TECHNICAL SPECIFICATION

ISO/TS 10303-17

First edition
2022-03

Industrial automation systems and integration — Product data representation and exchange —

Part 17: Description methods: EXPRESS to SysML CXMI transformation

Systèmes d'automatisation industrielle et intégration — Représentation et échange de données de produits —

Partie 17: Méthodes de description: Transformation EXPRESS vers SysML CXMInque

Reference number
ISO/TS 10303-17:2022(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 4, *Industrial data.*

A list of all parts in the ISO 10303 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The ISO 10303 series of international standards describe the computer-interpretable representation of product information in the exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product and independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases and archiving.

The EXPRESS language and the EXPRESS-G diagrams are specified in ISO 10303-11. It is used to specify the information requirements of other parts of the ISO 10303 series.

This document is a member of the description methods series. This document specifies a mapping of EXPRESS to SysML CXMI. This document supports the STEP Extended Architecture[9][10][11].

The Object Management Group (OMG) has standardized the XML Metadata Interchange specification (XMI), and the Canonical XML Metadata Interchange specification (CXMI) that integrates the OMG Systems Modeling Language (SysML), the OMG Unified Modeling Language (UML), and the World Wide Web Consortium (W3C) Extensible Mark-up Language (XML)[12]. SysML inherits the CXMI interchange capability from UML. CXMI is a mechanism for the interchange of metadata and formal diagrams between UML-based modeling tools. OMG has also standardized an CXMI compliant interchange format for the SysML thus specifying a lexical representation of SysML models based on a standardized metamodel of the SysML.

This document specifies a description method of the STEP parts family, which defines the transformation of a subset of EXPRESS constructs to SysML. Because the CXMI standard specifies the XML representation of SysML metamodel constructs, standardizing the mapping of EXPRESS constructs into SysML constructs supports the representation of EXPRESS schemas as SysML models.

The specified mapping is a one-way transformation from a subset of an EXPRESS schema to a SysML model represented by an CXMI specification. These limitations make the mapping unsuitable for the transformation of arbitrary EXPRESS schemas to SysML models.

A detailed knowledge of the of the EXPRESS and SysML languages is useful.

Industrial automation systems and integration — Product data representation and exchange —

Part 17: Description methods: EXPRESS to SysML CXMI transformation

1 Scope

This document specifies a mapping of EXPRESS (ISO 10303–11) constructs to SysML constructs for the purpose of representing EXPRESS schemas in SysML models. The specified mapping is a one-way transformation from an EXPRESS schema to a SysML information model.

The following are within the scope of this document:

— the specification of the transformation of EXPRESS constructs to SysML constructs represented in CXMI;

— the specification of the transformation of EXPRESS UNIQUE rules to SysML constructs represented in CXMI;

— the specification of the transformation of derived attributes to implement renaming to SysML constructs represented in CXMI.

The following are outside the scope of this document:

— the transformation of EXPRESS elements into SysML metamodel constructs that are not used in the STEP Extended Architecture[9][10][11];

— transformation of EXPRESS FUNCTIONS;

— transformation of EXPRESS-G to SysML diagrams;

— tools, codes, and scripts to transform an EXPRESS schema to SysML CXMI;

— transformation of EXPRESS interface specification;

— SysML constraints transformation from EXPRESS DERIVE attributes, except the ones used for renaming;

— SysML constraints transformation from EXPRESS SUPERTYPE expressions;

— SysML constraints transformation from EXPRESS GLOBAL rules;

— SysML constraints transformation from EXPRESS WHERE rules.

NOTE EXPRESS DERIVE attributes, EXPRESS SUPERTYPE expressions, EXPRESS GLOBAL rules and EXPRESS WHERE rules are transformed into OCL[2] specifications. OCL specifications are integrated in the SysML model based on a separate process.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 10303-11:2004, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual*

ISO/IEC 19505-1:2012, *Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 1: Infrastructure*

ISO/IEC 19509:2014, *Information technology — Object Management Group XML Metadata Interchange (XMI)*

ISO/IEC 19514:2017, *Information technology — Object management group systems modeling language (OMG SysML)*

3 Terms, definitions and abbreviated terms

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1 Terms related to generic concepts

3.1.1
EXPRESS
data specification language that consists of language elements that allow an unambiguous data definition and specification of constraints on the data defined

Note 1 to entry: The elements of the EXPRESS language are specified in ISO 10303-11.

3.1.2
data
representation of information in a formal manner suitable for communication, interpretation, or processing by human beings or computers

[SOURCE: ISO 10303-1:2021, 3.1.29]

3.1.3
data model
description of the organization of data in the management information system of an enterprise

[SOURCE: ISO/IEC 2382:2015, 2121422, modified — Notes 1 and 2 to entry were removed.]

3.1.4
information
facts, concepts, or instructions

[SOURCE: ISO 10303-1:2021, 3.1.41]

3.1.5
resulting CXMI
CXMI based on the transformation specification

3.1.6
information model
conceptual model of product data

Note 1 to entry: In ISO 10303, an information model is based on the Object-relationship modeling technique that organizes the product data as represented in different system aspects.

Note 2 to entry: In ISO 10303 information models may be developed using EXPRESS modeling language.

EXAMPLE Application Resource Model for ISO 10303-242 Managed model-based 3D engineering.

[SOURCE: ISO 10303-1:2021, 3.1.42 modified to detail the example]

3.2 Terms related to EXPRESS constructs

3.2.1
data type
<EXPRESS> domain of values

[SOURCE: ISO 10303-11:2004, 3.3.5]

3.2.2
defined data type
domain of values declared by the user with an identifier

3.2.3
entity
class of information defined by common properties

[SOURCE: ISO 10303-11:2004, 3.3.6]

3.2.4
entity data type
representation of an entity

[SOURCE: ISO 10303-11:2004, 3.3.7, modified — The second sentence was removed.]

3.2.5
entity (data type) instance
value of a class of information defined by common properties

3.2.6
enumeration data type
data type that has as its domain a set of names

3.2.7
aggregation data type
data type that has as its domain collections of values of a given base data type, e.g. array, list, bag, and set

3.2.8
instance
named value

[SOURCE: ISO 10303-11:2004, 3.3.10]

3.2.9
named data type
data type that may be declared in a formal specification, e.g. *entity data type* (3.2.4) and *defined data type* (3.2.2)

3.2.10
interface specification
construct which enables an item declared in one schema to be visible in another. e.g. USE and REFERENCE

3.2.11
select data type
data type that enables a choice among several named data types

3.2.12
simple data type
data type that defines the domains of the atomic data units, e.g. number, real, integer, string, Boolean, logical, and binary

3.2.13
value
unit of data

[SOURCE: ISO 10303-11:2004, 3.3.22]

3.2.14
global rule
restriction or implicit constraint that applies to one or more entities within the scope of a schema

3.2.15
local rule
assertion on the domain of every *entity instance* (3.2.5)

3.2.16
uniqueness rule
constraint assertions on the domain of an entity attribute or a combination of entity attributes for which their *instances* (3.2.8) must not be equal to every instance of the *entity* (3.2.3)

3.2.17
where rule
restriction constraint assertions on the domain of one or multiple entities and of one or a combination of these entities' attributes for every *instance* (3.2.8) of the *entity* (3.2.3)

3.3 Terms related to SysML constructs

3.3.1
Canonical XMI
specific constrained format of XMI that minimizes variability and provides more predictable identification and ordering

Note 1 to entry: A Canonical XMI file is itself a valid XMI file.

Note 2 to entry: The full definition is provided in ISO/IEC 19509:2014, Annex B.

3.3.2
association
classification of a set of tuples representing links between typed model elements

Note 1 to entry: The full definition is provided in ISO/IEC 19505-1:2012,11.5.

3.3.3
auxiliary
stereotype (3.3.12) applied to an abstract *block* (3.3.4) that has no properties

Note 1 to entry: The full definition is provided in ISO/IEC 19505-1:2012, Clause 22.

