TECHNICAL SPECIFICATION

ISO/TS 10303-16

First edition
2021-07

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

Part 16: Description methods: SysML XMI to EXPRESS transformation

Reference number
ISO/TS 10303-16:2021(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

ISO 10303 is an International Standard for the computer-interpretable representation and 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.

10303-16 is a member of the description methods series. This document specifies a mapping of SysML XMI to the EXPRESS language.

The STEP APs Domain models and the STEP Core Model are information model layer components that are part of the STEP Extended Architecture. These components are developed using SysML. In the past STEP information models were developed using the EXPRESS language. For legacy reasons it is therefore required to provide an EXPRESS schema derived from the SysML domain model. It is mandatory that the transformation from the SysML model to the EXPRESS schema is defined by a reference standard and guarantee a repeatable process applicable to future application protocols developments.

The Object Management Group (OMG) has standardized the XML Metadata Interchange specification (XMI) that integrates the OMG Systems Modeling Language (SysML), the OMG Unified Modeling Language (UML), the OMG Meta-Object Facility (MOF) and the World Wide Web Consortium (W3C) Extensible Markup Language (XML) standards. SysML inherits the XMI interchange capability from UML. XMI is a mechanism for the interchange of metadata between UML-based modeling tools and MOF-based metadata repositories. OMG has also standardized an XMI compliant interchange format for the SysML thus specifying a lexical representation of SysML models based on a standardized metamodel of the SysML. That lexical representation includes, among other things, the ability to interchange data type information, class information (or entities), groupings of classes providing namespaces for the classes (or schemas), associations between classes and inheritance between classes (or subtypes).

ISO has standardized the EXPRESS language (ISO 10303-11:2004). It is used to specify information requirements in ISO 10303.

10303-16 specifies a description method of the STEP Parts family, which defines the transformation of SysML constructs to the EXPRESS elements. Because the XMI standard specifies the XML representation of SysML metamodel constructs, standardizing the binding of SysML constructs into EXPRESS elements supports the representation of SysML models as EXPRESS schemas. SysML metamodel concepts that appear in SysML Block Diagrams are mapped into data specifications defined by EXPRESS schemas. This document does not map all SysML metamodel constructs to EXPRESS elements, because 10303 SysML models do not use all SysML metamodel constructs.

The EXPRESS schemas are derived from the domain model by applying the implementation bindings on the SysML XMI. The EXPRESS binding is realized with XSL transformations, which transforms the SysML model into an EXPRESS Schema. The specified binding is a one-way transformation from SysML information model represented in XMI into an EXPRESS schema. Due to this limitation 10303-16 does not define the transformation of arbitrary SysML models to EXPRESS.

Readers of 10303-16 require detailed knowledge of the EXPRESS language, and SysML.

The structure, conventions and concepts of the EXPRESS language are defined in ISO 10303-11:2004.

The main component of this standard is:

— the specification of the transformation from SysML XMI to EXPRESS for each STEP element modelled in SysML.

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

Part 16: Description methods: SysML XMI to EXPRESS transformation

1 Scope

This document specifies a mapping of SysML (ISO/IEC 19514:2017) constructs to EXPRESS (ISO 10303-11:2004) elements for the purpose of representing SysML model represented in XMI (ISO/IEC 19509:2014) as EXPRESS (ISO 10303-11:2004) schemas. The specified mapping is a one-way transformation from SysML information model represented in XMI into an EXPRESS schema.

NOTE Due to this limitation 10303-16 does not define the transformation of arbitrary SysML models to EXPRESS.

The following are within the scope of this document:

— the transformation of SysML metamodel constructs represented in XMI to EXPRESS elements for the purpose of representing SysML information models as EXPRESS schemas.

The following are outside the scope of this document:

— the transformation of SysML metamodel constructs into EXPRESS elements that are not used in the STEP Extended Architecture.

NOTE The STEP Extended Architecture is defined in References [8], [9] and [10].

