.pdf

- GoodRelations

GoodRelations is a standardized vocabulary for describing products, services, and business entities on the web. Developed using Semantic Web technologies, GoodRelations provides a framework for expressing detailed information about products and services in a machine-readable format. It allows businesses to describe their offerings with rich metadata, including details such as price, availability, delivery options, payment methods, and more. This structured data enables search engines and other applications to better understand and utilize the information, leading to improved visibility, interoperability, and user experience for e-commerce websites and online marketplaces.

http://wiki.goodrelations-vocabulary.org/

http://www.heppnetz.de/files/GoodRelationsEKAW2008-crc-final.pdf

- schema.org (2011)

Schema.org is a collaborative initiative aimed at creating a structured data vocabulary for describing entities on the web. It provides a standardized way to markup web content, enabling search engines to better understand and display information in search results. Schema.org offers a wide range of schemas covering various types of entities such as products, events, organizations, people, and more. By using Schema.org markup, webmasters can enhance the visibility and appearance of their content in search engine results pages, potentially leading to improved click-through rates and user engagement.

- PCS2OWL (2014)

The paper “PCS2OWL: A Generic Approach for Deriving Web Ontologies from Product Classification Systems” from Stolz, Rodriguez-Castro, Radinger and Hepp (2014) shows an extended approach of eClassOWL and GenTax which is compatible to GoodRelations.

http://www.heppnetz.de/files/pcs2owl-eswc2014.pdf

4 Concept

The basic concept of this specification is to split the set of rules into three parts:

1. ECLASS Content as pure RDF/OWL (this document)
    a. Definition of types as classes
    b. Definition of Properties
    c. Ruleset of ECLASS Mapping for each structural element
    d. IEC 61360 Data Types
2. Definition of constraints for validation with SHACL Shapes (separate document – Part 2)
3. A set of rules for the application of the ECLASS RDF Representation (separate document – Part 3), e.g.
    a. Basic instantiation
    b. Application in W3C WoT Thing Description
    c. Application in AAS

4.1 Concept of ECLASS Content as pure RDF/OWL (Part 1)

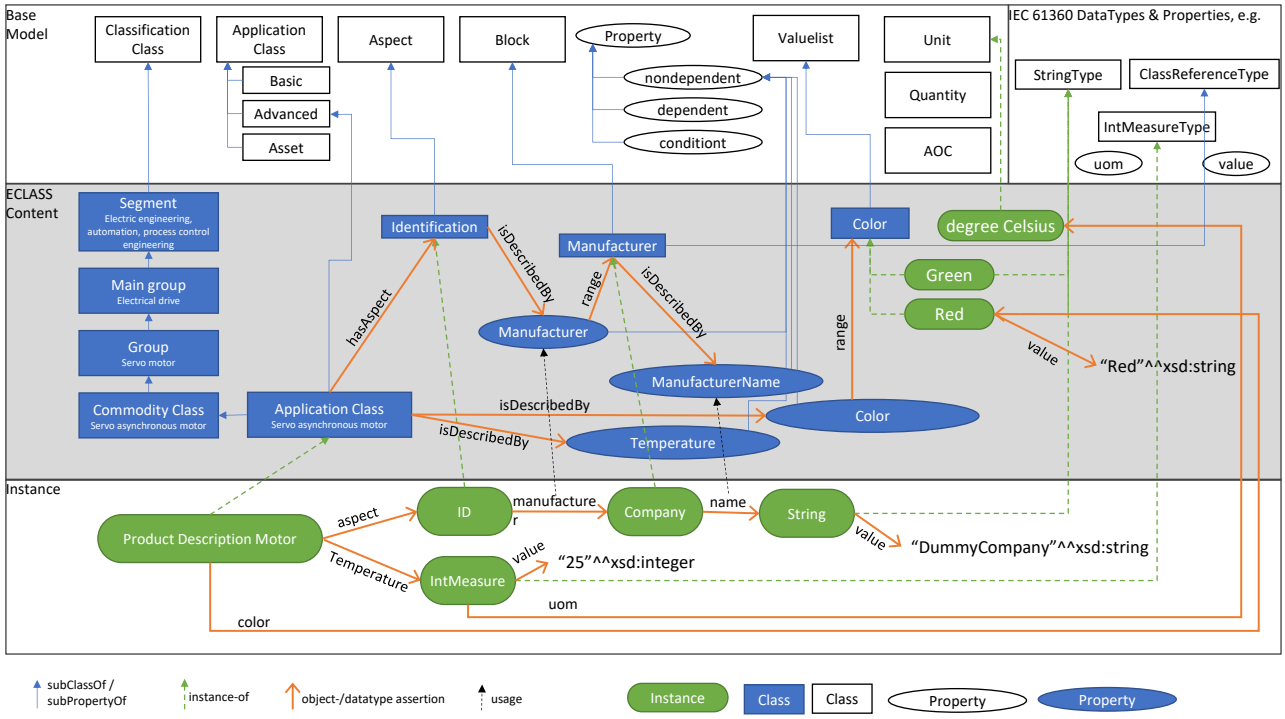

Figure 1: Generic concept of the ECLASS content in RDF

Source: Based on consulting from Derivo GmbH

4.2 Namespaces

For the transformation and the shown examples, the following namespaces are used and defined:

```
@prefix eclass:   <https://eclass-cdp.com/rdf/v1/eclass/> .
@prefix iec61360: <https://eclass-cdp.com/rdf/v1/iec61360/> .
@prefix owl:      <http://www.w3.org/2002/07/owl#> .
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:     <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#> .
```

The namespace “https://eclass-cdp.com/rdf/v1/iec61360/” defines the used Data Types. See the following chapter 5 for details.

The namespace “https://eclass-cdp.com/rdf/v1/eclass/” contains the ECLASS meta model, see chapter 6.

Note: Since the ECLASS Dictionary is a large model with many elements, ECLASS has decided to use “/” based URIs for flexibility and better usability on the client side. S.a. https://www.w3.org/wiki/HashVsSlash

4.3 Dictionaries & Releases

In addition to the generic namespace in the chapter above, it is necessary to define more namespaces. ECLASS defines one dictionary. However, it has several releases. Due to several reasons, it is not possible to merge all releases in one model in one namespace. The reasons are:

- Classification Classes have the same IRDI, but child classes can be extended.
- Application Classes are reused und classified according to several IRDIs from different releases.

To allow several ECLASS Releases in parallel, it is important to define specific release namespaces to allow one model per release. Therefore the namespace of the metamodel is extended with the release number according to the versioning of ECLASS defined in https://eclass.eu/support/content-creation/release-process/release-numbers-and-versioning. The "." in the versions are replaced by "-". In general, only the Major and Minor Release numbers are used.

The following table defines the namespaces for the currently existing releases:

Tabelle 1: ECLASS Release namespaces

ECLASS Release	Namespace	Prefix
5.1.4	https://eclass-cdp.com/rdf/v1/eclass5-1-4/	eclass5-1-4
6.0	https://eclass-cdp.com/rdf/v1/eclass6-0/	eclass6-0
6.0.1	https://eclass-cdp.com/rdf/v1/eclass6-0-1/	eclass6-0-1
6.2	https://eclass-cdp.com/rdf/v1/eclass6-2/	eclass6-2
7.0	https://eclass-cdp.com/rdf/v1/eclass7-0/	eclass7-0
7.1	https://eclass-cdp.com/rdf/v1/eclass7-1/	eclass7-1
8.0	https://eclass-cdp.com/rdf/v1/eclass8-0/	eclass8-0
8.1	https://eclass-cdp.com/rdf/v1/eclass8-1/	eclass8-1
9.0	https://eclass-cdp.com/rdf/v1/eclass9-0/	eclass9-0
9.1	https://eclass-cdp.com/rdf/v1/eclass9-1/	eclass9-1
10.0.1	https://eclass-cdp.com/rdf/v1/eclass10-0-1/	eclass10-0-1
10.1	https://eclass-cdp.com/rdf/v1/eclass10-0/	eclass10-1
11.0	https://eclass-cdp.com/rdf/v1/eclass11-0/	eclass11-0
11.1	https://eclass-cdp.com/rdf/v1/eclass11-1/	eclass11-1
12.0	https://eclass-cdp.com/rdf/v1/eclass12-0/	eclass12-0
13.0	https://eclass-cdp.com/rdf/v1/eclass13-0/	eclass13-0
14.0	https://eclass-cdp.com/rdf/v1/eclass14-0/	eclass14-0

Note: This table is a list for the current available Release of ECLASS. This list is therefore not complete. If you have any questions, please contact info@eclass.eu.

4.4 IRDI as URIs

If available, the IRDI of the ECLASS structural element is used as unique identifier as part of the RDF URI. Due the fact, that the IRDI contains the special character “#” which is not possible in RDF URIs.

Therefore, the following rules are defined:

The character “#” is replaced by “-”. This is according to https://eclass.eu/fileadmin/Redaktion/pdf-Dateien/Wiki/ECLASS-Technical-Specification-15-URI-Path-V1.0.pdf

Example:

For instance, the IRDI 0173-1#01-AAC063#019 is converted into 0173-1-01-AAC063-019.

The “Temperature transmitter” Classification Class with IRDI 0173-1#01-AAC063#022 in ECLASS 14.0 would have the following URI:

- https://eclass-cdp.com/rdf/v1/eclass14-0/0173-1-01-AAC063-022
- eclass14-0:0173-1-01-AAC063-022.

Note: In an Application Class with specific *BASIC*, *ADVANCED* or *ASSET* IRDIs the “_” are not changed. For instance, an IRDI would be transformed like the following:
0173-1---ADVANCED_1_1#01-AND579#012 becomes 0173-1---ADVANCED_1_1-01-ADN579-012.

4.5 IEC 61360

The conceptual model of ECLASS is based on ISO 61360. The goal was to transfer this conceptual model to RDF. Essential part are the data types defined in IEC 61360. Since no reusable model is available, a model for the Data Types is defined. Details can be found in the following chapter 5.

5 IEC 61360 Data Types

ECLASS contains and uses the data types defined in IEC 61360. All data types from ECLASS can be found here: https://eclass.eu/support/technical-specification/data-model/conceptual-data-model#c3035 . The following figure shows the data type system defined in IEC 61360-2:2012 (page 50):

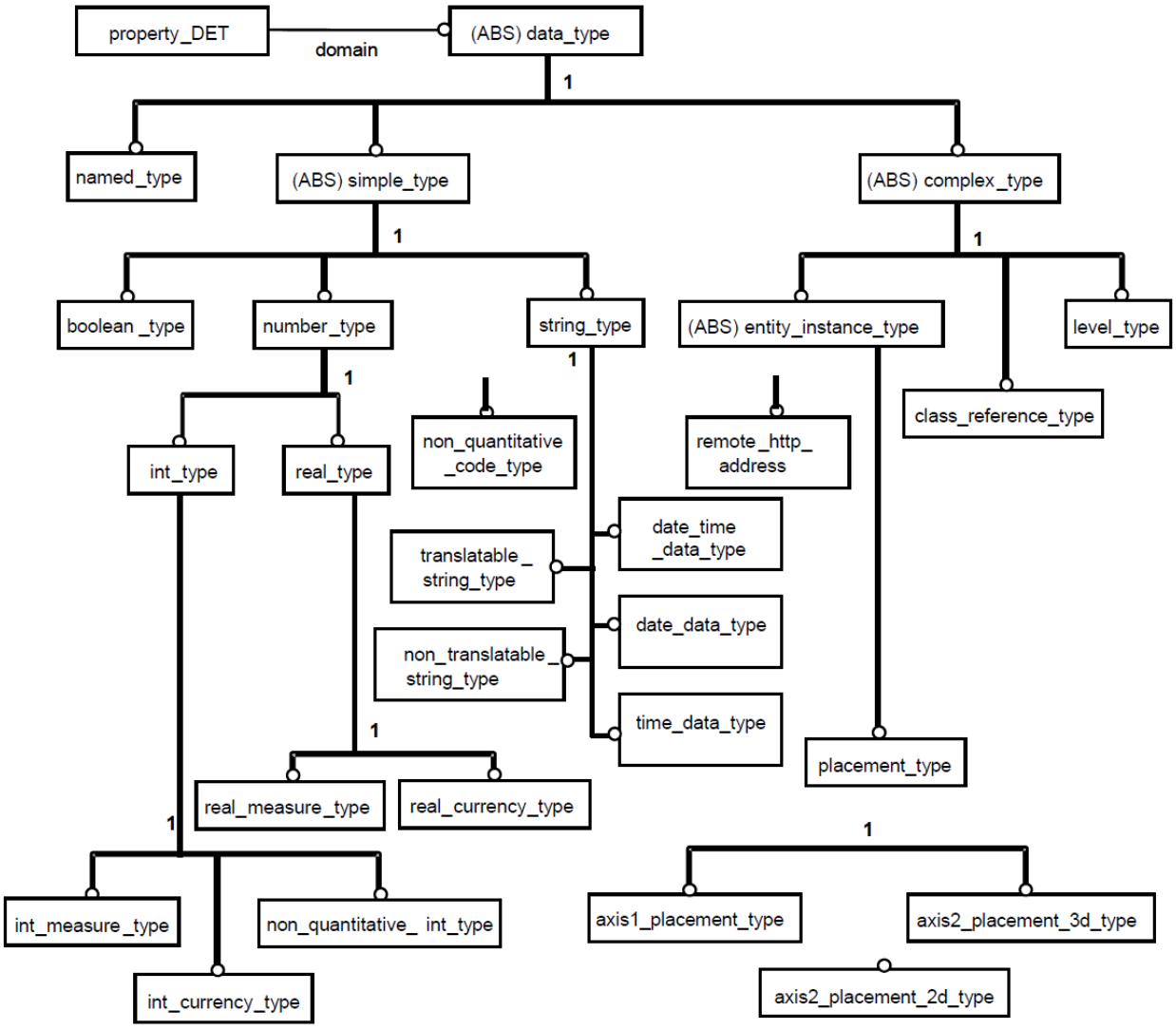

Figure 2: Data types according to IEC 61360-2

This type system has to be transferred to RDF. Therefore, first the namespace is defined:

```
iec61360: <https://eclass-cdp.com/rdf/v1/iec61360/>
```

In addition, a root element is defined:

```
iec61360:DataType
  rdf:type owl:Class ;
  rdfs:label "DataType" ;
.
```

Based on iec61360:DataType, corresponding classes for the data types are defined via subClassOf relations, resulting in the following hierarchy:

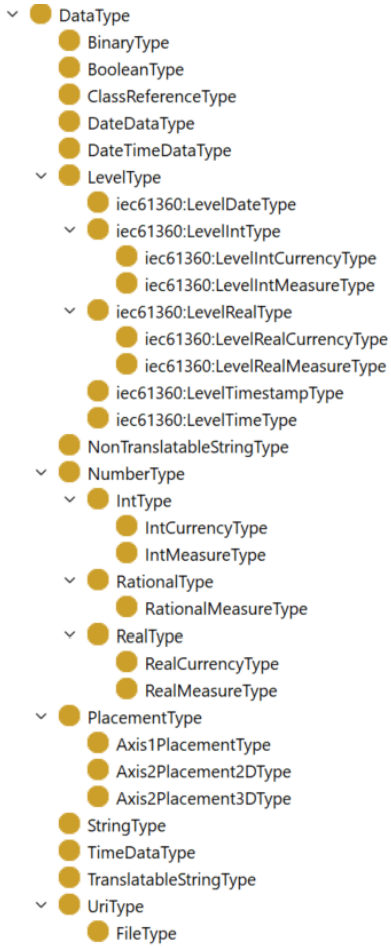

Figure 3: IEC 61360 data types in RDF

Note: Not all data types are mapped. Only the Data Types used by ECLASS are transformed.