3.3.4
block
modular construct used for defining an *entity* (3.2.3)

Note 1 to entry: Used for defining Application activity model concepts, Application Data Planning objects, Application Domain Model Business Objects, Core model objects and ARM in SysML Entities. They may include reference properties, part properties, value properties, and constraints. They may be specializations of other blocks.

Note 2 to entry: The full definition is provided in ISO/IEC 19514:2017, Clause 8.

3.3.5
composite aggregation
responsibility for the existence of composed object

Note 1 to entry: If a composite object is deleted, all of its part *instances* (3.2.8) that are objects are deleted with it.

Note 2 to entry: The full definition is provided in ISO/IEC 19505-1:2012, 11.4.1.

3.3.6
directed association
association (3.3.2) between a collection of source model elements and a collection of target model elements that is said to be directed from the source elements to the target elements

Note 1 to entry: The full definition is provided in ISO/IEC 19505-1:2012, 7.2.3.3.

3.3.7
enumeration
value type (3.3.16) whose values are enumerated

Note 1 to entry: The full definition is provided in ISO/IEC 19505-1:2012, 10.2.3.3.

3.3.8
enumeration literal
named value for an *enumeration* (3.3.7)

Note 1 to entry: The full definition is provided in ISO/IEC 19505-1:2012, 10.2.3.3.

3.3.9
data type
<SysML> type whose *instances* (3.2.8) are identified only by their value

Note 1 to entry: The full definition is provided in ISO/IEC 19505-1:2012, 10.2.3.1.

3.3.10
generalization
directed relationship between a more general supertype and a more specific subtype

Note 1 to entry: Each Generalization relates a specific Classifier to a more general Classifier. Given a Classifier, the transitive closure of its general Classifiers is often called its generalizations, and the transitive closure of its specific Classifiers is called its specializations. The immediate generalizations are also called the Classifier's subtype, and where the Classifier is a Class, its supertype.

Note 2 to entry: The full definition is provided in ISO/IEC 19505-1:2012, C.1.1.

3.3.11
primitive type
predefined data type without any substructure

Note 1 to entry: The full definition is provided in ISO/IEC 19505-1:2012, Clause 21.

3.3.12
stereotype
limited kind of metaclass that cannot be used by itself, but which must always be used in conjunction with one of the metaclasses it extends

Note 1 to entry: The full definition is provided in ISO/IEC 19505-1:2012, 8.3.2.3 paragraph 6.

3.3.13
part property
property that specifies a part with strong ownership and coincidental lifetime of its containing *block* (3.3.4)

Note 1 to entry: It describes a local usage or a role of the typing Block in the context of the containing Block. Every Part Property has *composite aggregation* (3.3.5) and is typed by a Block.

Note 2 to entry: The full definition is provided in ISO/IEC 19514:2017, 8.3.2.3 paragraph 6.

3.3.14
reference property
property that specifies a reference of its containing *block* (3.3.4) to another block

Note 1 to entry: the full definition is provided in ISO/IEC 19514:2017, 8.3.2.3 paragraph 6.

3.3.15
value property
property of a *block* (3.3.4) that is typed with a *value type* (3.3.16) *stereotype* (3.3.12)

Note 1 to entry: The full definition is provided in ISO/IEC 19514:2017, 8.3.2.3, paragraph 6.

3.3.16
value type
stereotype (3.3.12) of UML Data Type that is used to define types of values that may be used to express information but cannot be identified as the target of any reference

Note 1 to entry: The full definition is provided in OMG Systems Modeling Language [6], 8.3.2.14.

3.4 Abbreviated terms

CXMI	Canonical XMI
ID	IDentifier
OCL	Object Constraint Language
OMG	Object Management Group
STEP	STandard for the Exchange of Product model data
SysML	Systems Modeling Language
UML	Unified Modeling Language
UUID	Universal Unique Identifier
XMI	XML Meta-data Interchange
XML	eXtensible Markup Language

4 EXPRESS to SysML CXMI

4.1 General

This clause describes the concepts and rules for the transformation mapping from an EXPRESS schema to a SysML model, which is physically stored in a CXMI file.

This document shall use the language construct specifications defined in ISO 10303-11, ISO/IEC 19505-1, ISO/IEC 19514 and ISO/IEC 19509.

NOTE The mapping clauses are sequentially structured to build increasingly complex transformation definitions.

This document shall be unambiguously identified in an open information system by the code defined in Annex A.

EXPRESS constructs and the equivalent SysML constructs are presented in a summarized table in Annex B.

4.2 Presentation conventions

For ease of identification, the fragments of EXPRESS, CXMI, and SysML are presented in separate boxes.

EXAMPLE 1 EXPRESS, CXMI, and SysML representations presented in separate boxes.

```
EXPRESS:
…
```

```
CXMI:
…
```

```
SysML:
…
```

The items significant to support the explanations are formatted using text effects to aid identification of the equivalent items in the EXPRESS, CXMI, and SysML fragments. When there is more than one significant item, different text effects are used for the different items. The following text effects are used:

— **Bold**;

— *Underline*;

— ***Mixed effects***.

Triple dots ("…") are used to hide content not relevant to a fragment. Curly brackets "{xxx}" are used to contain descriptive words of the content in the resulting CXMI.

EXAMPLE 2 Usage of **Bold** and *Underline* text effects to ease the identification of the significant items in EXPRESS, CXMI, and SysML fragments, and the usage of triple dots "…" and curly brackets "{umlid}".

```
EXPRESS:

ENTITY StepEntityName;
…
```

```
CXMI:

<packagedElement xmi:type='uml:Class' xmi:id='{umlid}' xmi:uuid='...'>
```

```
   <name>StepEntityName</name>
   …
```

```
SysML:
Class <<Block>> named StepEntityName
```

4.3 Common mapping conventions

4.3.1 Reference to external files

All the references in the SysML CXMI fragments are given as `xmi:idref` which assumes that the referenced element is contained in the same CXMI file. When the referenced element is in a different CXMI file the `href` is used instead. This will be the case for all reference to primitives and may be case for other references.

`type href` shall specify a relative reference to element:

```
CXMI:
<ownedAttribute xmi:id="{...}" xmi:uuid="{...}" xmi:type="uml:Property">
   <name>Text</name>
   <type href="../../../DataTypes.xmi#STRING"/>
   ...other tags...
</ownedAttribute>
```

`general href` shall specify relative reference to element in another CXMI file:

```
CXMI:
<packagedElement xmi:id="{...}" xmi:uuid="{...}" xmi:type="uml:Class">
  <name>DateTimeAssignment</name>
  <generalization xmi:id="{...}" xmi:uuid="{...}" xmi:type="uml:Generalization">
     <general href="../../Core_model/RequirementManagement/RequirementManagement.xmi#_18_
4_1_8e001ed_1504250730055_679435_26318"/>
  </generalization>
   ...other tags...
</packagedElement>
```

4.3.2 Treatment of Xmi:id, xmi:uuid, and UUID

A CXMI file uses xmi:id value to make references between all kinds of element. An xmi:id may be in an xmi:idref attribute.

Xmi:uuid (UUID[3]), is not relevant to be included in the mapping transformations. After the first mapping clause, this attribute will be omitted.

4.3.3 Assumed sysml:Block in fragments

For all the fragments that refer to Block, the following shows how a block is defined in the Canonical XMI. This is not repeated in the remaining examples, where only *xmi:type="uml:Class"* is included and the *sysml:Block* is assumed:

```
SysML:
```

```
Class <<Block>>
```

```
CXMI:
<sysml:Block xmi:id="..." xmi:uuid="...">
    <base_Class xmi:idref="{umlid}"/>
</sysml:Block>

<packagedElement xmi:type='uml:Class' xmi:id='{umlid}' xmi:uuid='...'>
   <name>StepEntityName</name>
   ...
</packagedElement>
```

4.3.4 Containment and reference relationships

The EXPRESS language (ISO 10303-11) does not distinguish between reference relationships and containment relationships. SysML and XSD support both types of relationship.