— the transformation of SysML metamodel constructs into EXPRESS elements for other purposes than representing SysML constructs as STEP concepts;

— codes and scripts to transform SysML XMI to EXPRESS schema;

— the transformation of SysML constraints (OCL[5]) into EXPRESS global and local rules;

— the transformation of EXPRESS elements into SysML constructs.

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 10303-1, *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles*

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)*

W3C Recommendation: Extensible Markup Language (XML) 1.0 (Fifth Edition)

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 10303-11, ISO/IEC 19505-1, ISO/IEC 19514 and the following apply.

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

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

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

3.1.1 Terms and definitions for generic concepts

3.1.1.1
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]

3.1.1.2
data model
description of the organization of data in the management information system of an enterprise

[SOURCE: ISO/IEC 2382:2015, 2121422]

3.1.1.3
EXPRESS
language by which aspects of product data can be defined

[SOURCE: ISO/TS 10303-15:2021, 3.1.1.1]

3.1.1.4
information
facts, concepts, or instructions

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

3.1.1.5
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 — In the definition, "formal" has been replaced with "conceptual"; in Note 2 to entry, "are" has been replaced with "may be"; the Example has been changed.]

3.1.2 Terms and definitions for SysML constructs

3.1.2.1
association
association classifies 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.

3.1.2.2
auxiliary
stereotype applied to an abstract block that has no properties

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

3.1.2.3
block
modular construct used for defining an entity.

Note 1 to entry: Application activity model concepts, application data planning objects, application domain model business objects, core model objects and ARM in SysML Entities. They can include reference, part, value properties, and constraints. They can be specializations of other blocks.

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

3.1.2.4
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.

3.1.2.5
composite aggregation
responsibility for the existence of the composed object

Note 1 to entry: If a composite object is deleted, all of its part instances 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.

3.1.2.6
connector
link between two or more instances playing owned or inherited roles within a StructuredClassifier

Note 1 to entry: The full definition is provided in ISO/IEC 19505-1:2012, 11.2.3.3.

3.1.2.7
enumeration
value type whose values are enumerated in the model as enumeration literals

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

3.1.2.8
enumeration literal
user defined data value for an enumeration

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

3.1.2.9
data type
type whose instances 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.

3.1.2.10
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.

3.1.2.11
part property
property that specifies a part with strong ownership and coincidental lifetime of its containing block

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 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.

3.1.2.12
primitive type
definition of a predefined data type, without any substructure

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

3.1.2.13
reference property
property that specifies a reference of its containing block to another block

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

3.1.2.14
stereotype
limited kind of metaclass that cannot be used by itself but 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, 12.3.3.4.

3.1.2.15
value property
property of a block that is typed with a value type

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

3.1.2.16
value type
a stereotype 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 ISO/IEC 19514:2017, 8.3.2.14.

3.2 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 SysML XMI to EXPRESS

4.1 General

This clause describes the concepts and rules for the transformation mapping from a STEP SysML model stored as a CXMI file to an EXPRESS Schema.

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

4.2 Presentation conventions

For ease of identification, separate boxes are used for the fragments of SysML, CXMI and EXPRESS.

The items significant to the section are usually in **bold**, but more than text effects may be used where needed to support the explanation. Triple dots ("...") are used to hide content not relevant to an extract.

Curly brackets "{xxx}" are used to contain descriptive words of the content in the resulting CXMI.

4.3 Common mapping conventions

4.3.1 Reference to external files

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

Canonical XMI: *type href* relative reference to element in DataTypes.xmi

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

Canonical XMI: *general href* relative reference to element in another XMI 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 xmi:id, xmi:uuid, and UUID

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

Xmi:uuid (UUID[4]), 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 fragments, 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 relationship

The EXPRESS language does not distinguish between reference relationships and containment relationships. SysML supports both types of relationships. Therefore, in 4.11, no distinction is made whether an association between two entities is realized by reference or by relationship.