Note: As an addition to the current IEC 61360-2 Data Types and with regard to the AAS, the data types File as well as Blob are mapped as type. Binary is defined in IEC 61360-1 and could be used for the representation of the type Blob. However, Blob is defined explicitly.

Note: In contrast to the structure of the data types in IEC 61360-2 shown above, the structure below String is not mapped. This is due to the types used by ECLASS. According to

https://eclass.eu/support/technical-specification/structure-and-elements/property#c2041, ECLASS only uses String instead of NonTranslatableString. Due to this fact the structure cannot be mapped because of the complications caused by the SHACL shapes (see Part 2). The xsd:string mapping would then be inherited, and it would contradict the rdf:langString with StringTranslatable.

Furthermore, several OWL Data Type Property elements are defined to transport the values.

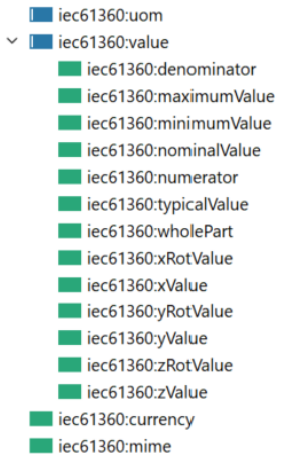

Figure 4: IEC 61360 Properties

The relation of these properties with the defined Data Type classes is defined in SHACL Shapes in Part 2. In addition, the corresponding XSD type mapping for the specific Value OWL Property in context of the Data Type class is important. Here, ECLASS has already defined an IEC 61360 Data Type to XSD mapping (see https://eclass.eu/support/technical-specification/data-model/datatype-to-xsd-mapping). However, it does not fit for all types. Especially Level and AxisType are complex and need a set of values. Furthermore, for the *Measure* Types a combination of unit and value is necessary.

Note: The XSD mapping is not part of the RDF model, instead it is included in SHACL Shapes. See Part 2 of this document series. It is only noted which XSD type should be used.

The following paragraphs show the corresponding representations of the OWL Data Type Property;

```
iec61360:value
  rdf:type owl:DatatypeProperty ;
  rdfs:label "value" ;
.
```

For a combination with a unit:

```
iec61360:uom
  rdf:type owl:ObjectProperty ;
  rdfs:label "unitOfMeasure" ;
.
```

For a combination with a currency:

```
iec61360:currency
  rdf:type owl:ObjectProperty ;
  rdfs:label "currency" ;
.
```

Note: rdfs:range is not defined for iec61360:uom and iec61360:currency to avoid a mixing of the ECLASS RDF representation with the IEC 61360 type system. This is a possible extension in Part 2.

See 5.9 for iec61360:wholePart, iec61360:numerator and iec61360:denominator.

See 5.14 for iec61360:minimalValue, iec61360:maximumValue, iec61360:nominalValue, iec61360:typicalValue.

See 5.15 for iec61360:xRotValue, iec61360:xValue, iec61360:yRotValue, iec61360:yValue, iec61360:zRotValue, iec61360:zValue.

In addition to the shown approach the generic concept is close to ISO 15926 Part 14 / IDO. It is skipped here to make it as simple as possible and to remove unnecessary dependencies. It would be possible to integrate e.g. https://rds.posccaesar.org/ontology/lis14/ont/core/ in the future.

5.1 String

```
iec61360:StringType
        rdf:type         owl:Class ;
        rdfs:label       "StringType" ;
        rdfs:subClassOf  iec61360:DataType .
```

Note: iec61360:value should be evaluated with xsd:string.

Note: See note in 5.

5.2 Non-Translatable String

```
iec61360:NonTranslatableStringType
        rdf:type          owl:Class ;
        rdfs:label        "NonTranslatableStringType" ;
        rdfs:subClassOf   iec61360:DataType .
```

Note: iec61360:value should be evaluated with xsd:string.

Note: This type is not used by ECLASS yet.

5.3 String Translatable

```
iec61360:TranslatableStringType
        rdf:type          owl:Class ;
        rdfs:label        "TranslatableStringType" ;
        rdfs:subClassOf   iec61360:DataType .
```

Note: iec61360:value should be evaluated with rdf:langString.

5.4 Boolean

```
iec61360:BooleanType
        rdf:type          owl:Class ;
        rdfs:label        "BooleanType" ;
        rdfs:subClassOf   iec61360:DataType .
```

Note: iec61360:value should be evaluated with xsd:boolean.

Note: The data type Boolean also uses a Value List in ECLASS. See 6.6.9.6 for details.

5.5 Uri

```
iec61360:UriType
        rdf:type          owl:Class ;
        rdfs:label        "UriType" ;
        rdfs:subClassOf   iec61360:DataType .
```

Note: iec61360:value should be evaluated with xsd:anyURI.

5.6 Number

```
iec61360:NumberType
        rdf:type          owl:Class ;
        rdfs:label        "NumberType" ;
        rdfs:subClassOf   iec61360:DataType .
```

Note: iec61360:NumberType should be seen as an abstract class and should not be used in individuum for any asset description.

5.7 Int

```
iec61360:IntType
        rdf:type         owl:Class ;
        rdfs:label       "IntType" ;
        rdfs:subClassOf  iec61360:NumberType .
```

Note: iec61360:value should be evaluated with xsd:integer.

5.7.1 Int Measure

Defines a set of value and unit. Therefore, iec61360:uom has to be set.

```
iec61360:IntMeasureType
        rdf:type         owl:Class ;
        rdfs:label       "IntMeasureType" ;
        rdfs:subClassOf  iec61360:IntType .
```

5.7.2 Int Currency

Defines a set of value and currency. Therefore, iec61360:currency has to be set.

```
iec61360:IntCurrencyType
        rdf:type         owl:Class ;
        rdfs:label       "IntCurrencyType" ;
        rdfs:subClassOf  iec61360:IntType .
```

5.8 Real

```
iec61360:RealType
        rdf:type         owl:Class ;
        rdfs:label       "RealType" ;
        rdfs:subClassOf  iec61360:NumberType .
```

Note: iec61360:value should be evaluated with xsd:float.

5.8.1 Real Measure

Defines a set of value and unit. Therefore, iec61360:uom has to be set.

```
iec61360:RealMeasureType
        rdf:type         owl:Class ;
        rdfs:label       "RealMeasureType" ;
        rdfs:subClassOf  iec61360:RealType .
```

5.8.2 Real Currency

Defines a set of value and currency. Therefore, iec61360:currency has to be set.

```
iec61360:RealCurrencyType
        rdf:type         owl:Class ;
        rdfs:label       "RealCurrencyType" ;
        rdfs:subClassOf  iec61360:RealType .
```

5.9 Rational

According to ISO 13584-32 (see Version 2012, page 58) the Rational data type defines a set of three values. Here, iec61360:fullInt, iec61360:numerator and iec61360:denominator should be evaluated with xsd:integer.

```
iec61360:RationalType
        rdf:type         owl:Class ;
        rdfs:label       "RationalType" ;
        rdfs:subClassOf  iec61360:NumberType .

iec61360:wholePart
        rdf:type  owl:DatatypeProperty ;
        rdfs:label          "wholePart" ;
        rdfs:subPropertyOf  iec61360:value .

iec61360:numerator
        rdf:type  owl:DatatypeProperty ;
        rdfs:label          "numerator" ;
        rdfs:subPropertyOf  iec61360:value .

iec61360:denominator
        rdf:type  owl:DatatypeProperty ;
        rdfs:label          "denominator" ;
        rdfs:subPropertyOf  iec61360:value .
```

Note: iec61360:value should not be used.

5.9.1 Rational Measure

Defines a set of value and unit. Therefore, iec61360:uom has to be set.

```
iec61360:RationalMeasureType
        rdf:type         owl:Class ;
        rdfs:label       "RationalMeasureType" ;
        rdfs:subClassOf  iec61360:RationalType .
```

5.10 Date

```
iec61360:DateType
        rdf:type          owl:Class ;
        rdfs:label        "DateType" ;
        rdfs:subClassOf   iec61360:DataType .
```

Note: iec61360:value should be evaluated with xsd:date.

5.11 Time

```
iec61360:TimeType
        rdf:type          owl:Class ;
        rdfs:label        "TimeType" ;
        rdfs:subClassOf   iec61360:DataType .
```

Note: iec61360:value should be evaluated with xsd:time.

5.12 DateTime

```
iec61360:DateTimeType
        rdf:type          owl:Class ;
        rdfs:label        "DateTimeType" ;
        rdfs:subClassOf   iec61360:DataType .
```

Note: iec61360:value should be evaluated with xsd:dateTime.

5.13 Class Reference

Reference is a specific Data Type in IEC 61360. It is used to reference a Block.

```
iec61360:ClassReferenceType
        rdf:type          owl:Class ;
        rdfs:label        "ClassReferenceType" ;
        rdfs:subClassOf   iec61360:DataType .
```

5.14 Level Type

For Date, Integer (x), Real (x), Timestamp and Time a Level Type is possible. See https://eclass.eu/support/technical-specification/structure-and-elements/level-type for further information.

The root type is as follows:

```
iec61360:LevelType
        rdf:type          owl:Class ;
        rdfs:label        "LevelType" ;
        rdfs:subClassOf   iec61360:DataType .
```

Note: iec61360:LevelType should be seen as an abstract class and should not be used in individuum for any asset description.

A Level Type is a set of four optional values. According to IEC 61360-2 (version 2012, page 65) iec61360:minimalValue, iec61360:maximumValue, iec61360:nominalValue and iec61360:typicalValue should be used.

```
iec61360:minimumValue
        rdf:type            owl:DatatypeProperty ;
        rdfs:label          "minimumValue" ;
        rdfs:subPropertyOf  iec61360:value .

iec61360:maximumValue
        rdf:type            owl:DatatypeProperty ;
        rdfs:label          "maximumValue" ;
        rdfs:subPropertyOf  iec61360:value .

iec61360:nominalValue
        rdf:type            owl:DatatypeProperty ;
        rdfs:label          "nominalValue" ;
        rdfs:subPropertyOf  iec61360:value .

iec61360:typicalValue
        rdf:type            owl:DatatypeProperty ;
        rdfs:label          "typicalValue" ;
        rdfs:subPropertyOf  iec61360:value .
```

5.14.1 Level Type Int

```
iec61360:LevelIntType
        rdf:type         owl:Class ;
        rdfs:subClassOf  iec61360:LevelType .
```

Here iec61360:minimalValue, iec61360:maximumValue, iec61360:nominalValue and iec61360:typicalValue should be evaluated with xsd:integer.

5.14.1.1 Level Type Int Measure

Defines a set of value and unit. Therefore, iec61360:uom has to be set.

```
iec61360:LevelIntMeasureType
        rdf:type         owl:Class ;
        rdfs:subClassOf  iec61360:LevelIntType .
```

5.14.1.2 Level Type Int Currency

Defines a set of value and currency. Therefore, iec61360:currency has to be set.

```
iec61360:LevelIntCurrencyType
        rdf:type         owl:Class ;
        rdfs:subClassOf  iec61360:LevelIntType .
```

5.14.2 Level Type Real

```
iec61360:LevelRealType
        rdf:type         owl:Class ;
        rdfs:subClassOf  iec61360:LevelType .
```

Here iec61360:minimalValue, iec61360:maximumValue, iec61360:nominalValue and iec61360:typicalValue should be evaluated with xsd:float.

5.14.2.1 Level Type Real Measure

Defines a set of value and unit. Therefore, iec61360:uom has to be set.

```
iec61360:LevelRealMeasureType
        rdf:type         owl:Class ;
        rdfs:subClassOf  iec61360:LevelRealType .
```

5.14.2.2 Level Type Real Currency

Defines a set of value and currency. Therefore, iec61360:currency has to be set.

```
iec61360:LevelRealCurrencyType
        rdf:type         owl:Class ;
        rdfs:subClassOf  iec61360:LevelRealType .
```

5.14.3 Level Type Timestamp

```
iec61360:LevelTimestampType
        rdf:type         owl:Class ;
        rdfs:subClassOf  iec61360:LevelType .
```

Here iec61360:minimalValue, iec61360:maximumValue, iec61360:nominalValue and iec61360:typicalValue should be evaluated with xsd:dateTime.

5.14.4 Level Type Date

```
iec61360:LevelDateType
        rdf:type         owl:Class ;
        rdfs:subClassOf  iec61360:LevelType .
```

Here iec61360:minimalValue, iec61360:maximumValue, iec61360:nominalValue and iec61360:typicalValue should be evaluated with xsd:date.

5.14.5 Level Type Time

```
iec61360:LevelTimeType
        rdf:type         owl:Class ;
        rdfs:subClassOf  iec61360:LevelType .
```

Here iec61360:minimalValue, iec61360:maximumValue, iec61360:nominalValue and iec61360:typicalValue should be evaluated with xsd:time.

5.15 Axis Type

The Axis Type describes a placement in three-dimensional space in terms of a point and an axis direction. There are three different Axis Types available. Axis1PlacementType, Axis2Placement2DType and Axis2Placement3DType. Now, ECLASS only uses Axis1PlacementType. For detailed information see https://eclass.eu/support/technical-specification/structure-and-elements/axis-1d.

Note: For this data type the Unit is optional. In ECLASS the length Unit is always “mm” and the rotation is given in “°”. If a Unit is set in BMEcat, only the length Unit and not the angle Unit is exported at the level of the Property itself. The angle Unit is expected to always be the default Unit.

The root element is:

```
iec61360:PlacementType
        rdf:type         owl:Class ;
        rdfs:label       "PlacementType" ;
        rdfs:subClassOf  iec61360:DataType .
```

Note: iec61360:PlacementType should be seen as an abstract class and should not be used in individuum for any asset description.