A reference relationship is a directed association attribute between two entities. A referenced entity by an attribute is an entity that may be referenced multiple times, and which exists standalone. Base root object entities are always referenced, never contained.

A containment relationship is a directed aggregation attribute between two entities. A contained entity is an entity that is owned by the entity that defines the containment relationship attribute. The contained entity does not exist if the containing entity does not exist. A simple type is necessarily contained.

4.3.5 Used stereotypes to represent EXPRESS concepts

Two existing UML stereotypes are used to represent specific STEP concepts:

<<Auxiliary>> stereotypes represent select data objects. Select data objects are represented as abstract Blocks in SysML.

<<Type>> stereotypes represent two specific types of Blocks:

— blocks that represent list of lists;

— block that represents Value Type in order to be able to include them as member in selects.

4.3.6 Select type and supertype

In STEP concepts, select types are not defined as entities but as types and are therefore not defined as supertypes of an entity. In SysML an entity identifies the supertype entities and select data types using the generalization relationship. For this document the term supertype excludes any select data types.

4.4 Mapping of a schema

The schema declaration shall be transformed to a `SysML package` that includes the STEP data model.

```
EXPRESS:


SCHEMA STEP_AP242_Domain_model;
           ...
```

```
CXMI:

<uml:Package xmi:type="uml:Package" xmi:id="...">
   <name>STEP_AP242_Domain_model</name>
       <packagedElement ...
```

```
SysML:
Package that includes directly the STEP data model represented in SysML model
```

4.5 Mapping of entities

4.5.1 General mapping of an entity

Each EXPRESS entity shall be transformed to a SysML block with the same name.

```
EXPRESS:

  ENTITY StepEntityName;
    ...
```

```
CXMI:

<packagedElement xmi:type='uml:Class' xmi:id='{umlid}' xmi:uuid='...'>
   <name>StepEntityName</name>
   ...
```

```
SysML:
Class <<Block>>
```

4.5.2 Mapping of abstract supertype

An abstract EXPRESS entity shall be transformed to a SysML abstract block.

```
EXPRESS:

ENTITY StepEntityName
 ABSTRACT SUPERTYPE;
...
```

```
CXMI:
<packagedElement xmi:type='uml:Class' xmi:id='{umlid}' >
   <name>StepEntityName</name>
    <isAbstract>true</isAbstract>
...
```

```
SysML:
Class <<Block>> with abstract parameter set as true
```

4.5.3 Mapping of entity with one supertype

An entity with one supertype shall be transformed to a SysML block that is a specialization of the supertype block.

```
EXPRESS:

ENTITY SubtypeEntity
    SUBTYPE OF (NameOfSupertypeEntity);
      ...
```

```
CXMI:
<packagedElement xmi:type='uml:Class' xmi:id='{umlid}'>
   <name>SubtypeEntity</name>
        <generalization xmi:id="…" xmi:uuid="..." xmi:type="uml:Generalization">
                  <general xmi:idref="{xmi:id of the Supertype}"/>
        </generalization>
```

```
SysML:
Class <<Block>> with general parameter including the name of the supertype block. The
supertyping is, formally, represented as a generalization relationship from the subtype
block to the supertype block.
```

4.5.4 Mapping of entity with multiple supertypes

An entity with multiple supertypes shall be transformed to a SysML block having multiple inheritances, which means that this block is the specialization of more than one block that is not an <<Auxiliary>> block. The following rules are applied, where **EntityX** has multiple inheritance of EntityA and EntityB: an EXPRESS entity with multiple supertypes shall be transformed to a SysML block that is a specialization of each supertype.

```
EXPRESS:

ENTITY EntityX
    SUBTYPE OF (EntityA, EntityB);
      ...
```

```
CXMI:
<packagedElement xmi:type='uml:Class' xmi:id='{umlid}' xmi:uuid='...'>
   <name> EntityX </name>
        <generalization xmi:id="…" xmi:uuid="..." xmi:type="uml:Generalization">
                  <general xmi:idref="="{xmi:id of the EntityA }"/>
        </generalization>
        <generalization xmi:id="…" xmi:uuid="..." xmi:type="uml:Generalization">
                  <general xmi:idref="="{xmi:id of the EntityB }"/>
        </generalization>
```

```
...
<packagedElement xmi:type='uml:Class' xmi:id='{umlid}' xmi:uuid='...'>
   <name> EntityA </name>
…

<packagedElement xmi:type='uml:Class' xmi:id='{umlid}' xmi:uuid='...'>
   <name> EntityB </name>
...
```

```
SysML:
Class <<Block>> with general parameter including the names of the supertype blocks. The
supertyping is, formally, represented as a generalization relationship from the subtype
block to the multiple supertype blocks.
```

4.5.5 Constraints between subtypes

The transformation is not defined in this document.

4.5.6 Mapping of entity attribute

4.5.6.1 General

Attributes of an entity are represented by SysML Properties. There are three types of such properties: Value Property, Part Property, and Reference Property:

— a value property is defined as any property whose target type is stereotypes as a <<ValueType>>. These are generated for EXPRESS attributes that are all simple types, enumeration types and select types of named simple types. Value properties are contained in the block when instantiated;

— a part property is defined as any property whose target type is stereotyped as a <<Block>> and that has an aggregation of 'composite'. These are generated for EXPRESS attributes that are of type Entity with a mandatory inverse attribute to the source or for nested aggregations that result in a <<Type>><<Block>> being generated. Part properties are contained in the block when instantiated;

— a reference property is defined as any property whose target type is stereotyped as a <<Block>> and that has an aggregation of 'none'. These are generated for EXPRESS attributes of type Entity that do not have a mandatory inverse attribute to the source or Select types of Entities. Reference properties are not contained in the block when instantiated.

For each EXPRESS explicit attribute of an EXPRESS entity, the corresponding SysML property of a SysML block declaration shall be defined.

The CXMI fragment below presents the generic target declaration of properties in a block. Key elements mentioned above are highlighted in bold.

```
CXMI:
<packagedElement ... xmi:type="uml:Class">
        <name>Entity...</name>

        <ownedAttribute xmi:id="..." xmi:uuid="..." xmi:type="uml:Property">
          <name>NameOfTheAttribute</name>
          <type xmi:idref="{xmi id of the type of the attribute}"/>
          <association xmi:idref="..."/>
        </ownedAttribute>
```

```
<ownedAttribute xmi:id="..." xmi:uuid="..." xmi:type="uml:Property">
  <name>NameOfAnotherAttribute</name>
  <aggregation>composite</aggregation>
  <type xmi:idref="..."/>
  <association xmi:idref="..."/>
  <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger"/>
  <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
    <value>*</value>
  </upperValue>
</ownedAttribute>
...
```

When the following CXMI fragment is declared in the ownedAttribute, it means the property is a Part Property. When it is omitted, it means the property is a Reference Property. In EXPRESS there is no distinction made between a Part and a Reference Property (see 4.3.4).

```
CXMI:
<aggregation>composite</aggregation>
```

4.5.6.2 Mapping of aggregate and optional attributes

SysML property multiplicity is the SysML parameter that permits to define aggregate attributes by setting the multiplicity different from 1.

In CXMI, <lowerValue> and <upperValue> defines the multiplicity and the optionality:

— when <lowerValue> is not declared, it means the attribute is mandatory (its value is equal to 1);

— when <lowerValue> is declared, without embedding a <value>, it means the attribute is optional (value = 0);

— when <upperValue> is not declared, it means its value is equal to 1;

— <value> provides the multiplicity.

The following fragments specify how EXPRESS attribute multiplicity and optionality are transformed to SysML Block properties.