4.3.5 Used stereotypes to represent STEP 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 represents 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 not treated as SysML 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 types using the generalization relationship. For this document supertype excludes any select types.

4.4 Mapping of the primary schema

```
SysML:
Package that includes directly the STEP data model represented in SysML and intended to
be transformed and implemented
```

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

```
EXPRESS:
SCHEMA STEP_AP242_Domain_model;
   …
```

4.5 Mapping of Entities

4.5.1 General mapping of Entity

For each SysML block declaration (that is not an abstract *<<auxiliary>>*), the EXPRESS Schema shall contain the definition of a new *entity data type* corresponding to that SysML block.

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

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

```
EXPRESS:
ENTITY StepEntityName;
   …
```

4.5.2 Mapping of abstract entity

A SysML abstract block shall be transformed to an EXPRESS abstract supertype.

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

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

```
EXPRESS:
ENTITY StepEntityName
   ABSTRACT SUPERTYPE;
   ...
```

4.5.3 Mapping of entity with one supertype

A SysML subtype Block shall be mapped to an EXPRESS subtype.

```
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.
```

```
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>
   …
```

```
EXPRESS:
ENTITY SubtypeEntity
   SUBTYPE OF (NameOfSupertypeEntity);
   …

If the SupertypeEntity has only one subtype:
ENTITY NameOfSupertypeEntity
   SUPERTYPE OF (SubtypeEntity);
   …

If the SupertypeEntity has more than one subtype:
ENTITY NameOfSupertypeEntity
   SUBTYPE OF (ONE OF(SubtypeEntity, AnotherSubtypeEntity, …));
   …
```

NOTE SysML GeneralizationSet is not included in the Extended Architecture, therefore a SYSML block cannot be the generalization of more than one other block at the same time. This is mapped to EXPRESS by a ONEOF constraint.

4.5.4 Mapping of entity with multiple supertypes

Multiple inheritance is where the SysML block has more than one supertype that is not an *<<Auxiliary>>*. This kind of SysML block shall be mapped to an EXPRESS subtype naming all its supertypes. The following rules are applied:

— the below fragment builds on the above fragment with the additional features:

  — EntityX has multiple inheritance of ***IdentifiableObject*** EntityA and ***ProgrammeObject EntityB***;

  — ***EntityB*** has a supertype ***EntityC***.

```
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.
```



```
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> EntityB </name>
   <generalization xmi:id="…" xmi:uuid="..." xmi:type="uml:Generalization">
      <general xmi:idref="="{xmi:id of the EntityC }"/>
</generalization>
   …
```

```
EXPRESS:
ENTITY EntityX
   SUBTYPE OF (EntityA, EntityB);
   …
ENTITY A
   SUPERTYPE OF (EntityX);
   …
ENTITY B
   SUPERTYPE OF (EntityX)
   SUBTYPE OF (EntityC);
   …
ENTITY C
   SUPERTYPE OF (EntityB);
   …
```

4.6 Mapping of simple type

In order to follow the semantics of EXPRESS (to enable EXPRESS from/to SysML mappings), it is necessary to use STEP own simple types (also named primitive types). For example, Number is a generalization of Real, that is a generalization of Integer. The UML/SysML types do not have this relationship between the primitive types. The fragments below provide the mapping requirements.

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

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

```
EXPRESS:
STRING
```

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

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

```
EXPRESS:
NUMBER
```

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

```
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>
```

```
EXPRESS:
REAL
```

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

```
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>
```

```
EXPRESS:
INTEGER
```

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

```
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>
```

```
EXPRESS:
LOGICAL
```

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

```
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>
```

```
EXPRESS:
BOOLEAN
```

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

EXAMPLE

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

```
EXPRESS:
TYPE BOOLEAN_VT = BOOLEAN;
END_TYPE;
```

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

4.7 Mapping of aggregation type

There are four types of aggregation:

— Bag;

— Set;

— List;

— Array.