For the valuation the following OWL Data Type Property elements are defined and should be used:

```
iec61360:xValue  rdf:type   owl:DatatypeProperty ;
        rdfs:label          "xValue" ;
        rdfs:subPropertyOf  iec61360:value .

iec61360:yValue  rdf:type   owl:DatatypeProperty ;
        rdfs:label          "yValue" ;
        rdfs:subPropertyOf  iec61360:value .

iec61360:zValue  rdf:type   owl:DatatypeProperty ;
        rdfs:label          "zValue" ;
        rdfs:subPropertyOf  iec61360:value .

iec61360:xRotValue  rdf:type  owl:DatatypeProperty ;
        rdfs:label          "xRotValue" ;
        rdfs:subPropertyOf  iec61360:value .

iec61360:yRotValue  rdf:type  owl:DatatypeProperty ;
        rdfs:label          "yRotValue" ;
        rdfs:subPropertyOf  iec61360:value .

iec61360:zRotValue  rdf:type  owl:DatatypeProperty ;
        rdfs:label          "zRotValue" ;
        rdfs:subPropertyOf  iec61360:value .
```

The evaluation of iec61360:value is always xsd:float.

5.15.1 Axis 1 Placement

```
iec61360:Axis1PlacementType
        rdf:type         owl:Class ;
        rdfs:label       "Axis1PlacementType" ;
        rdfs:subClassOf  iec61360:PlacementType .
```

5.15.2 Axis 2 Placement 2D

```
iec61360:Axis2Placement2DType
        rdf:type         owl:Class ;
        rdfs:label       "Axis2Placement2DType" ;
        rdfs:subClassOf  iec61360:PlacementType .
```

5.15.3 Axis 2 Placement 3D

```
iec61360:Axis2Placement3DType
        rdf:type         owl:Class ;
        rdfs:label       "Axis2Placement3DType" ;
        rdfs:subClassOf  iec61360:PlacementType .
```

5.16 Blob

```
iec61360:BlobType
        rdf:type         owl:Class ;
        rdfs:label       "BlobType" ;
        rdfs:subClassOf  iec61360:DataType .
```

iec61360:value should be evaluated with xsd:base64binary .

Note: BLOB is available from ECLASS Release 13.0.

Note: This type is similar to the Binary-Type from IEC 61360-1. However, ECLASS is according to IEC 61360-2. Therefore, this Blob Type is defined.

5.17 File

The data type File is not specified in IEC yet, but according to several groups it is a proposal in IEC and requirement for the Asset Administration Shell, which again is an IEC 63278-1 proposal. Therefore, it is managed under the same namespace here and defines an extension of iec61360:UriType.

```
iec61360:FileType
  rdf:type owl:Class ;
  rdfs:label "FileType" ;
  rdfs:subClassOf iec61360:UriType ;
.
```

Note: File is available from ECLASS Release 13.0.

Furthermore, iec61360:FileType needs a mime type:

```
iec61360:mime
  rdf:type owl:DatatypeProperty ;
  rdfs:label "mimeType" ;
.
```

6 ECLASS RDF Model

The following types of structural elements contained in ECLASS are also included in the provided RDF model. The goal is to support the full conceptual model of ECLASS defined in https://eclass.eu/support/technical-specification/data-model/conceptual-data-model.

Table 1: In ECLASS RDF included elements

Structural element	Definition
Classification Class	A product group identified by a preferred name and an eight-digit coded name that represents the four-level classification structure
Application Class	Class that comprises all characteristics described by Properties
Keyword	An alternative name of a class
Property	A characteristic of a class
Synonym	An alternative name of a Property
Unit incl. Quantities and Aspect of Conversions	Standardized unit of measure
Value List	A restrictive list of valid specifications of a Property
Value	A specification of a characteristic
Aspect	Sub-class of an Application Class that comprises all Properties describing a certain Aspect of a product, not the product itself, e.g. packing information
Block	Sub-class of an Application Class that comprises all Properties describing a certain part of a product

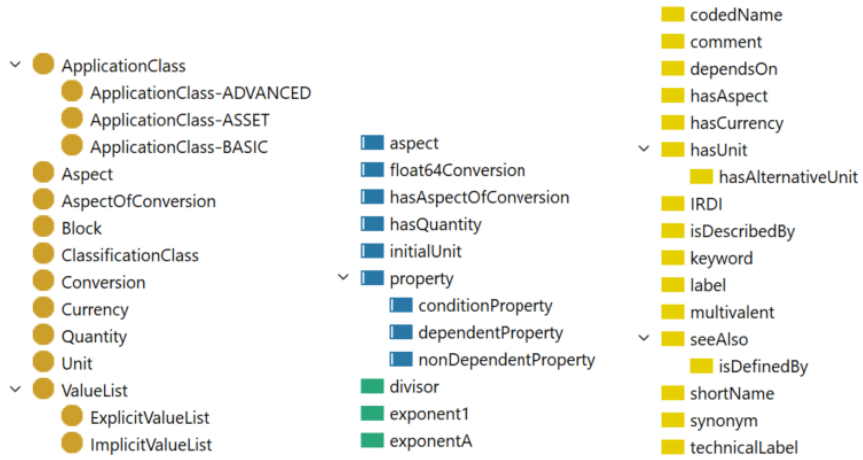

Figure 5: Excerpt of defined Classes and Properties

6.1 Common Attributes

The following identifying and descriptive attributes are applied to most of the structural elements.

Note: Currencies, Keywords and Synonyms have no definition.

Note: The modeling of restrictions like translatable string, RegEx or mandatory links or cardinalities is not part of the RDF model, instead it is included in SHACL Shapes. See Part 2 of this document series.

6.1.1 IRDI

The IRDI is a String (example: 0173-01#ZZ-AAA123#001) with the following RegEx:

^[0-9]{4}-[A-Z0-9:_.]{1,35}((-[A-Z0-9:_.]{1,35}(-[A-Z0-9]{1}(-[A-Z0-9:_.]{1,70})?)?)?|-([A-Z0-9:_.]{1,35})?--[A-Z0-9:_.]{1,70}|---[A-Z0-9:_.]{1,70})#[0-9A-Z]{2}-[A-Z0-9:_.]{1,131}#[0-9]{1,10}$

The IRDI is modelled as OWL Annotation Property:

```
eclass:IRDI  rdf:type  owl:AnnotationProperty ;
        rdfs:label  "IRDI" ;
        rdfs:range  xsd:string .
```

6.1.2 Preferred Name

The preferred name is a Translatable String and is transferred via rdfs:label.

6.1.3 Definition

The definition is a Translatable String and is transferred via rdfs:comment.

6.2 Classification Class

For ECLASS as a classification system, the most fundamental structural element is the Classification Class. With the help of Classification Classes products are divided into certain categories of similar products, the product groups. ECLASS is a mono hierarchical classification system, i.e. every product group is to be found only once in the hierarchical tree structure.

A Classification Class has all common attributes, has a root class with IRDI 0173-X#01-AAA002#001 and four classification levels:

```
eclass:0173-X-01-AAA002-001
        rdf:type         owl:Class ;
        rdfs:label       "ClassificationClass" ;
        eclass:IRDI      "0173-X#01-AAA002#001" .
```

The Classification Classes have child and parent relations. The parent relation is modeled via subClassOf. The child relations are not modelled, because their interpretation is possible via inference.

On the fourth level an Application Class is linked to the Classification Class. This is not included directly. The interpretation is possible via inference. See 6.3.

6.2.1 Coded Name

Additionally, to the common attributes Classification Classes have a coded name for the class code as String following a RegEx $^[0-9]{8}$

```
eclass:codedName  rdf:type  owl:AnnotationProperty ;
        rdfs:label  "codedName" ;
        rdfs:range  xsd:string .
```

6.2.2 Technical Label

Due to default elements sorting based on the URI or rdf:label it is not easy to navigate in tools like RDF Editors. Therefore, a technical label is introduced, which can be configured for visualization in editors. This OWL Annotation Property is optional. It is for representations only.

```
eclass:technicalLabel
        rdf:type    owl:AnnotationProperty ;
        rdfs:label  "technicalLabel" ;
        rdfs:range  xsd:string .
```

This Property supports a CDP like navigation like shown in the following figure:

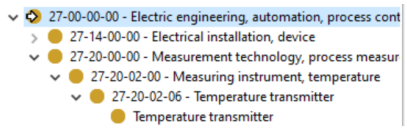

Figure 6: Example for eclass:technicalLabel

6.2.3 Keywords

Furthermore, Classification Classes can have alternative names as Keywords. In the Dictionary a Keyword is a set triple of an IRDI, a Preferred Name incl. a language code as String according to $^([a-z]{2})_([A-Z]{2})$ is defined.

Since the IRDI is not used, Keywords as a rdf:langString are linked via the defined OWL Annotation Property:

```
eclass:keyword  rdf:type  owl:AnnotationProperty ;
        rdfs:label  "keyword" ;
        rdfs:range  rdf:langString ;
.
```

Note: The language code is mapped to @lang annotation in rdf:langString

6.2.4 Example of Classification Class

In the following example a segment class is shown:

```
eclass:0173-X-01-XXX001-001
  rdf:type owl:Class ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "Test1"@en ;
  rdfs:subClassOf eclass:0173-X-01-AAA002-001 ;
  eclass:IRDI "0173-X#01-XXX001#001" ;
  eclass:codedName "01000000" ;
  eclass:keyword "TestKeyword"@en ;
  eclass:technicalLabel "01-00-00-00 - Test1" ;
.
```

6.3 Application Class

An Application Class is a representation of a type or family of product (device, equipment, software or service) which can be described by a generic set of Properties. An Application Classes forms a master for the product description. This type has all common attributes and has one root to group all Application Classes with IRDI 0173-X---BASIC_1_1#01-AAA001#001.

The root Application Class in RDF is as follows:

```
eclass:0173---BASIC_1_1-01-AAA001-001
        rdf:type     owl:Class ;
        rdfs:label   "ApplicationClass" ;
        eclass:IRDI  "0173---BASIC_1_1#01-AAA001#001" .
```

Further Application Classes are modelled via rdfs:subClassOf relation.

In ECLASS there are three types of Application Classes: Advanced, Basic and Asset (new in ECLASS Release 14.0). In general, the Application Classes can be differenced by the IRDI, e.g. the Application Classes classified by 0173-1#01-ADI126#013 – 27-05-04-03 – Rechargeable battery:

- 0173-1---**ADVANCED**_1_1#01-AEX756#011
- 0173-1---**ASSET**_1_1#01-AHM518#002
- 0173-1---**BASIC**_1_1#01-ADI127#013

However, to make the model as machine readable as possible the following three classes are defined:

```
eclass:Advanced
  rdf:type         owl:Class ;
  rdfs:label       "ApplicationClass-ADVANCED" ;
  rdfs:subClassOf  eclass:0173---BASIC_1_1-01-AAA001-001 .

eclass:Asset
  rdf:type         owl:Class ;
  rdfs:label       "ApplicationClass-ASSET" ;
  rdfs:subClassOf  eclass:0173---BASIC_1_1-01-AAA001-001 .

eclass:Basic
  rdf:type         owl:Class ;
  rdfs:label       "ApplicationClass-BASIC" ;
  rdfs:subClassOf  eclass:0173---BASIC_1_1-01-AAA001-001 .
```

Furthermore, an Application Class is classified as a specific Classification Class. A link between an Application Class and a minimum of one Classification Class is necessary. Therefore, in addition to a rdfs:subClassOf relation to one of the three Application Class types, a further rdfs:subClassOf relation to one or more Classification Class is defined.

6.3.1 Description with Properties

Additionally, an Application Class is described by Properties. To link possible Properties a new OWL Annotation Property is introduced:

```
eclass:isDescribedBy
  rdf:type owl:AnnotationProperty ;
  rdfs:label "isDescribedBy" ;
.
```

Note: This is also applied for Aspects and Blocks.

Note: The cardinality (if a Property is mandatory or optional) is not part of the RDF model, instead it is included in SHACL Shapes. See Part 2 of this document series. It is only noted by which Properties an Application Class, an Aspect or a Block can be described.

6.3.2 Description with Aspects

Moreover, an Application Class can be described by Aspects. For this purpose, a further OWL Annotation Property is defined:

```
eclass:hasAspect
  rdf:type owl:AnnotationProperty ;
  rdfs:label "hasAspect" ;
  rdfs:range eclass:0173-1-01-PAA001-001
.
```

6.3.3 Example Application Class

In the following example an Application Class is shown:

```
eclass:0173-X-ADVANCED-01-XXX999-001
  rdf:type owl:Class ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "TestACAdvancedAllIn"@en ;
  rdfs:subClassOf eclass:0173-X-01-XXX004-001 ;
  rdfs:subClassOf eclass:Advanced ;
  eclass:IRDI "0173-X-ADVANCED#01-XXX999#001" ;
  eclass:hasAspect eclass:0173-X-01-XXX010-001 ;
  eclass:isDescribedBy eclass:0173-X-02-XXX001-001 ;
.
```

6.3.4 Application Class Individuum

To describe a product an individuum can be created with type of the Application Class. For instance, the following individuum shows an instance of the defined Application Class in 6.3.3.

```
example:UUID_6c15a391-3622-4eef-8eb5-c4080d37fbb4
  rdf:type eclass:0173-X-ADVANCED_01-XXX999-001 ;
  eclass:0173-X-02-XXX001-001 example:UUID_56d0e29e-54ea-4ccc-8aa5-daf39cabd803 ;
  eclass:aspect example:UUID_d91f97d6-a03d-4a03-a2d4-397a0d0b5eec ;
.
```

To link the characteristics as done in line 3 of the example above the ECLASS Properties should be used. See 6.6 for details.

To link aspects a specific OWL Object Property eclass:aspect is defined:

```
eclass:aspect
  rdf:type     owl:ObjectProperty ;
  rdfs:label   "aspect" ;
  rdfs:range   eclass:0173-X-01-PAA001-001 .
```

6.4 Block

A Block is a subset of Properties within an Application Class describing an abstraction of a feature of the product or of a part of a composite device. To sum it up a Block bundles Properties. The description with Properties is similar to Application Classes. This type has all common attributes, and the root class is 0173-X#01-AZI000#002.

The root Block in RDF is as follows:

```
eclass:0173-X-01-AZI000-002
        rdf:type      owl:Class ;
        rdfs:label    "Block" ;
        eclass:IRDI   "0173-X#01-AZI000#002" .
```

Blocks can also have Sub-Blocks. This hierarchy defines an inheritance of Properties, too.

Note: In contrast to the serialization as XML or JSON, the inherited Properties are not repeated.

Due to the fact, that Properties are defined as OWL ObjectProperties (see Property6.6) and Blocks are linked to Application Classes, Aspects or Blocks itself via Properties with the Data Type Class Reference (see 5.13) each Block is defined as rdfs:subClassOf iec61360:ClassReferenceType as well.