An EXPRESS attribute with no optionality nor multiplicity declared shall be transformed to a mandatory SysML Property with the multiplicity equal to 1:

```
EXPRESS:
ENTITY …;
    NameOfTheAttribute: TypeOfTheAttribute;
```

```
CXMI:
lowerValue and upperValue are not set
```

```
SysML:
Property multiplicity is [1]
```

An optional EXPRESS attribute with no multiplicity declared shall be transformed to an SysML Property with the multiplicity equal to 0 to 1:

```
EXPRESS:
ENTITY …;
    NameOfTheAttribute: OPTIONAL TypeOfTheAttribute;
```

```
CXMI:
<lowerValue> set but <value> not set
<upperValue> is not set (as the default behaviour is 1).
```

```
SysML:
Property multiplicity is [0..1]
```

An optional EXPRESS attribute with aggregate with minimum value of 1 declared shall be transformed to a mandatory SysML Property with the multiplicity equal to 1:

```
EXPRESS:
ENTITY …;
    NameOfTheAttribute: OPTIONAL SET[1:?] OF TypeOfTheAttribute;
```

```
CXMI:
<lowerValue> set but <value> not set
<upperValue> set and <value> set as *
```

```
SysML:
Property multiplicity is [0..*]
```

A mandatory EXPRESS attribute with aggregate with declared multiplicity starting at 1 shall be transformed to a mandatory SysML Property with the multiplicity equal to 1 to many:

```
EXPRESS:
  ENTITY …;
    NameOfTheAttribute: SET[1:?] OF TypeOfTheAttribute;
```

```
CXMI:
<lowerValue> set and <value> set as 1
<upperValue> set and <value> set as *
```

```
SysML:
Property multiplicity is [1..*]
```

A mandatory EXPRESS attribute with aggregate with multiplicity declared from n to m, assuming n and m are positive integer, shall be transformed to a mandatory SysML Property with the multiplicity equal to n to m:

```
EXPRESS:
ENTITY …;
    NameOfTheAttribute: SET[n:m] OF TypeOfTheAttribute;
```

```
CXMI:
<lowerValue> set and <value> set as n
<upperValue> set and <value> set as m
```

```
SysML:
Property multiplicity is [n..m]
```

4.5.6.3 Attribute typed as simple type

Simple type attribute shall be transformed to SysML value properties. This clause provides the mapping of one of simple type attribute: **STRING**. The mapping of other simple type attributes for the other ones. The exhaustive list of simple types is presented in 4.7.1.

```
EXPRESS:
ENTITY Entity1;
    Attribute1: STRING;
```

```
CXMI:

<packagedElement ... xmi:type="uml:Class">
  <name>Entity1</name>
  <ownedAttribute xmi:id="..."  xmi:type="uml:Property">
          <name>Attribute1</name>
          <type xmi:idref="STRING"/>
          <association xmi:idref="{uml id of Association}"/>
       ...

<packagedElement xmi:id="STRING" xmi:uuid="..." xmi:type="uml:PrimitiveType">
    <name>String</name>
</packagedElement>
```

```
SysML:
<<Value Property>> typed PrimitiveType STRING <<ValueType>>
```

4.5.6.4 Attribute typed as an entity

The EXPRESS element corresponding to a SysML property whose data type is a block (not Auxiliary or Type) shall be mapped in one of following ways.

— In the default case when no mandatory inverse is present, the EXPRESS attribute shall be transformed to a SysML Reference Property:

```
EXPRESS:
ENTITY Entity1;
    Attribute1: Entity2;
```

```
CXMI:
<packagedElement ... xmi:type="uml:Class">
  <name>Entity1</name>
  <ownedAttribute xmi:id="..."  xmi:type="uml:Property">
          <name>Attribute1</name>
          <type xmi:idref="{uml id of Entity2}"/>
          <association xmi:idref="{uml id of Association}"/>
       ...
<packagedElement ... xmi:type="uml:Class">
  <name>Entity2</name>
...
<packagedElement xmi:id="..."  xmi:type="uml:Association">
  <memberEnd xmi:idref="{uml id of Attribute1}"/>
  <memberEnd xmi:idref="{uml id of OwnedEnd}"/>
  <ownedEnd xmi:id="..." xmi:type="uml:Property">
    <aggregation>composite</aggregation>
    <type xmi:idref="{uml id of Entity1}"/>
    <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger"/>
    <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
      <value>*</value>
    </upperValue>
  </ownedEnd>
```

```
SysML:
<<Reference Property>> typed as Entity2 <<Block>>
```

— In the case a mandatory inverse declaration of the attribute is present, this EXPRESS attribute shall be transformed to a SysML Part Property:

```
EXPRESS:
ENTITY Entity1;
  Attribute1: EntityWithInverse;
END_ENTITY;

ENTITY EntityWithInverse;
INVERSE
 E1 : Entity1 FOR Attribute1;
```

```
END_ENTITY;
```

```
CXMI:
<packagedElement ... xmi:type="uml:Class">
  <name>Entity1</name>
  <ownedAttribute xmi:id="..."  xmi:type="uml:Property">
          <name>Attribute1</name>
          <type xmi:idref="{uml id of EntityWithInverse}"/>
          <association xmi:idref="{uml id of Association}"/>
        ...
<packagedElement ... xmi:type="uml:Class">
  <name>EntityWithInverse</name>
  <ownedAttribute xmi:id="..."  xmi:type="uml:Property">
          <name>E1</name>
          <isReadOnly>true</isReadOnly>
          <type xmi:idref="{uml id of Entity2}"/>
          <association xmi:idref="{uml id of Association}"/>
...
<packagedElement xmi:id="..."  xmi:type="uml:Association">
  <memberEnd xmi:idref="{uml id of Attribute1}"/>
  <memberEnd xmi:idref="{uml id of E1}"/>
```

```
SysML:
<<Part Property>> typed as EntityWithInverse <<Block>>
```

4.5.6.5 Attribute typed as select

In SysML, a Select type is a SysML Abstract Auxiliary block. An EXPRESS select type attribute shall be transformed to a SysML Part or Reference Property that is the select type.

```
EXPRESS:
ENTITY Entity1;
    Attribute1: name_of_the_typing_Select_express_type;
```

```
CXMI:
<packagedElement ... xmi:type="uml:Class">
  <name>Entity1</name>
  <ownedAttribute xmi:id="..."  xmi:type="uml:Property">
          <name>Attribute1</name>
          <type xmi:idref="{uml id of the typing Select Block}"/>
          <association xmi:idref="..."/>
```

```
SysML:
<<Reference or Part Property>> typed as a select Class <<Block>> <<Auxiliary>>
```

4.5.6.6 Attribute typed as enumeration type

An EXPRESS enumeration type attribute shall be mapped to a SysML Enumeration Value Property.

```
EXPRESS:
TYPE NameOfTheEnumeration = ENUMERATION OF(an enumeration string, another enumeration
string);
  END_TYPE;


 ENTITY Entity1;
   Attribute1: NameOfTheEnumeration;
```

```
CXMI:
<packagedElement ... xmi:type="uml:Class">
  <name>Entity1</name>
  <ownedAttribute xmi:id="..."  xmi:type="uml:Property">
          <name>Attribute1</name>
          <type xmi:idref="{uml id of the typing Enumeration Block}"/>
          <association xmi:idref="..."/>


 <packagedElement xmi:id="="{uml id of the typing Enumeration Block "
xmi:type="uml:Enumeration">
        <name>NameOfTheEnumeration</name>
        <ownedLiteral xmi:id="..." xmi:type="uml:EnumerationLiteral">
          <name>an enumeration string</name>
        </ownedLiteral>
        <ownedLiteral xmi:id="..."  xmi:type="uml:EnumerationLiteral">
          <name>another enumeration string</name>
...
```

```
SysML:
<<Value Property>> typed as an Enumeration <<Value Property>> with EnumerationLiterals,
which enumerates the strings
```

4.5.6.7 Derived attributes

The transformation is not defined in this document unless the derivation is purely to define a renamed attribute.