SysML CXMI supports all of those types. But in this document, all aggregations types are *Set* in the EXPRESS. *isOrdered* is default false if omitted and *isUnique is* default true if omitted:

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

The fragments below provide the mapping requirements. But consequently, they do not specify *isOrdered* nor *isUnique*.

```
SysML:
Class <<Block>> <<Type>> Set of an entity
```

```
CXMI:
<packagedElement xmi:id"..." xmi:type="uml:Class">
   <name>NameSet</name>
   <ownedAttribute xmi:id="... " xmi:type="uml:Property">
      <name>elements</name>
      <aggregation>composite</aggregation>
      <type xmi:idref="{umlid of the Entity}"/>
      <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger">
         <value>2</value>
      </lowerValue>
      <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
         <value>*</value>
      </upperValue>
   </ownedAttribute>
</packagedElement>
```

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

NOTE In CXMI, the combination of *isOrdered* (default false if omitted) and *isUnique* (default true if omitted) are used to define the four types of aggregations:

— Not ordered and not unique is mapped to Bag;

— Not ordered and unique is mapped to Set;

— Ordered and unique is mapped to List;

— Ordered and not unique is mapped to Array.

4.8 Mapping of aggregation of aggregation type

A list of lists of a simple type is represented by a block, with an “elements” property, with the naming convention for aggregation of aggregation.

Below is the definition of a List of List of Real.

```
SysML:
Class <<Block>> <<Type>> List of List of Real
```

```
CXMI:
<packagedElement xmi:id="..." xmi:type="uml:Class">
   <name>NameOfTheListOfListType</name>
   <ownedAttribute xmi:id="..." xmi:type="uml:Property">
      <name>elements</name>
      <isOrdered>true</isOrdered>
      <isUnique>false</isUnique>
      <type href="../../../DataTypes.xmi#REAL"/>
      <lowerValue xmi:id="..." xmi:type="uml:LiteralInteger">
         <value>2</value>
      </lowerValue>
      <upperValue xmi:id="..." xmi:type="uml:LiteralUnlimitedNatural">
         <value>3</value>
      </upperValue>
   </ownedAttribute>
   …
```

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

An array of array is represented by a block, with an “elements” property, which the naming convention for aggregation of aggregation.

Below is the definition of an Array of Array of an Entity. Hereafter the name of the type is ARRAYEntity.

```
SysML:
Class <<Block>> <<Type>> Array of Array of an Entity
```

```
CXMI:
<packagedElement xmi:id="..." xmi:type="uml:Class">
   <name>ARRAYEntity</name>
   <ownedAttribute xmi:id="..." xmi:type="uml:Property">
      <name>elements</name>
      <type xmi:idref="{xmi:id of the Entity}"/>
      <upperValue xmi:id="..." xmi:uuid="..." xmi:type="uml:LiteralUnlimitedNatural">
         <value>3</value>
      </upperValue>
   </ownedAttribute>
   …
```

```
EXPRESS:
ENTITY ARRAYEntity;
   elements : ARRAY[1:3] OF ARRAY[1:3] OF NumericalValue;
   …
```

4.9 Mapping of Select type

A SysML Auxiliary Abstract Block shall be transformed to an EXPRESS select type.

4.9.1 General mapping of select type

A select type in SysML is an abstract auxiliary block. Its members are subtypes of this block.

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

```
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>
   <generalization xmi:id="..." xmi:type="uml:Generalization">
      <general xmi:idref="{xmi id of another 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>
   <generalization xmi:id="..." xmi:type="uml:Generalization">
      <general xmi:idref="{xmi id of another select}"/>
   </generalization>
   …

<StandardProfile:Auxiliary xmi:id="..." xmi:uuid="…">
   <base_Class xmi:idref="{xmi id of the select}"/>
</StandardProfile:Auxiliary>
```

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

NOTE A select type can contain another select type.