6.4.1 Example Block

In the following example a Block is shown:

```
eclass:0173-X-01-XXX009-001
  rdf:type owl:Class ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "TestBlock"@en ;
  rdfs:subClassOf eclass:0173-X-01-AZI000-002 ;
  rdfs:subClassOf iec61360:ClassReferenceType ;
  eclass:IRDI "0173-X#01-XXX009#001" ;
  eclass:isDescribedBy eclass:0173-X-02-XXX001-001 ;
.
```

6.4.2 Example Block Individuum

```
example:UUID_954b2768-09a5-4ed7-8e8a-ef6a0de38410
  rdf:type eclass:0173-X-01-XXX009-001 ;
  eclass:0173-X-02-XXX001-001 example:UUID_1bf47b23-c3ba-4482-8c8c-10d86bebdbbb ;
.
```

6.5 Aspect

Aspects bundles Properties to describe a specific view of an asset. This type has all common attributes and is described by Properties like Application Classes or Blocks. The root class is 0173-X#01-PAA001#001.

The root Aspect in RDF is as follows:

```
eclass:0173-X-01-PAA001-001
        rdf:type     owl:Class ;
        rdfs:label   “Aspect” ;
        eclass:IRDI  “0173-X#01-PAA001#001” .
```

6.5.1 Example Aspect

In the following example an Aspect is shown:

```
eclass:0173-X-01-XXX010-001
  rdf:type owl:Class ;
  rdfs:comment “Test Definition EN”@en ;
  rdfs:label “TestAspect”@en ;
  rdfs:subClassOf eclass:0173-X-01-PAA001-001 ;
  eclass:IRDI “0173-X#01-XXX010#001” ;
  eclass:isDescribedBy eclass:0173-X-02-XXX001-001 ;
.
```

6.5.2 Example Aspect Individuum

```
example:UUID_fd3f193b-299f-4b56-827b-1bcc5a21390d
  rdf:type eclass:0173-X-01-XXX010-001 ;
  eclass:0173-X-02-XXX001-001 example:UUID_c8303724-9023-4a42-8556-6637ae238920 ;
.
```

6.6 Property

In addition to pure grouping by classification, product description by Properties is an important aspect of ECLASS. A Property describes a characteristic of a product. These Properties are mapped in RDF as OWL Object Properties. These Properties have all common attributes.

The root Property is defined as follows:

```
eclass:property  rdf:type  owl:ObjectProperty .
```

6.6.1 Property type

There are three different Property types: non-dependent, dependent, condition:

- Non-Depending Property: A Property of which the (instantiated) Value is not depending on any other Value.
- Dependent Property: There are cases, where a Property’s Value make only sense if another Property’s Value is mandatory existing. In this case the Property depends on a non-empty set of Properties, its Conditions.
- Condition Property: A Property which acts as condition for a Dependent Property.

These are defined as subPropertyOf eclass:property as follows:

```
eclass:nonDependent
  rdf:type  owl:ObjectProperty ;
  rdfs:label "nonDependentProperty" ;
  rdfs:subPropertyOf  eclass:property .
```

```
eclass:dependent
  rdf:type  owl:ObjectProperty ;
  rdfs:label "dependentProperty" ;
  rdfs:subPropertyOf  eclass:property .
```

```
eclass:condition
  rdf:type  owl:ObjectProperty ;
  rdfs:label "conditionProperty" ;
  rdfs:subPropertyOf  eclass:property .
```

6.6.2 Multivalent

A Property in ECLASS defines if a characteristic can carry one or many values. This information is transferred via an OWL Annotation Property:

```
eclass:multivalent
  rdf:type  owl:AnnotationProperty ;
  rdfs:label  “multivalent” ;
  rdfs:range  xsd:boolean .
```

6.6.3 Synonyms

Furthermore, a Property can have additional Synonyms. Such Synonyms are modeled like Keywords. See 6.2.3. To link a Synonym to a Property the following OWL Annotation Property is defined:

```
eclass:synonym
  rdf:type  owl:AnnotationProperty ;
  rdfs:label  “synonym” ;
  rdfs:range  rdf:langString .
```

6.6.4 Data Type

A very important information of a Property is a specific Data Type. The Data Type is expressed via rdfs:range. This points to the defined IEC 61360 Data Types from chapter 5.

6.6.5 Unit

For specific Data Types a Property also has a defined Unit. To link a Unit the following OWL Annotation Property is defined:

```
eclass:hasUnit
  rdf:type owl:AnnotationProperty ;
  rdfs:label “hasUnit” ;
  rdfs:range eclass:Unit ;
.
```

The Unit from above is defined as default Unit. In context of a Quantity (see next chapter) other Units are possible. However, not all Unit make sense for a Property. Therefore, possible alternative Units can be defined via the following OWL Annotation Property:

```
eclass:hasAlternativeUnit
  rdf:type owl:AnnotationProperty ;
  rdfs:subPropertyOf eclass:hasUnit ;
  rdfs:label "hasAlternativeUnit" ;
  rdfs:range eclass:Unit ;
.
```

6.6.6 Quantity

In addition to Unit, a Quantity can be linked. To link a Quantity the following OWL Annotation Property is defined:

```
eclass:hasQuantity
  rdf:type owl:ObjectProperty;
  rdfs:label “hasQuantity” ;
  rdfs:range eclass:Quantity ;
.
```

Note: The Quantity is necessary because a Unit can belong to several Quantities. (e.g., the Unit Volt can have the Quantity Energy per Electric Charge, Voltage, Electric Potential or Electric Potential Difference)

6.6.7 Currency

For specific Data Types a Property also has a defined Currency. To link a Currency the following OWL Annotation Property is defined:

```
eclass:hasCurrency
  rdf:type owl:AnnotationProperty ;
  rdfs:label "hasCurrency" ;
  rdfs:range eclass:Currency ;
.
```

6.6.8 Value List

In addition to a free valuation of a characteristic ECLASS contains Value Lists. A Value List can be linked to a Property for a further description. In this case rdfs:range is used to define which values can used for a specific Property.

6.6.9 Examples

6.6.9.1 Example Property with Data Type String & Individuum

```
eclass:0173-X-02-XXX001-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment “Test Definition EN”@en ;
  rdfs:label “TestString”@en ;
  rdfs:range iec61360:StringType ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI “0173-X#02-XXX001#001” ;
  eclass:multivalent “true”^^xsd:boolean ;
  eclass:synonym “TestSynonym”@en ;
.
```

For an individuum of a Property another individuum defines a context. For instance, the example Aspect individuum from 6.5.2. The Property itself is just the link to an individuum from the specified Data Type:

```
example:UUID_fd3f193b-299f-4b56-827b-1bcc5a21390d
  rdf:type eclass:0173-X-01-XXX010-001 ;
  eclass:0173-X-02-XXX001-001 example:UUID_c8303724-9023-4a42-8556-6637ae238920 ;
.
```

Here, the Property from 6.6.8 points to an individuum as follows:

```
example:UUID_c8303724-9023-4a42-8556-6637ae238920
  rdf:type iec61360:StringType ;
  iec61360:value “Test” ;
.
```

This individuum has the type iec61360:StringType described with iec61360:value and the specific value “Test”.

Note: This modelling and instantiation concept is valid for all primitive Data Types without Unit, Currency, or Data Types without any other value combinations.

6.6.9.2 Example Individuum with a Property of Data Type String & Multivalued

To express more than one value several individuum of the Data Type class have to be linked via the ECLASS Property.

```
example:UUID_fd3f193b-299f-4b56-827b-1bcc5a21390d
  rdf:type eclass:0173-X-01-XXX010-001 ;
  eclass:0173-X-02-XXX001-001 example:value1 ;
  eclass:0173-X-02-XXX001-001 example:value2 ;
.

example:value1
  rdf:type iec61360:StringType ;
  iec61360:value “Test-1” ;
.

example:value2
  rdf:type iec61360:StringType ;
  iec61360:value “Test-2” ;
.
```

6.6.9.3 Example Property with Data Type Boolean (Value List)

With the Boolean Data Type, an additional Value List is linked in ECLASS to describe the Values true and false.

```
eclass:0173-X-02-XXX006-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "TestBoolean"@en ;
  rdfs:range eclass:0173-X-09-COD001-001 ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-X#02-XXX006#001" ;
  eclass:multivalent "false"^^xsd:boolean ;
.
```

Note: See 6.7.3.2 for the Value List definition.

A characteristic of this Property using a Value from this Value List would look like:

```
example:UUID_bc0fdb90-9375-442c-86bc-e946bf9600d1
  rdf:type eclass:0173-X-ADVANCED-01-XXX999-001 ;
  eclass:0173-X-02-XXX006-001 eclass:0173-X-07-XXX001-001 .
```

6.6.9.4 Example Property with Data Type IntMeasure (with Unit) & Individuum

In the following example, a Property from Data Type Integer Measure including a Unit definition is shown:

```
eclass:0173-X-02-XXX004-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment “Test Definition EN”@en ;
  rdfs:label “TestIntMeasure”@en ;
  rdfs:range iec61360:IntMeasureType ;
  rdfs:subPropertyOf eclass:nondependent ;
  eclass:IRDI “0173-X#02-XXX004#001” ;
  eclass:hasUnit eclass:0173-X-05-XXX001-001 ;
  eclass:hasAlternativeUnit eclass:0173-X-05-XXX002-001 ;
  eclass:multivalent “true”^^xsd:boolean ;
  eclass:hasQuantity eclass:0173-X-Z4-XXX001-001 ;
.
```

An individuum of iec61360:IntMeasureType would look like:

```
example:UUID_f7a3447c-ac37-43f3-9e26-4e478a2e4a57
  rdf:type iec61360:IntMeasureType ;
  iec61360:uom eclass:0173-X-05-XXX001-001 ;
  iec61360:value 1 ;
.
```

6.6.9.5 Example Property with Data Type IntCurrency (with Currency) & Individuum

In the following example, a Property from Data Type Integer Currency including a Currency definition is shown:

```
eclass:0173-X-02-XXX005-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "TestIntCurrency"@en ;
  rdfs:range iec61360:IntCurrencyType ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-X#02-XXX005#001" ;
  eclass:hasCurrency eclass:0173-X-08-XXX001-001 ;
  eclass:multivalent "true"^^xsd:boolean ;
.
```

An individuum of iec61360:IntCurrencyType would look like:

```
example:UUID_e00b4192-f527-4e9c-8b5d-a8d12d07af1c
  rdf:type iec61360:IntCurrencyType ;
  iec61360:currency eclass:0173-X-08-XXX001-001 ;
  iec61360:value 1 ;
.
```

6.6.9.6 Example Property with Data Type Class Reference

A specific Data Type is the reference. In this case, a Property references a Block. Here, the range of the Property points to the Block. In the following an example is shown:

```
eclass:0173-X-02-XXX013-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "TestReference"@en ;
  rdfs:range eclass:0173-X-01-XXX009-001 ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-X#02-XXX013#001" ;
  eclass:multivalent "true"^^xsd:boolean ;
.
```

Note: That is the reason why all Blocks are also a rdfs:subClassOf iec61360:ClassReferenceType (see 6.4)

6.6.9.7 Example Property with Data Type File & Individuum

```
eclass:0173-X-02-XXX022-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "TestFile"@en ;
  rdfs:range iec61360:FileType ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-X#02-XXX022#001" ;
  eclass:multivalent "true"^^xsd:boolean ;
.
```

The individuum is a set of value and mime type:

```
example:UUID_144d2f44-776f-473c-b9c1-b2093d1b980e
  rdf:type iec61360:FileType ;
  iec61360:mime "image/png" ;
  iec61360:value
"https://eclass.eu/fileadmin/_processed_/9/3/csm_visual_home_apfel2_31aff2f195.png"^^xs
d:anyURI ;
.
```

6.6.9.8 Example Property with Data Type Blob & Individuum

```
eclass:0173-X-02-XXX023-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "TestBlob"@en ;
  rdfs:range iec61360:BlobType ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-X#02-XXX023#001" ;
  eclass:multivalent "true"^^xsd:boolean ;
.
```

An individuum of iec61360:BlobType would look like:

```
example:UUID_9edf6203-4c9f-4b4e-9fef-1e0e988dbedd
  rdf:type iec61360:BlobType ;
  iec61360:value "iVBORw0KGgoAAA****"^^xsd:base64Binary ;
.
```

Note: The Explicit Value is shortened for the visualization.

6.6.9.9 Example Property with Level Type & Individuum

```
eclass:0173-X-02-LEV001-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "TestIntCountLevel"@en ;
  rdfs:range iec61360:LevelIntType ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-X#02-LEV001#001" ;
  eclass:multivalent "true"^^xsd:boolean ;
.
```

Note: The level enumeration for Level Type according to IEC 61360-2 (min, nom, typ, max) is omitted here. That is a possible future extension and could be integrated in Part 2.

The individuum is a set of four values:

```
example:UUID_75703e55-970d-471a-b40f-9ca1475481ee
  rdf:type iec61360:LevelIntType ;
  iec61360:maximumValue 4 ;
  iec61360:minimumValue 1 ;
  iec61360:nominalValue 3 ;
  iec61360:typicalValue 2 ;
.
```

6.6.9.10 Example Property with Data Type Axis 1 Placement & Individuum

```
eclass:0173-X-02-XXX010-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "TestAxis1"@en ;
  rdfs:range iec61360:Axis1PlacementType ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-X#02-XXX010#001" ;
  eclass:hasQuantity eclass:0173-X-Z4-XXX001-001 ;
  eclass:hasUnit eclass:0173-X-05-XXX001-001 ;
  eclass:multivalent "true"^^xsd:boolean ;
.
```

Note: The set of possible values is omitted here. That is a possible future extension and could be integrated in Part 2.

The individuum is a set of six Values:

```
example:UUID_eecd3270-17f1-4008-994e-53decbd0b591
  rdf:type iec61360:Axis1PlacementType ;
  iec61360:uom eclass:0173-X-05-XXX001-001 ;
  iec61360:xRotValue "4.0"^^xsd:float ;
  iec61360:xValue "1.0"^^xsd:float ;
  iec61360:yRotValue "5.0"^^xsd:float ;
  iec61360:yValue "2.0"^^xsd:float ;
  iec61360:zRotValue "6.0"^^xsd:float ;
  iec61360:zValue "3.0"^^xsd:float ;
.
```

6.7 Value List

A Value List defines a list of possible Values. The root of all Value Lists is as follow:

```
eclass:valuelist
  rdf:type  owl:Class ;
  rdfs:label  "Value list"@en .
```

Specific Value Lists are subclasses of this type. Furthermore, the Value List also defines the Data Type of the Values, can also be defined by a Unit (via eclass:hasUnit; see 6.6.9.4) or Currency (via eclass:hasCurrency; see 6.6.9.5). Data Type and Unit or Currency are handled the same as for Properties.