Renaming derivations are identified by an EXPRESS expression that just contains a simple factor of the form "`SELF\Entity.attribute`" with no operators being present. If such a derived attribute is found, then a corresponding property named as the derived attribute shall be in the SysML with an appropriate Redefined Property attribute set to the original property.

4.5.6.8 Inverse attributes

The EXPRESS inverse attributes shall be transformed a `SysML readOnly` property without stereotype. The association between the two blocks will not contain an `ownedEnd` since both properties are owned by their respective blocks. In the case where the EXPRESS inverse is constrained to be a single mandatory attribute, then the explicit attribute shall be transformed to a part property in SysML (see 4.5.6.4).

4.5.7 Local rules

4.5.7.1 Where rules

The transformation is not defined in this document.

4.5.7.2 Uniqueness rules

EXPRESS Uniqueness rules shall be transformed into corresponding OCL population constraints.

```
EXPRESS:
ENTITY EntityX
    attr: STRING;
UNIQUE
    UR1: attr;
      ...
```

```
CXMI:
<ownedRule xmi:type="uml:Constraint" xmi:id="..." xmi:uuid="..." name="UR1">
  <constrainedElement xmi:idref=" {uml id of EntityX}"/>
  <specification xmi:type="uml:OpaqueExpression" xmi:id="..." xmi:uuid="...">
      <body>EntityX::allInstances()->isUnique(attr)</body>
      <language>OCL2.0</language>
…
```

```
SysML:
SysML Constraint declared in the Class <<Block>> specifying an OCL uniqueness population constraint of the a property.
```

If a joint uniqueness rule consisting of more than one attribute is encountered, then the corresponding OCL `isUnique` clause shall contain an OCL Sequence of the attributes.

EXAMPLE Joint uniqueness rule consisting of two attributes is encountered then the corresponding OCL `isUnique` clause contains an OCL Sequence of the attributes:

```
CXMI:
isUnique(Sequence {attrib1, attrib2})
```

4.6 Mapping of global rules

The transformations of EXPRESS SUPERTYPE expressions and EXPRESS GLOBAL are not defined in this document.

EXAMPLE EXPRESS DERIVE attributes, EXPRESS SUPERTYPE expressions, EXPRESS GLOBAL rules and EXPRESS WHER rules may be integrated in the resulting CXMI of the SysML model based on the transformation from EXPRESS rules to OCL constraints, by using a separate process based on a look-up table.

4.7 Mapping of type

4.7.1 Mapping of simple type

In order to follow the semantics of EXPRESS and to enable EXPRESS from and to SysML mappings, it is necessary to use STEP own simple types, also named primitive types.

EXAMPLE 1 NUMBER is a generalization of REAL, that is a generalization of INTEGER. The UML and SysML types do not have this relationship between the primitive types.

EXPRESS STRING shall be transformed to SysML string value type:

```
EXPRESS:
STRING
```

```
CXMI:
    <packagedElement xmi:id="STRING" xmi:uuid="..." xmi:type="uml:PrimitiveType">
      <name>String</name>
    </packagedElement>
```

```
SysML:
    PrimitiveType STRING <<ValueType>>
```

EXPRESS NUMBER shall be transformed to SysML number value type:

```
EXPRESS:
    NUMBER
```

```
CXMI:
    <packagedElement xmi:id="NUMBER" xmi:uuid="..." xmi:type="uml:PrimitiveType">
      <name>Number</name>
      <isAbstract>true</isAbstract>
    </packagedElement>
```

```
SysML:
   PrimitiveType NUMBER <<ValueType>>
```

EXPRESS REAL shall be transformed to SysML real value type:

```
EXPRESS:
    REAL
```

```
CXMI:
    <packagedElement xmi:id="REAL" xmi:uuid="..." xmi:type="uml:PrimitiveType">
      <name>Real</name>
```

```
      <generalization xmi:id="_generalization-REAL_NUMBER" xmi:uuid="..."
xmi:type="uml:Generalization">
        <general xmi:idref="NUMBER"/>
      </generalization>
    </packagedElement>
```

```
SysML:
    PrimitiveType REAL <<ValueType>>
```

EXPRESS INTEGER shall be transformed to SysML integer value type:

```
EXPRESS:
   INTEGER
```

```
CXMI:
    <packagedElement xmi:id="INTEGER" xmi:uuid="..." xmi:type="uml:PrimitiveType">
      <name>Integer</name>
      <generalization xmi:id="_generalization-INTEGER_REAL" xmi:uuid="..."
xmi:type="uml:Generalization">
        <general xmi:idref="REAL"/>
      </generalization>
    </packagedElement>
```

```
SysML:
     PrimitiveType INTEGER <<ValueType>>
```

EXPRESS LOGICAL shall be transformed to SysML logical value type:

```
EXPRESS:
    LOGICAL
```

```
CXMI:
    <packagedElement xmi:id="LOGICAL" xmi:uuid="...." xmi:type="uml:Enumeration">
      <name>Logical</name>
      <ownedLiteral xmi:id="UNKNOWN" xmi:uuid="..." xmi:type="uml:EnumerationLiteral">
        <name>Unknown</name>
      </ownedLiteral>
    </packagedElement>
```

```
SysML:
     PrimitiveType LOGICAL <<ValueType>>
```

EXPRESS BOOLEAN shall be transformed to SysML Boolean value type:

```
EXPRESS:
  BOOLEAN
```

```
CXMI:
<packagedElement xmi:id="BOOLEAN" xmi:uuid="..." xmi:type="uml:Enumeration">
      <name>Boolean</name>
      <generalization xmi:id="_generalization-BOOLEAN-LOGICAL" xmi:uuid="..."
xmi:type="uml:Generalization">
        <general xmi:idref="LOGICAL"/>
      </generalization>
      <ownedLiteral xmi:id="TRUE" xmi:uuid="..." xmi:type="uml:EnumerationLiteral">
        <name>True</name>
      </ownedLiteral>
      <ownedLiteral xmi:id="FALSE" xmi:uuid="..." xmi:type="uml:EnumerationLiteral">
        <name>False</name>
      </ownedLiteral>
    </packagedElement>
  </uml:Package>
```

```
SysML:
     PrimitiveType BOOLEAN from specialized LOGICAL <<ValueType>>
```

For each of the defined `uml:Enumeration` and `uml:PrimitiveType`, a corresponding `sysml:ValueType` is defined.

EXAMPLE 2 Declaration of the `BOOLEAN` SysML value type:

```
CXMI:
  <sysml:ValueType xmi:id="BOOLEAN_VT" xmi:uuid="...">
    <base_DataType xmi:idref="BOOLEAN"/>
  </sysml:ValueType>
```

Binary simple type is not mapped as it is not used.

4.7.2 Mapping of aggregation types

4.7.2.1 General

There are four types of aggregations defined by STEP EXPRESS concepts:

— Bag;

— Set;

— List;

— Array.

SysML CXMI supports `Bag`, `Set` and `List`. `Array` is not fully supported by the mapping specifications in this version of this document. Consequently, the indexing concept of an `Array` is not mapped, and an EXPRESS `Array` attribute shall be transformed to a SysML list of unique property.

In CXMI, the combination of isOrdered and isUnique parameters are used to define four types of aggregations:

— Not ordered + not unique = `Bag`;

— Not ordered + is unique = `Set`;

— Is ordered + is unique = `List` of unique;

— Is ordered + not unique = `List`.

`isOrdered` is default **`false`** if omitted and `isUnique` is default **`true`** if omitted:

```
CXMI:
<isOrdered>true</isOrdered>
<isUnique>false</isUnique>
```

The fully subclauses specify the mapping of one-dimension aggregation types. Higher dimension aggregation types are documented in 4.7.3.

Multiplicity mappings for EXPRESS attributes are specified in 4.5.6.2.

4.7.2.2 Set mapping

An EXPRESS Set attribute shall be transformed to an SysML block property with `parameters isOrdered` set as **`false`** and `isUnique` set as **`true`**.

```
EXPRESS:
ENTITY NameSet;
    members: SET[2:?] OF NameOfEntity;
       …
```

```
CXMI:
<packagedElement xmi:id="..." xmi:type="uml:Class">
        <name>NameSet</name>
        <ownedAttribute xmi:id="..." xmi:type="uml:Property">
          <name>members</name>
      <isOrdered>false</isOrdered>
      <isUnique>true</isUnique>
          <type xmi:idref="{umlid of the NameOfEntity}"/>
          <lowerValue xmi:id="..." xmi:ty”e="uml:LiteralInteger">
            <value>2</value>
          </lowerValue>
          <upperValue xmi:id="..." xmi:ty”e="uml:LiteralUnlimitedNatural">
            <value>*</value>
          </upperValue>
        </ownedAttribute>
      </packagedElement>
```

```
SysML:
Set property of a block.
```

4.7.2.3 List mapping

An EXPRESS `List` attribute shall be transformed to an SysML block property with parameters `isOrdered` set as **`true`** and `isUnique` set as **`false`**.

```
EXPRESS:
 ENTITY NameList;
    members: LIST[2:?] OF NameOfEntity;
```

```
CXMI:
<packagedElement xmi:id"...” xmi:type="uml:Class">
        <name>NameList</name>
        <ownedAttribute xmi:id="... " xmi:type="uml:Property">
          <name>members</name>
           <isOrdered>true</isOrdered>
            <isUnique>false</isUnique>
          <type xmi:idref="{umlid of the NameofEntity}"/>
          <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger">
             <value>2</value>
          </lowerValue>
          <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
             <value>*</value>
          </upperValue>
        </ownedAttribute>
      </packagedElement>
```

```
SysML:
List property of a block.
```

4.7.2.4 Bag mapping

An EXPRESS `Bag` attribute shall be transformed to an SysML block property with parameters `isOrdered` set as **`false`** and `isUnique` set as **`false`**.

```
EXPRESS:
ENTITY NameList;
    members: LIST[2:?] OF NameOfEntity;
```

```
CXMI:
<packagedElement xmi:id"...” xmi:type="uml:Class">
        <name>NameList</name>
        <ownedAttribute xmi:id="... " xmi:type="uml:Property">
          <name>members</name>
            <isOrdered>false</isOrdered>
            <isUnique>false</isUnique>
          <type xmi:idref="{umlid of the NameOfEntity}"/>
          <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger">
            <value>2</value>
          </lowerValue>
```

```
          <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
            <value>*</value>
          </upperValue>
        </ownedAttribute>
      </packagedElement>
```

```
SysML:
Bag property of a block.
```

4.7.2.5 Array mapping

An `Array` is an ordered fixed size aggregation, so the lower and upper multiplicities shall correspond to the number of elements declared in the `Array`. The Indexing of an Array is currently not supported as indicated in 4.7.2.1.

An EXPRESS `Array` attribute shall be transformed to an SysML block property with parameters `isOrdered` set as **`true`** and `isUnique` set as **`true`**.

```
EXPRESS:
  ENTITY ArrayOfEntity;
    members: ARRAY[1:3] OF NameOfEntity;
       …
```

```
CXMI:
<packagedElement xmi:id"…" xmi:type="uml:Class">
        <name> ArrayOfEntity </name>
        <ownedAttribute xmi:id="..." xmi:type="uml:Property">
          <name>members</name>
           <isOrdered>true</isOrdered>
           <isUnique>true</isUnique>
         <type xmi:idref="{umlid of the Entity 'NameOfEntity'}"/>
          <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger">
            <value>3</value>
          </lowerValue>
          <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
            <value>3</value>
          </upperValue>
        </ownedAttribute>
      </packagedElement>
```

```
SysML:
List of unique property of a block.
```

4.7.3 Mapping of nested of aggregation types

4.7.3.1 General

There are several type classes in EXPRESS:

— named types (declared by `TYPE` or `ENTITY` keyword);

— anonymous Types declared in-line when declaring attributes.

As an introduction to this clause, the transformation keeps any 1-dimensional anonymous types as anonymous aggregations. Any anonymous types with aggregation greater than 1 are converted into anonymous aggregation for the first dimension and creates SysML blocks for the further aggregation dimensions (handling one level at a time). Any named types that are aggregations of just 1 dimension, are lost in the mapping (they become anonymous aggregations). Named types that are higher level aggregations are converted into SysML blocks and used accordingly.

Four nested aggregation type mappings are specified in this clause: `list` of `list`, `array` of `array`, `list` of `array` and `array` of `list`. Other types of nested aggregation are not used.

4.7.3.2 List of list mapping

Below is the definition of a `list` of `list` of `REAL` attribute and its mapping. A `list` of `list` of a simple type is represented by an intermediary SysML block, with an "`element`" <<Value Property>>, which is the naming convention for elements of a constructed aggregation. The name of the type is produced as follows: name of the aggregation + lower multiplicity + upper multiplicity + name of domain. In the following extracts, the second aggregate defined to be a `LIST [2:3] OF REAL` is called `List23Real`. This maintains any multiplicity constraints on the target SysML model. The new block `List23Real` is stereotyped as <<Type>> to indicate that it is a convenience object and should not be considered as a conceptual artefact of the model.

```
EXPRESS:
  ENTITY NameOfTheListOfListEntity;
 members : LIST[2:3] OF LIST[2:3] OF REAL;
```

```
CXMI:
<packagedElement xmi:id="..." xmi:type="uml:Class">
        <name>NameOfTheListOfListEntity</name>
        <ownedAttribute xmi:id="..." xmi:type="uml:Property">
          <name>members</name>
          <isOrdered>true</isOrdered>
          <isUnique>false</isUnique>
          <type xmi:idref="{umlid of type 'List23Real'}"/>
          <association xmi:idref="..."/>
          <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger">
            <value>2</value>
          </lowerValue>
          <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
            <value>3</value>
          </upperValue>
        </ownedAttribute>

<packagedElement xmi:id="..." xmi:type="uml:Class">
        <name>List23Real</name>
        <ownedAttribute xmi:id="..." xmi:type="uml:Property">
          <name>elements</name>
          <isOrdered>true</isOrdered>
          <isUnique>false</isUnique>
          <type xmi:idref="{ ../../../DataTypes.xmi#REAL }"/>
          <aggregation>composite</aggregation>
          <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger">
```

```
            <value>2</value>
          </lowerValue>
          <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
            <value>3</value>
          </upperValue>
        </ownedAttribute>
```

```
SysML:
  Class <<Block>> <<Type>> List of REAL with a <<Value Property>> attribute named
elements
```

4.7.3.3 Array of array mapping

Below is the definition of an `Array` of `Array` of an entity arbitrary named `EntityX` and its mapping. An `Array` of `Array` of an Entity is represented by an intermediary SysML block, with an "`element`" <<Reference Property>>, which is the naming convention for elements of a constructed aggregation. The name of the type is produced as follows: name of the aggregation + lower multiplicity + upper multiplicity + name of domain. In the following extracts, the second aggregate defined to be an `ARRAY [2:3] OF EntityX` is called `Array23EntityX`. This maintains any multiplicity constraints on the target SysML model. The new block `Array23EntityX` is stereotyped as <<Type>> to indicate that it is a convenience object and should not be considered as a conceptual artefact of the model.

```
EXPRESS:
  ENTITY NameOfTheArraysofArrayEntity;
 members : ARRAY[2:3] OF ARRAY[2:3] OF EntityX;
```

```
CXMI:
<packagedElement xmi:id="..." xmi:type="uml:Class">
        <name>NameOfTheArrayOfArrayEntity</name>
        <ownedAttribute xmi:id="..." xmi:type="uml:Property">
          <name>members</name>
          <isOrdered>true</isOrdered>
          <isUnique>false</isUnique>
          <type xmi:idref=" {umlid of type 'Array23EntityX'} "/>
          <association xmi:idref="..."/>
          <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger">
            <value>2</value>
          </lowerValue>
          <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
            <value>2</value>
          </upperValue>
        </ownedAttribute>

<packagedElement xmi:id="..." xmi:type="uml:Class">
        <name>Array23EntityX</name>
        <ownedAttribute xmi:id="..." xmi:type="uml:Property">
          <name>elements</name>
          <isOrdered>true</isOrdered>
          <isUnique>false</isUnique>
          <type xmi:idref=" { xmi id of the EntityX } "/>
```

```
          <association xmi:idref="..."/>
          <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger">
            <value>2</value>
          </lowerValue>
          <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
            <value>2</value>
          </upperValue>
        </ownedAttribute>
```

```
SysML:
  Class <<Block>> <<Type>> Array of a <<Reference Property>> attribute named elements
```