Furthermore, a Value List according to IEC 61360 and ECLASS can be implicit and explicit. To integrate this information into and to support a type safe approach like for Application Classes two further types are defined:

```
eclass:ExplicitValueList
  rdf:type owl:Class ;
  rdfs:label "ExplicitValueList" ;
  rdfs:subClassOf eclass:ValueList ;
.

eclass:ImplicitValueList
  rdf:type owl:Class ;
  rdfs:label "ImplicitValueList" ;
  rdfs:subClassOf eclass:ValueList ;
.
```

6.7.1 Values

The Value List class defines a type only. The Values themselves are important. Values are modeled as specific Individuums of the Value List. In addition, Values are type of the corresponding iec61360:DataType and the value is expressed via iec61360:value.

6.7.2 Explicit Value List

A Value can be explicit. In this case the Value has no IRDI. For a unique URI in RDF, it is defined that the Value identifier is a combination of the IRDI of the Value List and a counter of the Explicit Value.

Note: The reason for a counter in contrast to any other attribute from ECLASS is the possible character set of e.g., Value or Code.

6.7.2.1 Example Explicit Value List with Data Type String

```
eclass:0173-X-09-EXP002-001
  rdf:type owl:Class ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "StringVLE"@en ;
  rdfs:subClassOf eclass:ExplicitValueList ;
  rdfs:subClassOf iec61360:StringType ;
  eclass:IRDI "0173-X#09-EXP002#001" ;
.
```

Possible Values are as follows:

```
eclass:0173-X-09-EXP002-001-1
  rdf:type eclass:0173-X-09-EXP002-001 ;
  iec61360:value "Test1" ;
.

eclass:0173-X-09-EXP002-001-2
  rdf:type eclass:0173-X-09-EXP002-001 ;
  iec61360:value "Test2" ;
.
```

6.7.3 Implicit Value List

Furthermore, a Value List can be implicit or coded. In this case the Value has an IRDI, which is the code as well. In addition, coded Values have a preferred name and definition. Both are transferred like the other elements do.

6.7.3.1 Example Implicit Value List with Data Type String

The Value List is modeled in an equal way like the explicit one.

```
eclass:0173-X-09-COD002-001
  rdf:type owl:Class ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "StringVLC"@en ;
  rdfs:subClassOf eclass:ImplicitValueList ;
  rdfs:subClassOf iec61360:StringType ;
  eclass:IRDI "0173-X#09-COD002#001" ;
.
```

However, in contrast to the example in 6.7.2.1, here the Value has an rdfs:label, rdfs:comment and eclass:IRDI.

```
eclass:0173-X-07-XXX003-001
  rdf:type eclass:0173-X-09-COD002-001 ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "String1"@en ;
  eclass:IRDI "0173-X#07-XXX003#001" ;
  iec61360:value "Test1" ;
.

eclass:0173-X-07-XXX004-001
  rdf:type eclass:0173-X-09-COD002-001 ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "String2"@en ;
  eclass:IRDI "0173-X#07-XXX004#001" ;
  iec61360:value "Test2" ;
.
```

Note: An Implicit Value can be used in several Value Lists.

6.7.3.2 Example Implicit Value List with Data Type Boolean

```
eclass:0173-X-09-COD001-001
  rdf:type owl:Class ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "BooleanVLC"@en ;
  rdfs:subClassOf eclass:ImplicitValueList ;
  rdfs:subClassOf iec61360:BooleanType ;
  eclass:IRDI "0173-X#09-COD001#001" ;
.
```

The Value “True” would be:

```
eclass:0173-X-07-XXX001-001
  rdf:type eclass:0173-X-09-COD001-001 ;
  rdf:type iec61360:BooleanType ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "True"@en ;
  eclass:IRDI "0173-X#07-XXX001#001" ;
  iec61360:value "true"^^xsd:boolean ;
.
```

The Value “False” would be:

```
eclass:0173-X-07-XXX002-001
  rdf:type eclass:0173-X-09-COD001-001 ;
  rdf:type iec61360:BooleanType ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "False"@en ;
  eclass:IRDI "0173-X#07-XXX002#001" ;
  iec61360:value "false"^^xsd:boolean ;
.
```

6.7.4 Extensions with proprietary Values

In general (except from Data Type Boolean, in case of Polymorphism or if modelled explicit in the ECLASS Dictionary), Value Lists are suggested in ECLASS and can be extend with proprietary values, if necessary. In such case the proprietary value needs to be expressed like any other Value in ECLASS RDF. However, it should not be an individuum of the Value List itself. Only the Data Type is important to avoid accidental extensions of the dictionary.

For instance, a proprietary value for the Value List from 6.7.3.1 could look like the following:

```
example:UUID_0fd4d3b6-1e19-4932-9efc-71cd9ede5d24
  rdf:type iec61360:StringType ;
  iec61360:value "Test free Text" ;
.
```

6.8 Unit

The ECLASS Dictionary also contains a set of Units. This type has all common attributes. The root Unit class is defined as follows:

```
eclass:Unit  rdf:type  owl:Class ;
        rdfs:label  "Unit"@en .
```

In addition, a Unit has a shortName which is transferred with the following OWL Annotation Property:

```
eclass:shortName  rdf:type  owl:AnnotationProperty ;
        rdfs:label  "Short name" .
```

A Unit also points to an Aspect of Conversion which is possible via this OWL Object Property:

```
eclass:hasAoC
  rdf:type owl:ObjectProperty ;
  rdfs:label "hasAspectOfConversion" ;
  rdfs:range eclass:AoC
.
```

Moreover, a Unit points to a Conversion. This is possible via this OWL Object Property:

```
eclass:float64Conversion
  rdf:type owl:ObjectProperty ;
  rdfs:label "float64 conversion" ;
  rdfs:range eclass:Conversion ;
.
```

A Unit itself is expressed as individuum of eclass:unit .

6.8.1 Example Unit

```
eclass:0173-X-05-XXX001-001
  rdf:type eclass:Unit ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "meter"@en ;
  eclass:IRDI "0173-X#05-XXX001#001" ;
  eclass:float64Conversion eclass:a-to-a ;
  eclass:hasAoC eclass:0173-X-Z2-XXX001-001 ;
  eclass:shortName "m"@en ;
.
```

6.9 Conversion

A Conversion defines the transformation between two Units in one Quantity for instance between meter and millimeter and defines the transformation from one Unit to the initial Unit (e.g. meter). The root type of all Conversions is as follows:

```
eclass:Conversion
  rdf:type owl:Class ;
  rdfs:label "Conversion"@en ;
.
```

A Conversion itself is expressed as individuum of eclass:Conversion.

To describe the Conversion the following OWL Data Type Properties are defined:

```
eclass:multiplicand
  rdf:type owl:DatatypeProperty ;
  rdfs:label "multiplicand" ;
  rdfs:range xsd:float ;
.

eclass:divisor
  rdf:type owl:DatatypeProperty ;
  rdfs:label "divisor" ;
  rdfs:range xsd:float ;
.

eclass:initialAddent
  rdf:type owl:DatatypeProperty ;
  rdfs:label "initialAddent" ;
  rdfs:range xsd:float ;
.

eclass:finalAddent
  rdf:type owl:DatatypeProperty ;
  rdfs:label "finalAddent" ;
  rdfs:range xsd:float ;
.
```

```
eclass:initialUnit
  rdf:type owl:ObjectProperty ;
  rdfs:label "initialUnit" ;
  rdfs:range eclass:Unit ;
.
```

6.9.1 Example Conversion

```
eclass:a-to-a
  rdf:type owl:Class ;
  rdfs:subClassOf eclass:conversion ;
  eclass:divisor "1.0"^^xsd:double ;
  eclass:finalAddent "0.0"^^xsd:double ;
  eclass:initialAddent "0.0"^^xsd:double ;
  eclass:initialUnit eclass:0173-X-05-XXX001-001 ;
  eclass:multiplicand "1.0"^^xsd:double ;
.
```

6.10 Quantity

A Quantity collects a set of Units between which a conversion is defined and intended. The root type is defined as follows:

```
eclass:Quantity
  rdf:type owl:Class ;
  rdfs:label "Quantity" ;
.
```

A Quantity itself is expressed as individuum of eclass:Quantity.

To describe an eclass:Quantity and express which Units belong to this Quantity the OWL Object Property eclass:unit is defined.

```
eclass:unit
  rdf:type owl:ObjectProperty ;
  rdfs:label "unit" ;
.
```

6.10.1 Example Quantity

```
eclass:0173-X-Z4_XXX001-001
  rdf:type eclass:Quantity ;
  rdfs:comment "Test Definition EN"@en ;
  rdfs:label "length"@en ;
  eclass:IRDI "0173-X#Z4-XXX001#001" ;
  eclass:hasAoC eclass:0173-X-Z2-XXX001-001 ;
  eclass:unit eclass:0173-X-05-XXX001-001 ;
  eclass:unit eclass:0173-X-05-XXX002-001 ;
.
```

6.11 Aspect of Conversion

An Aspect of Conversion defines the physical dimensions. The root of Aspect of Conversion is as follows:

```
eclass:AoC
  rdf:type owl:Class ;
  rdfs:label "AspectOfConversion" ;
.
```

An Aspect of Conversion itself is expressed as individuum of eclass:AoC.

To describe an eclass:AoC the following OWL Data Type Properties are necessary:

```
eclass:exponent1
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponent1"@en ;
.
eclass:exponentA
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentA"@en ;
.
eclass:exponentB
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentB"@en ;
.
eclass:exponentBd
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentBd"@en ;
.
eclass:exponentBit
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentBit"@en ;
.
eclass:exponentByte
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentByte"@en ;
.
eclass:exponentCd
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentCd"@en ;
.
eclass:exponentDec
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentDec"@en ;
.
eclass:exponentE
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentE"@en ;
.
eclass:exponentHart
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentHart"@en ;
.
eclass:exponentK
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentK"@en ;
.
eclass:exponentKg
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentKg"@en ;
.
eclass:exponentM
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentM"@en ;
.
```

```
eclass:exponentMol
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentMol"@en ;
.
eclass:exponentNat
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentNat"@en ;
.
eclass:exponentNp
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentNp"@en ;
.
eclass:exponentOctave
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentOctave"@en ;
.
eclass:exponentPhon
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentPhon"@en ;
.
eclass:exponentRad
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentRad"@en ;
.
eclass:exponentS
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentS"@en ;
.
eclass:exponentSh
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentSh"@en ;
.
eclass:exponentSone
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentSone"@en ;
.
eclass:exponentSr
  rdf:type owl:DatatypeProperty ;
  rdfs:label "exponentSr"@en ;
.
```

6.11.1 Example Aspect of Conversion

```
eclass:0173-X-Z2-XXX001-001
  rdf:type eclass:AoC ;
  rdfs:label "TestAOC" ;
  eclass:IRDI "0173-X#Z2-XXX001#001" ;
  eclass:exponent1 "0"^^xsd:int ;
  eclass:exponentA "0"^^xsd:int ;
  eclass:exponentB "0"^^xsd:int ;
  eclass:exponentBd "0"^^xsd:int ;
  eclass:exponentBit "0"^^xsd:int ;
  eclass:exponentByte "0"^^xsd:int ;
  eclass:exponentCd "0"^^xsd:int ;
  eclass:exponentDec "0"^^xsd:int ;
  eclass:exponentE "0"^^xsd:int ;
  eclass:exponentHart "0"^^xsd:int ;
  eclass:exponentK "0"^^xsd:int ;
  eclass:exponentKg "0"^^xsd:int ;
```

```
  eclass:exponentM "1"^^xsd:int ;
  eclass:exponentMol "0"^^xsd:int ;
  eclass:exponentNat "0"^^xsd:int ;
  eclass:exponentNp "0"^^xsd:int ;
  eclass:exponentOctave "0"^^xsd:int ;
  eclass:exponentPhon "0"^^xsd:int ;
  eclass:exponentRad "0"^^xsd:int ;
  eclass:exponentS "0"^^xsd:int ;
  eclass:exponentSh "0"^^xsd:int ;
  eclass:exponentSone "0"^^xsd:int ;
  eclass:exponentSr "0"^^xsd:int ;
  eclass:hasQuantity eclass:0173-X-Z4-XXX001-001 ;
.
```

6.12 Currency

Additional to the Unit system ECLASS defines a set of Currencies with a specific element type. The root type is:

```
eclass:Currency
  rdf:type owl:Class ;
  rdfs:label "Currency" ;
.
```

This type has an IRDI and a preferred name expressed like always. In addition, an alphaCode is used.

Note: For alphaCode only Codes according to ISO 4217 are allowed.

This is transferred via the following OWL Annotation Property:

```
eclass:alphaCode
  rdf:type owl:AnnotationProperty;
  rdfs:label "alphaCode" ;
  rdfs:range xsd:string ;
.
```

6.12.1 Example Currency

```
eclass:0173-X-08-XXX001-001
  rdf:type eclass:Currency ;
  rdfs:label “Euro”@en ;
  eclass:IRDI “0173-X#08-XXX001#001” ;
  eclass:alphaCode “EUR” ;
.
```

6.13 Structure and Context

Besides the pure elements, ECLASS has special modeling features. How these are mapped is shown in this section. For the relation between Property and Characterization Class see 6.3.1 and for the relation between Application Class and Aspect see 6.3.2.

6.13.1 Cardinality

The Cardinality is a specific modelling feature. The feature consists of

- a Counter Property
- a Reference Property.

Note: See Part 2 of this document series for validation.

6.13.1.1 Counter Property

The Counter Property defines a condition and is from type condition via subPropertyOf eclass:conditionProperty (see 6.6.1).