4.7.3.4 List of array mapping

Below is the definition of a `List of Array` of an entity arbitrary named `EntityX` and its mapping. The same conventions and transformation rules of 4.7.3.2 and 4.7.3.3 apply in this clause.

```
EXPRESS:
  ENTITY NameOfTheListOfListEntity;
 members : LIST[2:3] OF ARRAY[2:3] OF EntityX;
```

```
CXMI:
<packagedElement xmi:id="..." xmi:type="uml:Class">
        <name>NameOfTheListOfListEntity</name>
        <ownedAttribute xmi:id="..." xmi:type="uml:Property">
          <name>members</name>
          <isOrdered>true</isOrdered>
          <isUnique>false</isUnique>
          <type xmi:idref="{umlid of type 'List23Real'}"/>
          <association xmi:idref="..."/>
          <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger">
            <value>2</value>
          </lowerValue>
          <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
            <value>3</value>
          </upperValue>
        </ownedAttribute>

<packagedElement xmi:id="..." xmi:type="uml:Class">
        <name>List23Real</name>
        <ownedAttribute xmi:id="..." xmi:type="uml:Property">
          <name>elements</name>
          <isOrdered>true</isOrdered>
          <isUnique>false</isUnique>
          <type xmi:idref="{ ../../../DataTypes.xmi#REAL }"/>
          <aggregation>composite</aggregation>
          <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger">
            <value>2</value>
          </lowerValue>
```

```
          <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
            <value>3</value>
          </upperValue>
        </ownedAttribute>
```

```
SysML:
  Class <<Block>> <<Type>> List of a <<Reference Property>> attribute named elements
```

4.7.3.5 Array of list mapping

Below is the definition of an `Array of List` of REAL attribute and its mapping. The same conventions and transformation rules of 4.7.3.2 and 4.7.3.3 apply in this clause.

```
EXPRESS:
  ENTITY NameOfTheArraysofArrayEntity;
  members : ARRAY[2:3] OF LIST[2:3] OF REAL;
```

```
CXMI:
<packagedElement xmi:id="..." xmi:type="uml:Class">
        <name>NameOfTheArrayOfArrayEntity</name>
        <ownedAttribute xmi:id="..." xmi:type="uml:Property">
          <name>members</name>
          <isOrdered>true</isOrdered>
          <isUnique>false</isUnique>
          <type xmi:idref=" {umlid of type 'Array23EntityX'} "/>
          <association xmi:idref="..."/>
          <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger">
            <value>2</value>
          </lowerValue>
          <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
            <value>2</value>
          </upperValue>
        </ownedAttribute>

<packagedElement xmi:id="..." xmi:type="uml:Class">
        <name>Array23EntityX</name>
        <ownedAttribute xmi:id="..." xmi:type="uml:Property">
          <name>elements</name>
          <isOrdered>true</isOrdered>
          <isUnique>false</isUnique>
          <type xmi:idref=" { xmi id of the EntityX } "/>
          <association xmi:idref="..."/>
          <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger">
            <value>2</value>
          </lowerValue>
          <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
            <value>2</value>
          </upperValue>
        </ownedAttribute>
```

```
SysML:
  Class <<Block>> <<Type>> Array of REAL with a <<Value Property>> attribute named
elements
```

4.7.4 Mapping of select type

4.7.4.1 General

In STEP concepts, select types are not defined as entities but as types and are therefore not defined as supertypes of an entity. In SysML, an entity identifies the supertype entities and select data types using the generalization relationship. For this document the term supertype excludes any select data types.

4.7.4.2 Mapping of simple select type of entities

An EXPRESS select type shall be transformed to a SysML abstract auxiliary block. Its select members shall be specializations of this block.

```
EXPRESS:
  TYPE NameOfTheSelect = SELECT(name of the member, name of another member);
  END_TYPE;
```

```
CXMI:
<packagedElement xmi:id="{xmi id of the Select}" xmi:type="uml:Class">
        <name>NameOfTheSelect</name>
        <isAbstract>true</isAbstract>
   ...
<packagedElement xmi:id="{xmi id of the member}" xmi:type="uml:Class">
        <name>{name of the member}</name>
        <generalization xmi:id="..." xmi:type="uml:Generalization">
          <general xmi:idref="{xmi id of the select}"/>
        </generalization>   ...
<packagedElement xmi:id="{xmi id of another member}" xmi:type="uml:Class">
        <name>{name of another member}</name>
        <generalization xmi:id="..."  xmi:type="uml:Generalization">
          <general xmi:idref="{xmi id of the select}"/>
        </generalization>   ...
<StandardProfile:Auxiliary  xmi:id="...">
    <base_Class  xmi:idref="{xmi id of the select}"/>
</StandardProfile:Auxiliary>
```

```
SysML:
Class <<Block>> <<Auxiliary>> Abstract
```

4.7.4.3 Mapping of nested select type

An EXPRESS select type shall be transformed to a SysML abstract auxiliary block. Its select members shall be specializations of this block. A select type may contain other select types.

```
EXPRESS:
```

```
  TYPE NameOfTheSelect = SELECT(NameOfTheANOTHERSelect, …);
  END_TYPE;
```

```
CXMI:
<packagedElement xmi:id="{xmi id of the Select}" xmi:type="uml:Class">
        <name>NameOfTheSelect</name>
        <isAbstract>true</isAbstract>
   ...
<packagedElement xmi:id="{xmi id of another select}" xmi:type="uml:Class">
        <name>{NameOfTheANOTHERSelect}</name>
        <isAbstract>true</isAbstract>
         <generalization xmi:id="..."  xmi:type="uml:Generalization">
          <general xmi:idref="{xmi id of the select}"/>
        </generalization>
```

4.7.4.4 Mapping of constrained select types

The domain of an EXPRESS Select Type may be restricted when a specialization of the Type is referenced in an EXPRESS entity declaration that is a subtype of the EXPRESS entity declaration that specified the original TYPE. The transformation of these structures into corresponding SysML is defined in such a way as to enable SysML tools to accurately perform Type compatibility checks without having to interpret OCL constraints. The following is the transformation to be applied:

a) Create SysML abstract blocks for each of the Select types in the source EXPRESS;

b) Create generalization relationships between these blocks to represent the specializations;

c) For each leaf node in the generalization structure, consider the allowed population from the root select type and remove all those constrained by where rules in the leaf node. Assign these as specializations of the leaf node;

d) For each non-leaf node consider the allowed population of the root select type, remove from this considered population any types already populated in the specializations of this node and by where rules defined in this node. Assign any remaining types as specializations of this node.