6.13.1.2 Cardinal Block via Reference Property

The Block is referenced via a Property with Data Type Reference Class (see 6.6.9.6). However, the Property is a subPropertyOf eclass:dependentProperty. To express the relation between condition and the Reference Property an additional cardinality predicate is needed. Therefore, the following OWL Annotation Property is defined:

```
eclass:dependsOn rdf:type  owl:AnnotationProperty ;
        rdfs:label  “dependsOn” .
```

Furthermore, a specific Property is required to identify the order of the Blocks in a cardinality.

```
eclass:orderNumber
  rdf:type owl:DatatypeProperty ;
  rdfs:label “orderNumber” ;
.
```

6.13.1.3 Example Cardinality

Counter Property:

```
eclass:0173-X-02-CON101-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment “Test Definition EN”@en ;
  rdfs:label “TestCardinalityCounter”@en ;
  rdfs:range iec61360:IntType ;
  rdfs:subPropertyOf eclass:condition ;
  eclass:IRDI “0173-X#02-CON101#001” ;
  eclass:multivalent “false”^^xsd:boolean ;
.
```

Reference Property

```
eclass:0173-X-02-CAR102-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment “Test Definition EN”@en ;
  rdfs:label “TestCardinalityReference”@en ;
  rdfs:range eclass:0173-X-01-CAB103-001 ;
  rdfs:subPropertyOf eclass:dependent ;
  eclass:IRDI “0173-X#02-CAR102#001” ;
  eclass:dependsOn eclass:0173-X-02-CON101-001 ;
  eclass:multivalent “true”^^xsd:boolean ;
.
```

Block

```
eclass:0173-X-01-CAB103-001
  rdf:type owl:Class ;
  rdfs:comment “Test Definition EN”@en ;
  rdfs:label “TestBlockCardinality”@en ;
  rdfs:subClassOf eclass:0173-1-01-AZI000-002 ;
  rdfs:subClassOf iec61360:ClassReferenceType ;
  eclass:IRDI “0173-X#01-CAB103#001” ;
  eclass:isDescribedBy eclass:0173-X-02-XXX001-001 ;
.
```

Individuum – Context

```
example:UUID_6dfe9f91-cb53-437e-9fcc-080f6451e4db
  rdf:type eclass:0173-X-ADVANCED-01-XXX999-001 ;
  eclass:0173-X-02-CON101-001 example:UUID_96fff50a-5ec4-4eb5-94b1-fd5588008bf4 ;
  eclass:0173-X-02-CAR102-001 example:UUID_c159869d-fff2-4e97-b5a7-9e8f6c5b3d29 ;
```

Counter characteristic

```
example:UUID_96fff50a-5ec4-4eb5-94b1-fd5588008bf4
  rdf:type iec61360:IntType ;
  iec61360:value 1 ;
.
```

Individuum – Block

```
example:UUID_c159869d-fff2-4e97-b5a7-9e8f6c5b3d29
  rdf:type eclass:0173-X-01-CAB103-001 ;
  eclass:0173-X-02-XXX001-001 example:UUID_1a03181e-7db5-4462-8aba-5b0efd1cc8c8 ;
  eclass:0173-X-02-XXX001-001 example:UUID_9f0a39aa-42f4-4475-a391-cb1ef7cbc4a8 ;
  eclass:orderNumber 1
.
```

6.13.2 Polymorphism

The Polymorphism is a further specific modelling feature. The feature consists of

- a Reference Property,
- a Refenced Block as Base Block and
- n child Blocks (see 6.4).

The Base Block is referenced via the Reference Property (see 6.6.9.6). This Block contains a minimum of one Property, the Polymorphism controlling Property, which again uses a Value List (see 6.6.9.3). The child Blocks have further Properties.

Note: In Polymorphism also the reference between a selected Value and the necessary Block has to be expressed. This Class Value Assignment is not mapped in RDF, but only in SHACL. See Part 2.

Note: See Part 2 for validation.

6.13.2.1 Example Polymorphism

Initial Reference Property:

```
eclass:0173-X-02-XXX201-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment “Test Definition EN”@en ;
  rdfs:label “TestPolyReference”@en ;
  rdfs:range eclass:0173-X-01-XXX202-001 ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI “0173-X#02-XXX201#001” ;
  eclass:multivalent “true”^^xsd:boolean ;
.
```

Referenced Base Block

```
eclass:0173-X-01-XXX202-001
  rdf:type owl:Class ;
  rdfs:comment “Test Definition EN”@en ;
  rdfs:label “TestPolyBlock”@en ;
  rdfs:subClassOf eclass:0173-1-01-AZI000-002 ;
  rdfs:subClassOf iec61360:ClassReferenceType ;
  eclass:IRDI “0173-X#01-XXX202#001” ;
  eclass:isDescribedBy eclass:0173-X-02-XXX208-001 ;
.
```

Polymorphism controlling Property

```
eclass:0173-X-02-XXX208-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment “Test Definition EN”@en ;
  rdfs:label “TestPolySelection”@en ;
  rdfs:range eclass:0173-X-09-XXX205-001 ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI “0173-X#02-XXX208#001” ;
  eclass:multivalent “false”^^xsd:boolean ;
.
```

Value List and Values

```
eclass:0173-X-09-XXX205-001
  rdf:type owl:Class ;
  rdfs:comment “Test Definition EN”@en ;
  rdfs:label “PolySelectionVL”@en ;
  rdfs:subClassOf eclass:ImplicitValuelist ;
  rdfs:subClassOf iec61360:StringType ;
  eclass:IRDI “0173-X#09-XXX205#001” ;
.


eclass:0173-X-07-XXX206-001
  rdf:type eclass:0173-X-09-XXX205-001 ;
  rdfs:comment “Test Definition EN”@en ;
  rdfs:label “V1”@en ;
  eclass:IRDI “0173-X#07-XXX206#001” ;
  iec61360:value “v1” ;
.


eclass:0173-X-07-XXX207-001
  rdf:type eclass:0173-X-09-XXX205-001 ;
  rdfs:comment “Test Definition EN”@en ;
  rdfs:label “V2”@en ;
  eclass:IRDI “0173-X#07-XXX207#001” ;
  iec61360:value “v2” ;
.
```

Assumption: If the Value is eclass:0173-X-07-XXX206-001, the Block eclass:0173-X-01-XXX203-001 should be evaluated.

Child Block

```
eclass:0173-X-01-XXX203-001
  rdf:type owl:Class ;
  rdfs:comment “Test Definition EN”@en ;
  rdfs:label “TestPoly1”@en ;
  rdfs:subClassOf eclass:0173-X-01-XXX202-001 ;
  rdfs:subClassOf iec61360:ClassReferenceType ;
  eclass:IRDI “0173-X#01-XXX203#001” ;
  eclass:isDescribedBy eclass:0173-X-02-XXX001-001 ;
.
```

Individuum

```
example:UUID_16e4e806-e901-4c94-859d-c3d38ed7c223
  rdf:type eclass:0173-X-01-XXX203-001 ;
  eclass:0173-X-02-XXX001-001 example:UUID_3b85b1b9-fdbf-405a-82c7-8f76542a358b ;
  eclass:0173-X-02-XXX208-001 eclass:0173-X-07-XXX206-001 ;
.
```

By the Value eclass:0173-X-07-XXX206-001 at the Property eclass:0173-X-02-XXX208-001 is the type eclass:0173-X-01-XXX203-001.

6.13.3 Advanced Polymorphism

In addition to the Polymorphism in 6.13.2, ECLASS includes an extended version of this feature. See https://eclass.eu/support/content-creation/content-development-platform/polymorphism-help-page#c1715 for details. The Advanced Polymorphism adds a Value List to the Reference Property which is handled like a Constraint to the Polymorphism controlling Property Value List. That means not all child Blocks are possible. The selective Property allows all Values from the Value List. However, not all Blocks are allowed.

This feature is not handled in this part. It is express via SHACL. See Part 2 for details.

7 Real ECLASS Content Example

Example is done in the following structure:

- Based on ECLASS 13.0 with Release IRDI 0173-1#11-ECLASS13.0#001, defines the following namespaces:

```
@prefix eclass:   <https://eclass-cdp.com/rdf/v1/eclass/> .
@prefix eclass13-0<https://eclass-cdp.com/rdf/v1/eclass13-0/> .
@prefix iec61360: <https://eclass-cdp.com/rdf/v1/iec61360/> .
@prefix owl:      <http://www.w3.org/2002/07/owl#> .
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:     <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#> .
```

- Classification Class 0173-1#01-BAA055#019 – 27-24-26-04 - Field bus, decentralized peripheral - digital I/O module

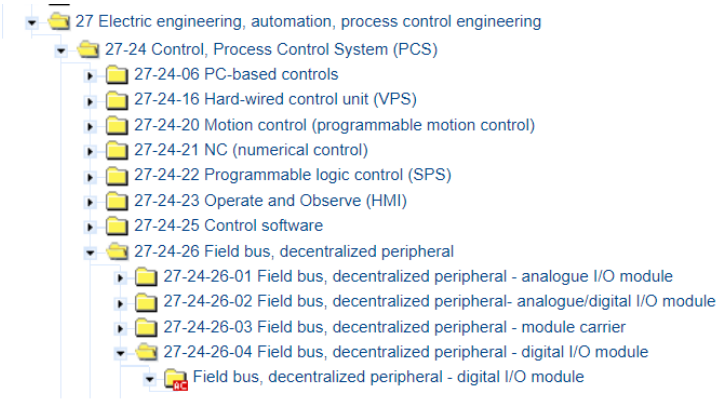

Figure 7: ECLASS 13 - Example – Classification (excerpt)

- Application Class 0173-1---ADVANCED_1_1#01-ADN579#012 - Field bus, decentralized peripheral - digital I/O module

Figure 8: ECLASS 13 - Example – Application Class (excerpt)

7.1 Classification Class

```
eclass13-0:0173-1-01-AAB572-007
  rdf:type owl:Class ;
  rdfs:label "Electric engineering, automation, process control engineering"@en ;
  rdfs:subClassOf eclass:0173-1-01-AAA002-001 ;
  eclass:IRDI "0173-1#01-AAB572#007" ;
  eclass:codedName "27000000" ;
  eclass:keyword "Electrical-engineering"@en ;
  eclass:keyword "Process control-engineering"@en ;
  eclass:technicalLabel "27-00-00-00 - Electric engineering, automation, process
control engineering" ;
.

eclass13-0:0173-1-01-AAC146-010
  rdf:type owl:Class ;
  rdfs:comment "Controlling is the directed influence of the properties of a system
from the outside by input. To that end information is to be transferred, processed and
passed on via sensors. Main group 27240000 is divided into controlling devices of
various function principles, control software and external devices (storage and field
buses). Furthermore it is divided into parts, accessories, briefing, training, start-
up, assembly, maintenance, inspection and service of control units"@en ;
  rdfs:label "Control, Process Control System (PCS)"@en ;
  rdfs:subClassOf eclass13-0:0173-1-01-AAB572-007 ;
  eclass:IRDI "0173-1#01-AAC146#010" ;
  eclass:codedName "27240000" ;
  eclass:technicalLabel "27-24-00-00 - Control, Process Control System (PCS)" ;
.

eclass13-0:0173-1-01-BAA065-010
  rdf:type owl:Class ;
  rdfs:label "Field bus, decentralized peripheral"@en ;
  rdfs:subClassOf eclass13-0:0173-1-01-AAC146-010 ;
  eclass:IRDI "0173-1#01-BAA065#010" ;
  eclass:codedName "27242600" ;
  eclass:keyword "Decentralized peripheral component"@en ;
  eclass:keyword "Decentralized peripheral system"@en ;
  eclass:keyword "Field bus, decentralised peripheral"@en ;
  eclass:technicalLabel "27-24-26-00 - Field bus, decentralized peripheral" ;
.


eclass13-0:0173-1-01-BAA055-019
  rdf:type owl:Class ;
  rdfs:label "Field bus, decentralized peripheral - digital I/O module"@en ;
  rdfs:subClassOf eclass13-0:0173-1-01-BAA065-010 ;
  eclass:IRDI "0173-1#01-BAA055#019" ;
  eclass:codedName "27242604" ;
  eclass:keyword "Add-on module"@en ;
  eclass:keyword "Decentralized compact digital input/output module"@en ;
  eclass:keyword "Decentralized compact similar I/O module"@en ;
  eclass:keyword "Digital input"@en ;
  eclass:keyword "Digital module"@en ;
  eclass:keyword "Digital output"@en ;
  eclass:keyword "Input subrack"@en ;
  eclass:keyword "Input/output module"@en ;
  eclass:keyword "Local add-on module"@en ;
  eclass:keyword "Output subrack"@en ;
```

```
  eclass:keyword "Safety-oriented fieldbus - input/output module"@en ;
  eclass:keyword "Safety-oriented fieldbus – digital Input/output module"@en ;
  eclass:technicalLabel "27-24-26-04 - Field bus, decentralized peripheral - digital
I/O module" ;
.
```

7.2 Application Class

```
eclass13-0:0173-1---ADVANCED_1_1-01-ADN579-012
  rdf:type owl:Class ;
  rdfs:label "Field bus, decentralized peripheral - digital I/O module"@en ;
  rdfs:subClassOf eclass13-0:0173-1-01-BAA055-019 ;
  rdfs:subClassOf eclass:Advanced ;
  eclass:IRDI "0173-1---ADVANCED_1_1#01-ADN579#012" ;
  eclass:hasAspect eclass13-0:0173-1-01-ADN228-011 ;
  eclass:hasAspect eclass13-0:0173-1-01-ADN229-010 ;
#shortened
  eclass:isDescribedBy eclass13-0:0173-1-02-AAB728-007 ;
  eclass:isDescribedBy eclass13-0:0173-1-02-AAB907-008 ;
#shortened
.
```

Note: Not all hasAspect & isDescribedBy are shown.