In the following extracts, the original EXPRESS declaration is considered as follows:

```
EXPRESS:
TYPE attachment_method = SELECT (nail, glue, weld, needle, tape); END_TYPE;
TYPE permanent_attachment = attachment_method;
WHERE
wr1: NOT ('LONGFORM.NEEDLE' IN TYPEOF(SELF);
wr2: NOT ('LONGFORM.TAPE' IN TYPEOF(SELF);
END_TYPE;
TYPE simple_attachment = attachment_method;
WHERE
wr1: NOT ('LONGFORM.GLUE' IN TYPEOF(SELF);
wr2: NOT ('LONGFORM.WELD' IN TYPEOF(SELF);
END_TYPE;
```

The resulting CXMI shall then be:

```
CXMI:
```

```
<packagedElement xmi:id="..." xmi:type="uml:Class">
        <name>attachment_method</name>
        <isAbstract>true</isAbstract>
...
<packagedElement xmi:id="..." xmi:type="uml:Class">
        <name>permanent_attachment</name>
        <isAbstract>true</isAbstract>
         <generalization xmi:id="..."  xmi:type="uml:Generalization">
          <general xmi:idref="{xmi id of attachment_method}"/>
         </generalization>
...
<packagedElement xmi:id="..." xmi:type="uml:Class">
        <name>simple_attachment</name>
        <isAbstract>true</isAbstract>
         <generalization xmi:id="..."  xmi:type="uml:Generalization">
          <general xmi:idref="{xmi id of attachment_method}"/>
         </generalization>
...
<packagedElement xmi:id="{xmi id of another select}" xmi:type="uml:Class">
        <name>nail</name>
        <isAbstract>true</isAbstract>
         <generalization xmi:id="..."  xmi:type="uml:Generalization">
          <general xmi:idref="{xmi id of simple_attachment}"/>
          <general xmi:idref="{xmi id of permanent_attachment}"/>
         </generalization>
...
<packagedElement xmi:id="{xmi id of another select}" xmi:type="uml:Class">
        <name>needle</name>
        <isAbstract>true</isAbstract>
         <generalization xmi:id="..."  xmi:type="uml:Generalization">
          <general xmi:idref="{xmi id of simple_attachment}"/>
         </generalization>
...
<packagedElement xmi:id="{xmi id of another select}" xmi:type="uml:Class">
        <name>tape</name>
        <isAbstract>true</isAbstract>
         <generalization xmi:id="..."  xmi:type="uml:Generalization">
          <general xmi:idref="{xmi id of simple_attachment}"/>
         </generalization>
...
<packagedElement xmi:id="{xmi id of another select}" xmi:type="uml:Class">
        <name>glue</name>
        <isAbstract>true</isAbstract>
         <generalization xmi:id="..."  xmi:type="uml:Generalization">
          <general xmi:idref="{xmi id of permanent_attachment}"/>
         </generalization>
...
<packagedElement xmi:id="{xmi id of another select}" xmi:type="uml:Class">
        <name>weld</name>
        <isAbstract>true</isAbstract>
         <generalization xmi:id="..."  xmi:type="uml:Generalization">
```

```
          <general xmi:idref="{xmi id of permanent_attachment}"/>
        </generalization>
```

4.7.5 Proxy artefact

In SysML, it is not possible to make a Value Type, such as a string, a subtype of a block. The concept of Proxy is introduced in order to allow Value Types as members of Selects. Proxies are Blocks stereotyped as <<Type>> and its members are Value properties named as "value" per the naming convention. Therefore, a proxy is a model artefact that allows a generic data type to be used as a class object for a select type. As it is a modelling artefact, it is not represented in EXPRESS. In EXPRESS the Value Type is directly represented in the select list.

EXAMPLE The ClassStringProxy is used to represent a class as a string. The name of the corresponding Value Type is ClassString.

```
EXPRESS:
  TYPE ClassSelect = SELECT(
   Class,
   ClassString,
   ExternalOwlClass
    );
  END_TYPE;

  TYPE ClassString = STRING;
  END_TYPE;
```

```
 CXMI:
 <packagedElement xmi:id="..." xmi:type="uml:Class">
        <name>ClassStringProxy</name>
        <generalization xmi:id="..." xmi:type="uml:Generalization">
          <general xmi:idref="{xmi id of a select block}"/>
        </generalization>
...
```

```
SysML:
Class <<Block>> <<Type>> Proxy of ClassStringProxy typed as String <<Value Type>>
```

4.7.6 Mapping of enumeration type

4.7.6.1 General

Two kinds of enumeration types are distinguished: simple enumeration and constrained enumeration.

4.7.6.2 Simple enumeration type mapping

An EXPRESS enumeration type shall be transformed to a SysML enumeration. EXPRESS enumeration string members shall be transformed to the `EnumerationLiterals` strings of the SysML enumeration.

```
EXPRESS:
  TYPE NameOfTheEnumeration = ENUMERATION OF(an enumeration string, another enumeration
string);
```

```
  END_TYPE;
```

```
CXMI:
  <packagedElement xmi:id="..."  xmi:type="uml:Enumeration">
        <name>NameOfTheEnumeration</name>
        <ownedLiteral xmi:id="..." xmi:type="uml:EnumerationLiteral">
          <name>an enumeration string</name>
        </ownedLiteral>
        <ownedLiteral xmi:id="..."  xmi:type="uml:EnumerationLiteral">
          <name>another enumeration string</name>
...
```

```
SysML:
Enumeration <<Value Type>> with EnumerationLiterals
```

4.7.6.3 Mapping of constrained enumeration types

EXPRESS type structure allows for an Enumeration Type to be specialized to remove particular elements.

The transformation of these structures into corresponding SysML is defined in such a way as to enable SysML tools to accurately perform date type compatibility checks without having to interpret OCL constraints.

a) Create SysML enumerations for each of the Enumeration types in the source EXPRESS;

b) Create generalization relationships between these enumerations to represent the specializations;

c) For each leaf node in the generalization structure, consider the allowed population from the root enumeration type and remove all those constrained by where rules in the leaf node. Assign these as literals of the leaf node;

d) For each non-leaf node consider the allowed population of the root enumeration type, remove from this considered population any types already populated in the specializations of this node and by where rules defined in this node. Assign any remaining types as specializations of this node.

In the following extracts, the original EXPRESS declaration is considered as follows:

```
EXPRESS:
TYPE colour = ENUMERATION OF (red, yellow, green, white); END_TYPE;

TYPE stop_light = colour;
WHERE
  wr1: SELF <> white;
END_TYPE;

TYPE canadian_flag = colour;
WHERE
  wr1: SELF <> yellow;
  wr2: SELF <> green;
END_TYPE;
```

The resulting CXMI shall then be:

```
CXMI:
  <packagedElement xmi:id="..."  xmi:type="uml:Enumeration">
        <name>colour</name>
  </packagedsElement>
  <packagedElement xmi:id="..."  xmi:type="uml:Enumeration">
        <name>stop_light</name>
        <generalization xmi:id="..."  xmi:type="uml:Generalization">
          <general xmi:idref="{xmi id of colour}"/>
        </generalization>
        <ownedLiteral xmi:id="..." xmi:type="uml:EnumerationLiteral">
          <name>red</name>
        </ownedLiteral>
        <ownedLiteral xmi:id="..."  xmi:type="uml:EnumerationLiteral">
          <name>yellow</name>
        </ownedLiteral>
        <ownedLiteral xmi:id="..."  xmi:type="uml:EnumerationLiteral">
          <name>green</name>
        </ownedLiteral>
  </packagedsElement>
  <packagedElement xmi:id="..."  xmi:type="uml:Enumeration">
        <name>canadian_flag</name>
        <generalization xmi:id="..."  xmi:type="uml:Generalization">
          <general xmi:idref="{xmi id of colour}"/>
        </generalization>
        <ownedLiteral xmi:id="..." xmi:type="uml:EnumerationLiteral">
          <name>red</name>
        </ownedLiteral>
        <ownedLiteral xmi:id="..."  xmi:type="uml:EnumerationLiteral">
          <name>white</name>
        </ownedLiteral>
  </packagedsElement>
```

```
SysML:
Enumeration <<Value Type>> with the subset of the initial EnumerationLiterals list
according to the constrained population declaration
```