7.3 Property

```
eclass13-0:0173-1-02-AAP798-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment "Reference to one or more norms and guidelines instead of repeating
detailed specs. COMMENT: An application norm is a specification of a pattern of use
which must be followed during use, or of a quality which must be maintained during use.
It serves to cause artifacts which arise independently of one another to be
manufactured with uniform properties or qualities. National and international standards
are established by so-called standards committees"@en ;
  rdfs:label "Application standards"@en ;
  rdfs:range iec61360:StringType ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-1#02-AAP798#001" ;
  eclass:multivalent "true"^^xsd:boolean ;
.
```

7.3.1 Property with Unit

```
eclass13-0:0173-1-02-BAD046-007
  rdf:type owl:ObjectProperty ;
  rdfs:comment "Maximum required time at change of signal between when a required
function is triggered and when execution of it is completed"@en ;
  rdfs:label "Max. delay time with change of signal"@en ;
  rdfs:range iec61360:RealMeasureType ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-1#02-BAD046#007" ;
  eclass:hasQuantity eclass13-0:0173-1-Z4-BAJ217-003 ;
  eclass:hasUnit eclass13-0:0173-1-05-AAA114-003 ;
  eclass:multivalent "true"^^xsd:boolean ;
  eclass:synonym "delay time with change of signal, max"@en ;
.
#unit
eclass13-0:0173-1-05-AAA114-003
  rdf:type eclass:Unit ;
  rdfs:comment "0,001-fold of the SI base unit second"@en ;
  rdfs:label "millisecond"@en ;
  eclass:IRDI "0173-1#05-AAA114#003" ;
  eclass:float64Conversion eclass13-0:AAA114-to-AAA203 ;
  eclass:float64Conversion eclass13-0:AAA114-to-AAA452 ;
  eclass:hasAoC eclass13-0:0173-1-Z2-AAA168-001 ;
  eclass:shortName "ms"@en ;
.

#Aspect of conversion
eclass13-0:0173-1-Z2-AAA168-001
  rdf:type eclass:AoC ;
  rdfs:label "s" ;
  eclass:IRDI "0173-1#Z2-AAA168#001" ;
  eclass:exponent1 "0"^^xsd:int ;
  eclass:exponentA "0"^^xsd:int ;
  eclass:exponentB "0"^^xsd:int ;
  eclass:exponentBd "0"^^xsd:int ;
  eclass:exponentBit "0"^^xsd:int ;
  eclass:exponentByte "0"^^xsd:int ;
  eclass:exponentCd "0"^^xsd:int ;
  eclass:exponentDec "0"^^xsd:int ;
  eclass:exponentE "0"^^xsd:int ;
  eclass:exponentHart "0"^^xsd:int ;
  eclass:exponentK "0"^^xsd:int ;
  eclass:exponentKg "0"^^xsd:int ;
  eclass:exponentM "0"^^xsd:int ;
  eclass:exponentMol "0"^^xsd:int ;
  eclass:exponentNat "0"^^xsd:int ;
  eclass:exponentNp "0"^^xsd:int ;
  eclass:exponentOctave "0"^^xsd:int ;
  eclass:exponentPhon "0"^^xsd:int ;
  eclass:exponentRad "0"^^xsd:int ;
  eclass:exponentS "1"^^xsd:int ;
  eclass:exponentSh "0"^^xsd:int ;
  eclass:exponentSone "0"^^xsd:int ;
  eclass:exponentSr "0"^^xsd:int ;
  eclass:hasQuantity eclass13-0:0173-1-Z4-BAJ217-003 ;
.
```

```
#Quantity
eclass13-0:0173-1-Z4-BAJ217-003
  rdf:type eclass:Quantity ;
  rdfs:comment "non-negative quantity attributed to a time interval, the value of which
is equal to the difference between the time points of the final instant and the initial
instant of the time interval, when the time points are quantitative marks"@en ;
  rdfs:label "time"@en ;
  eclass:IRDI "0173-1#Z4-BAJ217#003" ;
  eclass:hasAoc eclass:0173-1-Z2-AAA168-001 ;
  eclass:unit eclass13-0:0173-1-05-AAA021-003 ;
  eclass:unit eclass13-0:0173-1-05-AAA039-003 ;
  eclass:unit eclass13-0:0173-1-05-AAA095-004 ;
#shortened
.
```

7.3.1.1 Property with Value List - Implicit

```
eclass13-0:0173-1-02-BAC065-008
  rdf:type owl:ObjectProperty ;
  rdfs:comment "Required type of voltage at the voltage input of the device"@en ;
  rdfs:label "Input voltage, type"@en ;
  rdfs:range eclass13-0:0173-1-09-AAB500-005 ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-1#02-BAC065#008" ;
  eclass:multivalent "false"^^xsd:boolean ;
.

#value list
eclass13-0:0173-1-09-AAB500-005
  rdf:type owl:Class ;
  rdfs:comment "Required type of voltage at the voltage input of the device."@en ;
  rdfs:label "Input voltage, type"@en ;
  rdfs:subClassOf eclass:ImplicitValuelist  ;
  rdfs:subClassOf iec61360:StringType ;
  eclass:IRDI "0173-1#09-AAB500#005" ;
.

#one value
eclass:0173-1-07-BAA161-001
  rdf:type eclass13-0:0173-1-09-AAB477-004 ;
  rdf:type eclass13-0:0173-1-09-AAB500-005 ;
  rdf:type eclass13-0:0173-1-09-AAB856-005 ;
  rdfs:label "AC"@en ;
  eclass:IRDI "0173-1#07-BAA161#001" ;
  iec61360:value "BAA161" ;
.
```

7.3.1.2 Property with Value List – Explicit

```
eclass13-0:0173-1-02-ABH413-001
  rdf:type owl:ObjectProperty ;
  rdfs:comment "part of the transmission protocol used for characterizing the
transmitted information units (e.g. character-by-character or block-by-block
transmission) and as a specification or agreement for controlling asynchronous
transmission"@en ;
  rdfs:label "transmission format at telecontrol"@en ;
  rdfs:range eclass13-0:0173-1-09-AAO361-001 ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-1#02-ABH413#001" ;
  eclass:hasQuantity eclass13-0:0173-1-Z4-BAJ436-002 ;
  eclass:hasUnit eclass13-0:0173-1-05-AAA578-003 ;
  eclass:multivalent "true"^^xsd:boolean ;
.

#Value list
eclass13-0:0173-1-09-AAO361-001
  rdf:type owl:Class ;
  rdfs:label "transmission format at telecontrol"@en ;
  rdfs:subClassOf eclass:ExplicitValuelist ;
  rdfs:subClassOf iec61360:IntMeasureType ;
  eclass:IRDI "0173-1#09-AAO361#001" ;
.

#Example value
eclass13-0:0173-1-09-AAO361-001
  rdf:type eclass:0173-1-09-AAO361-001 ;
  iec61360:uom eclass:0173-1-05-AAA578-003 ;
  iec61360:value 10 ;
.
```

7.3.2 Property with Reference

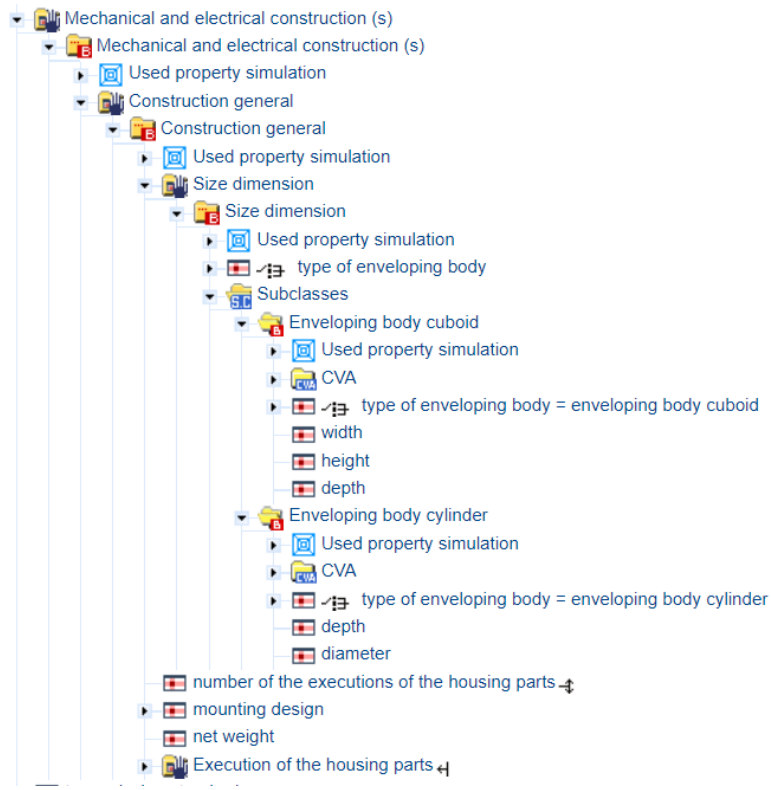

Figure 9: ECLASS 13 - Example – Mechanical electrical construction

```
#Refernence Property
eclass13-0:0173-1-02-AAR080-012
  rdf:type owl:ObjectProperty ;
  rdfs:comment "Building criteria in the areas of mechanical and electrical"@en ;
  rdfs:label "Mechanical and electrical construction (s)"@en ;
  rdfs:range eclass13-0:0173-1-01-ADS444-012 ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-1#02-AAR080#012" ;
  eclass:multivalent "true"^^xsd:boolean ;
.
```

Note: See Block in the next chapter.

7.4 Block

Note: see Figure 9.

```
#Block
eclass13-0:0173-1-01-ADS444-012
  rdf:type owl:Class ;
  rdfs:comment "Building criteria in the areas of mechanical and electrical"@en ;
  rdfs:label "Mechanical and electrical construction (s)"@en ;
  rdfs:subClassOf eclass13-0:0173-1-01-ADR430-012 ;
  rdfs:subClassOf iec61360:ClassReferenceType ;
  eclass:IRDI "0173-1#01-ADS444#012" ;
.

#Parent-Block
eclass13-0:0173-1-01-ADR430-012
  rdf:type owl:Class ;
  rdfs:comment "containing the constructional details of the device and its parts"@en ;
  rdfs:label "Mechanical and electrical construction"@en ;
  rdfs:subClassOf eclass13-0:0173-1-01-AZI000-002 ;
  rdfs:subClassOf iec61360:ClassReferenceType ;
  eclass:IRDI "0173-1#01-ADR430#012" ;
  eclass:isDescribedBy eclass13-0:0173-1-02-AAQ640-012 ;
.
```

7.5 Aspect

```
eclass13-0:0173-1-01-ADN228-011
  rdf:type owl:Class ;
  rdfs:comment "information necessary for unambiguous identification of the device or a
component thereof"@en ;
  rdfs:label "Identification"@en ;
  rdfs:subClassOf eclass:0173-1-01-PAA001-001 ;
  eclass:IRDI "0173-1#01-ADN228#011" ;
  eclass:isDescribedBy eclass13-0:0173-1-02-AAQ373-011 ;
  eclass:isDescribedBy eclass13-0:0173-1-02-AAQ376-010 ;
.
```

7.6 Cardinality

Note: see Figure 9.

```
#Counter
eclass13-0:0173-1-02-AAN520-003
  rdf:type owl:ObjectProperty ;
  rdfs:comment "cardinality property"@en ;
  rdfs:label "number of the executions of the housing parts"@en ;
  rdfs:range iec61360:IntType ;
  rdfs:subPropertyOf eclass:condition ;
  eclass:IRDI "0173-1#02-AAN520#003" ;
  eclass:multivalent "true"^^xsd:boolean ;
.

#Cardinal Reference Property
eclass13-0:0173-1-02-AAQ666-010
  rdf:type owl:ObjectProperty ;
  rdfs:comment "Design of the object"@en ;
  rdfs:label "Execution of the housing parts"@en ;
  rdfs:range eclass:0173-1-01-ADN460-010 ;
  rdfs:subPropertyOf eclass:dependent ;
  eclass:IRDI "0173-1#02-AAQ666#010" ;
  eclass:dependsOn eclass13-0:0173-1-02-AAN520-003 ;
  eclass:multivalent "true"^^xsd:boolean ;
.
```

7.7 Polymorphism

Note: see Figure 9.

```
#Reference Property
eclass13-0:0173-1-02-AAQ667-004
  rdf:type owl:ObjectProperty ;
  rdfs:comment "Expansion of a body"@en ;
  rdfs:label "Size dimension"@en ;
  rdfs:range eclass13-0:0173-1-01-ADN458-004 ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-1#02-AAQ667#004" ;
  eclass:multivalent "true"^^xsd:boolean ;
.

#Block
eclass13-0:0173-1-01-ADN458-004
  rdf:type owl:Class ;
  rdfs:comment "Expansion of a body"@en ;
  rdfs:label "Size dimension"@en ;
  rdfs:subClassOf eclass:0173-1-01-AZI000-002 ;
  rdfs:subClassOf iec61360:ClassReferenceType ;
  eclass:IRDI "0173-1#01-ADN458#004" ;
  eclass:isDescribedBy eclass13-0:0173-1-02-AAN483-003 ;
.
```

```
#polymorphism controlling Property
eclass13-0:0173-1-02-AAN483-003
  rdf:type owl:ObjectProperty ;
  rdfs:comment "selection of the object shape"@en ;
  rdfs:label "type of enveloping body"@en ;
  rdfs:range eclass13-0:0173-1-09-AAD488-002 ;
  rdfs:subPropertyOf eclass:nonDependent ;
  eclass:IRDI "0173-1#02-AAN483#003" ;
  eclass:multivalent "true"^^xsd:boolean ;
.

#Child-Block 1
eclass13-0:0173-1-01-ADS435-007
  rdf:type owl:Class ;
  rdfs:comment "Enveloping reduce complex objects, which may consist of a large number
of polygons in a fundamental, geometric Körper.Hüllkörper not be made visible, but are
auxiliary constructions in order to simplify calculations and accelerate"@en ;
  rdfs:label "Enveloping body cuboid"@en ;
  rdfs:subClassOf eclass13-0:0173-1-01-ADN458-004 ;
  rdfs:subClassOf iec61360:ClassReferenceType ;
  eclass:IRDI "0173-1#01-ADS435#007" ;
  eclass:isDescribedBy eclass13-0:0173-1-02-BAA020-010 ;
  eclass:isDescribedBy eclass13-0:0173-1-02-BAB577-008 ;
  eclass:isDescribedBy eclass13-0:0173-1-02-BAF016-006 ;
.

#Child Block 2
eclass13-0:0173-1-01-ADS436-007
  rdf:type owl:Class ;
  rdfs:comment "Enveloping reduce complex objects, which may consist of a large number
of polygons in a fundamental, geometric Körper.Hüllkörper not be made visible, but are
auxiliary constructions in order to simplify calculations and accelerate"@en ;
  rdfs:label "Enveloping body cylinder"@en ;
  rdfs:subClassOf eclass13-0:0173-1-01-ADN458-004 ;
  rdfs:subClassOf iec61360:ClassReferenceType ;
  eclass:IRDI "0173-1#01-ADS436#007" ;
  eclass:isDescribedBy eclass13-0:0173-1-02-AAC895-007 ;
  eclass:isDescribedBy eclass13-0:0173-1-02-BAB577-008 ;
.
```

8 References

- Apache Jena: A free and open-source Java framework for building Semantic Web and Linked Data applications. https://jena.apache.org/ (RDF API & Fuseki - Version 4.6.0)

Annex A – Turtle Representation of ECLASS Conceptual Model

```
@prefix eclass:   <https://eclass-cdp.com/rdf/v1/eclass/> .
@prefix iec61360: <https://eclass-cdp.com/rdf/v1/iec61360/> .
@prefix owl:      <http://www.w3.org/2002/07/owl#> .
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:     <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#> .

eclass:AoC  rdf:type  owl:Class;
        rdfs:label  "AspectOfConversion" .

eclass:initialAddent  rdf:type  owl:DatatypeProperty;
        rdfs:label  "initialAddent";
        rdfs:range  xsd:float .

eclass:nonDependent  rdf:type  owl:ObjectProperty;
        rdfs:label          "nonDependentProperty";
        rdfs:subPropertyOf  eclass:property .

eclass:exponentSone  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentSone" .

eclass:exponentSh  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentSh" .

eclass:exponentDec  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentDec" .

eclass:exponentE  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentE" .

eclass:ImplicitValueList
        rdf:type         owl:Class;
        rdfs:label       "ImplicitValueList";
        rdfs:subClassOf  eclass:ValueList .

eclass:finalAddent  rdf:type  owl:DatatypeProperty;
        rdfs:label  "finalAddent";
        rdfs:range  xsd:float .

eclass:exponentCd  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentCd" .

eclass:exponentK  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentK" .

eclass:exponentOctave
        rdf:type    owl:DatatypeProperty;
        rdfs:label  "exponentOctave" .

eclass:0173-1-01-PAA001-001
        rdf:type     owl:Class;
        rdfs:label   "Aspect";
        eclass:IRDI  "0173-1#01-PAA001#001" .

eclass:float64Conversion
        rdf:type    owl:ObjectProperty;
```

```
        rdfs:label  "float64Conversion";
        rdfs:range  eclass:Conversion .

eclass:keyword  rdf:type  owl:AnnotationProperty;
        rdfs:label  "keyword";
        rdfs:range  rdf:langString .

eclass:dependent  rdf:type  owl:ObjectProperty;
        rdfs:label          "dependentProperty";
        rdfs:subPropertyOf  eclass:property .

eclass:divisor  rdf:type  owl:DatatypeProperty;
        rdfs:label  "divisor";
        rdfs:range  xsd:float .

eclass:multiplicand  rdf:type  owl:DatatypeProperty;
        rdfs:label  "multiplicand";
        rdfs:range  xsd:float .

eclass:Unit  rdf:type  owl:Class;
        rdfs:label  "Unit" .

eclass:exponentNp  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentNp" .

eclass:alphaCode  rdf:type  owl:AnnotationProperty;
        rdfs:label  "alphaCode";
        rdfs:range  xsd:string .

eclass:exponentKg  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentKg" .

eclass:0173-1-01-AAA002-001
        rdf:type     owl:Class;
        rdfs:label   "ClassificationClass";
        eclass:IRDI  "0173-1#01-AAA002#001" .

eclass:Conversion  rdf:type  owl:Class;
        rdfs:label  "Conversion" .

eclass:hasAlternativeUnit
        rdf:type            owl:AnnotationProperty;
        rdfs:label          "hasAlternativeUnit";
        rdfs:range          eclass:Unit;
        rdfs:subPropertyOf  eclass:hasUnit .

eclass:codedName  rdf:type  owl:AnnotationProperty;
        rdfs:label  "codedName";
        rdfs:range  xsd:string .

eclass:hasAoC  rdf:type  owl:ObjectProperty;
        rdfs:label  "hasAspectOfConversion";
        rdfs:range  eclass:AoC .

eclass:exponentA  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentA" .

eclass:condition  rdf:type  owl:ObjectProperty;
        rdfs:label          "conditionProperty";
```

```
        rdfs:subPropertyOf  eclass:property .

eclass:initialUnit  rdf:type  owl:ObjectProperty;
        rdfs:label  "initialUnit";
        rdfs:range  eclass:Unit .

eclass:technicalLabel
        rdf:type    owl:AnnotationProperty;
        rdfs:label  "technicalLabel";
        rdfs:range  xsd:string .

eclass:hasAspect  rdf:type  owl:AnnotationProperty;
        rdfs:label  "hasAspect";
        rdfs:range  eclass:0173-1-01-PAA001-001 .

eclass:ValueList  rdf:type  owl:Class;
        rdfs:label  "ValueList" .

eclass:Quantity  rdf:type  owl:Class;
        rdfs:label  "Quantity" .

eclass:exponentM  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentM" .

eclass:ExplicitValueList
        rdf:type         owl:Class;
        rdfs:label       "ExplicitValueList";
        rdfs:subClassOf  eclass:ValueList .

eclass:unit  rdf:type  owl:ObjectProperty;
        rdfs:label  "unit" .

eclass:Basic  rdf:type   owl:Class;
        rdfs:label       "ApplicationClass-BASIC";
        rdfs:subClassOf  eclass:0173---BASIC_1_1-01-AAA001-001 .

eclass:Asset  rdf:type   owl:Class;
        rdfs:label       "ApplicationClass-ASSET";
        rdfs:subClassOf  eclass:0173---BASIC_1_1-01-AAA001-001 .

eclass:exponentMol  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentMol" .

eclass:exponentByte  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentByte" .

eclass:exponentPhon  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentPhon" .

eclass:isDescribedBy  rdf:type  owl:AnnotationProperty;
        rdfs:label  "isDescribedBy" .

eclass:exponentS  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentS" .

eclass:exponentB  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentB" .

eclass:Currency  rdf:type  owl:Class;
```

```
        rdfs:label  "Currency" .

eclass:exponent1  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponent1" .

eclass:property  rdf:type  owl:ObjectProperty;
        rdfs:label  "property" .

eclass:dependsOn  rdf:type  owl:AnnotationProperty;
        rdfs:label  "dependsOn" .

eclass:0173-1-01-AZI000-002
        rdf:type     owl:Class;
        rdfs:label   "Block";
        eclass:IRDI  "0173-1#01-AZI000#002" .

eclass:exponentNat  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentNat" .

eclass:exponentBd  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentBd" .

eclass:Advanced  rdf:type  owl:Class;
        rdfs:label       "ApplicationClass-ADVANCED";
        rdfs:subClassOf  eclass:0173---BASIC_1_1-01-AAA001-001 .

eclass:synonym  rdf:type  owl:AnnotationProperty;
        rdfs:label  "synonym";
        rdfs:range  rdf:langString .

eclass:0173---BASIC_1_1-01-AAA001-001
        rdf:type     owl:Class;
        rdfs:label   "ApplicationClass";
        eclass:IRDI  "0173---BASIC_1_1#01-AAA001#001" .

eclass:exponentHart  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentHart" .

eclass:exponentBit  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentBit" .

eclass:orderNumber  rdf:type  owl:DatatypeProperty;
        rdfs:label  "orderNumber" .

eclass:IRDI  rdf:type  owl:AnnotationProperty;
        rdfs:label  "IRDI";
        rdfs:range  xsd:string .

eclass:exponentSr  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentSr" .

eclass:hasQuantity  rdf:type  owl:ObjectProperty;
        rdfs:label  "hasQuantity";
        rdfs:range  eclass:Quantity .

eclass:exponentRad  rdf:type  owl:DatatypeProperty;
        rdfs:label  "exponentRad" .

eclass:hasUnit  rdf:type  owl:AnnotationProperty;
```

```
        rdfs:label  "hasUnit";
        rdfs:range  eclass:Unit .

eclass:hasCurrency  rdf:type  owl:AnnotationProperty;
        rdfs:label  "hasCurrency";
        rdfs:range  eclass:Currency .

eclass:aspect  rdf:type  owl:ObjectProperty;
        rdfs:comment  "This Property is used to link an Aspect in a product description
individuum."@en;
        rdfs:label    "aspect";
        rdfs:range    eclass:0173-1-01-PAA001-001 .

eclass:multivalent  rdf:type  owl:AnnotationProperty;
        rdfs:label  "multivalent";
        rdfs:range  xsd:boolean .

eclass:shortName  rdf:type  owl:AnnotationProperty;
        rdfs:label  "shortName";
        rdfs:range  xsd:string .
```

Annex B – Turtle Representation of IEC 61360 Data Types

```
@prefix iec61360: <https://eclass-cdp.com/rdf/v1/iec61360/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

iec61360:DataType
  a owl:Class ;
  rdfs:label "DataType" ;
.
iec61360:NumberType
  a owl:Class ;
  rdfs:label "NumberType" ;
  rdfs:subClassOf iec61360:DataType ;
.
iec61360:IntType
  a owl:Class ;
  rdfs:label "IntType" ;
  rdfs:subClassOf iec61360:NumberType ;
.
iec61360:IntCurrencyType
  a owl:Class ;
  rdfs:label "IntCurrencyType" ;
  rdfs:subClassOf iec61360:IntType ;
.
iec61360:IntMeasureType
  a owl:Class ;
  rdfs:label "IntMeasureType" ;
  rdfs:subClassOf iec61360:IntType ;
.
iec61360:RationalType
  a owl:Class ;
  rdfs:label "RationalType" ;
  rdfs:subClassOf iec61360:NumberType ;
.
iec61360:RationalMeasureType
  a owl:Class ;
  rdfs:label "RationalMeasureType" ;
  rdfs:subClassOf iec61360:RationalType ;
.
iec61360:RealType
  a owl:Class ;
  rdfs:label "RealType" ;
  rdfs:subClassOf iec61360:NumberType ;
.
iec61360:RealCurrencyType
  a owl:Class ;
  rdfs:label "RealCurrencyType" ;
  rdfs:subClassOf iec61360:RealType ;
.
iec61360:RealMeasureType
  a owl:Class ;
  rdfs:label "RealMeasureType" ;
  rdfs:subClassOf iec61360:RealType ;
.
```

```
iec61360:StringType
  a owl:Class ;
  rdfs:label "StringType" ;
  rdfs:subClassOf iec61360:DataType ;
.
iec61360:NonTranslatableStringType
  a owl:Class ;
  rdfs:label "NonTranslatableStringType" ;
  rdfs:subClassOf iec61360:DataType ;
.
iec61360:TranslatableStringType
  a owl:Class ;
  rdfs:label "TranslatableStringType" ;
  rdfs:subClassOf iec61360:DataType ;
.
iec61360:PlacementType
  a owl:Class ;
  rdfs:label "PlacementType" ;
  rdfs:subClassOf iec61360:DataType ;
.
iec61360:Axis1PlacementType
  a owl:Class ;
  rdfs:label "Axis1PlacementType" ;
  rdfs:subClassOf iec61360:PlacementType ;
.
iec61360:Axis2Placement2DType
  a owl:Class ;
  rdfs:label "Axis2Placement2DType" ;
  rdfs:subClassOf iec61360:PlacementType ;
.
iec61360:Axis2Placement3DType
  a owl:Class ;
  rdfs:label "Axis2Placement3DType" ;
  rdfs:subClassOf iec61360:PlacementType ;
.
iec61360:BlobType
  a owl:Class ;
  rdfs:label "BlobType" ;
  rdfs:subClassOf iec61360:DataType ;
.
iec61360:BooleanType
  a owl:Class ;
  rdfs:label "BooleanType" ;
  rdfs:subClassOf iec61360:DataType ;
.
iec61360:ClassReferenceType
  a owl:Class ;
  rdfs:label "ClassReferenceType" ;
  rdfs:subClassOf iec61360:DataType ;
.
iec61360:UriType
  a owl:Class ;
  rdfs:subClassOf iec61360:DataType ;
  rdfs:label "UriType" ;
.
iec61360:FileType
  a owl:Class ;
  rdfs:subClassOf iec61360:UriType ;
  rdfs:label "FileType" ;
```

```
.
iec61360:DateType
  a owl:Class ;
  rdfs:subClassOf iec61360:DataType ;
  rdfs:label "DateDataType" ;
.
iec61360:TimeType
  a owl:Class ;
  rdfs:subClassOf iec61360:DataType ;
  rdfs:label "TimeDataType" ;
.
iec61360:DateTimeType
  a owl:Class ;
  rdfs:subClassOf iec61360:DataType ;
  rdfs:label "DateTimeDataType" ;
.
iec61360:LevelType
  a owl:Class ;
  rdfs:label "LevelType" ;
  rdfs:subClassOf iec61360:DataType ;
.
iec61360:LevelIntType
  a owl:Class ;
  rdfs:subClassOf iec61360:LevelType ;
.
iec61360:LevelIntMeasureType
  a owl:Class ;
  rdfs:subClassOf iec61360:LevelIntType ;
.
iec61360:LevelIntCurrencyType
  a owl:Class ;
  rdfs:subClassOf iec61360:LevelIntType ;
.
iec61360:LevelRealType
  a owl:Class ;
  rdfs:subClassOf iec61360:LevelType ;
.
iec61360:LevelRealMeasureType
  a owl:Class ;
  rdfs:subClassOf iec61360:LevelRealType ;
.
iec61360:LevelRealCurrencyType
  a owl:Class ;
  rdfs:subClassOf iec61360:LevelRealType ;
.
iec61360:LevelDateType
  a owl:Class ;
  rdfs:subClassOf iec61360:LevelType ;
.
iec61360:LevelTimeType
  a owl:Class ;
  rdfs:subClassOf iec61360:LevelType ;
.
iec61360:LevelTimestampType
  a owl:Class ;
  rdfs:subClassOf iec61360:LevelType ;
.
iec61360:value
      a owl:ObjectProperty ;
```

```
      rdfs:label "value" ;
.
iec61360:uom
      a owl:ObjectProperty ;
      rdfs:label "unitOfMeasure" ;
.
iec61360:currency
      a owl:DatatypeProperty ;
      rdfs:label "currency" ;
.
iec61360:mime
      a owl:DatatypeProperty ;
      rdfs:label "mimeType" ;
.
iec61360:wholePart
  a owl:DatatypeProperty ;
  rdfs:label "wholePart" ;
  rdfs:subPropertyOf iec61360:value ;
.
iec61360:numerator
  a owl:DatatypeProperty ;
  rdfs:label "numerator " ;
  rdfs:subPropertyOf iec61360:value ;
.
iec61360:denominator
  a owl:DatatypeProperty ;
  rdfs:label "denominator" ;
  rdfs:subPropertyOf iec61360:value ;
.
iec61360:maximumValue
  a owl:DatatypeProperty ;
  rdfs:label "maximumValue" ;
  rdfs:subPropertyOf iec61360:value ;
.
iec61360:minimumValue
  a owl:DatatypeProperty ;
  rdfs:label "minimumValue" ;
  rdfs:subPropertyOf iec61360:value ;
.
iec61360:nominalValue
  a owl:DatatypeProperty ;
  rdfs:label "nominalValue" ;
  rdfs:subPropertyOf iec61360:value ;
.
iec61360:typicalValue
  a owl:DatatypeProperty ;
  rdfs:label "typicalValue" ;
  rdfs:subPropertyOf iec61360:value ;
.
iec61360:xRotValue
  a owl:DatatypeProperty ;
  rdfs:label "xRotValue" ;
  rdfs:subPropertyOf iec61360:value ;
.
iec61360:xValue
  a owl:DatatypeProperty ;
  rdfs:label "xValue" ;
  rdfs:subPropertyOf iec61360:value ;
.
```

```
iec61360:yRotValue
  a owl:DatatypeProperty ;
  rdfs:label "yRotValue" ;
  rdfs:subPropertyOf iec61360:value ;
.
iec61360:yValue
  a owl:DatatypeProperty ;
  rdfs:label "yValue" ;
  rdfs:subPropertyOf iec61360:value ;
.
iec61360:zRotValue
  a owl:DatatypeProperty ;
  rdfs:label "zRotValue" ;
  rdfs:subPropertyOf iec61360:value ;
.
iec61360:zValue
  a owl:DatatypeProperty ;
  rdfs:label "zValue" ;
  rdfs:subPropertyOf iec61360:value ;
.
```