EXAMPLE Select type containing another select type:

```
CXMI:
   <name>NameOfTheSelect</name>
   <isAbstract>true</isAbstract>
   …

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

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

4.9.2 Mapping of select type containing value type

In SysML, because Select types are Blocks and the members are subtypes, 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.

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

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

TYPE ClassString = STRING;
END_TYPE;
```

4.10 Mapping of enumeration type

A SysML Enumeration Value Type with EnumerationLiterals shall be transformed to an EXPRESS enumeration type.

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

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

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

4.11 Mapping of entity attribute

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 will be typed as a simple type or an enumeration type and is necessarily contained in the block when instantiated;

— a part property will be typed by a select or by a block and is necessarily contained in the block when instantiated;

— a reference property will be typed by a select or by a block and is not contained in the block when instantiated.

For each SysML explicit property of a SysML block declaration the corresponding entity type in the EXPRESS Schema Definition contains an **attribute** definition, with exceptions (see below). The following general rules are applied:

— the order of elements is fixed;

— the name of the EXPRESS attribute is the name of the property in the SysML model;

— for each inverse composite aggregation property of a SysML Block declaration, the associated EXPRESS entity contains an inverse attribute corresponding to the SysML Block property.

The CXMI below presents the generic declaration of properties in a block. Text in bold spots the key elements.

```
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 is declared in the ownedAttribute, it means it is a Part Property. When it is omitted, it means it is a Reference Property. In EXPRESS there is no distinction made between a Part and a Reference Property. Both are mapped as explicit attributes.

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

<lowerValue> and <upperValue> defines the multiplicity and the optionality:

— when <lowerValue> not is declared, it means the attribute is mandatory;

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

— <value> provides the multiplicity.

4.11.1 Mapping of multiplicity and optionality

The SysML Block properties multiplicity shall be transformed to EXPRESS as presented in the fragments below:


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

```
CXMI:
lowerValue and upperValue are not set (as the default behaviour is 1).
```

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

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

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

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

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

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

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

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

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

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

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

```
CXMI:
<lowerValue> set nad <value> seet as n (integer)
<upperValue> set and <value> set as *
```

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

4.11.2 Attribute typed as an Entity

The EXPRESS element corresponding to a SysML property whose data type is a block and stereotyped as Part Property (not Auxiliary or Type) shall be transformed to EXPRESS as presented in the fragments below:

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

```
CXMI:
   <ownedAttribute xmi:id="..." xmi:type="uml:Property">
      <name>NameOfTheAttribute</name>
      <aggregation>composite</aggregation>
      <type xmi:idref="{uml id of the typing Block}"/>
      <association xmi:idref="..."/>
      …
```

```
EXPRESS:
NameOfTheAttribut: name of the typing block;
…
```

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

```
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 Block}"/>
      <association xmi:idref="..."/>
      …
It should not contain:
<aggregation>composite</aggregation>
```

```
EXPRESS:
ENTITY Entity1;
   Attribute1: name of the typing block;
   …
```

4.11.3 Attribute typed as Select

The EXPRESS attribute corresponding to a SysML property whose data type is an Abstract Auxiliary block (Select type) data type shall be transformed to EXPRESS in the following way:

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

```
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="..."/>
      …
If this is a reference property, it should NOT contain:
<aggregation>composite</aggregation>
```

```
EXPRESS:
ENTITY Entity1;
   Attribute1: name of the typing Select Block;
   …
```

4.11.4 Attribute typed as Enumeration type

Properties of SysML blocks that are of enumeration types shall be transformed to EXPRESS with the name of an EXPRESS Enumeration type.

```
SysML:
<<Part Property>> typed as a Enumeration <<Value Type>> with EnumerationLiterals
```

```
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>
      …
```

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

ENTITY Entity1;
   Attribute1: NameOfTheEnumeration;
   …
```

Annex A
(normative)

Information object registration

To provide for unambiguous identification of an information object in an open system, the following object identifier is assigned to this document:

```
{iso standard 10303 part(16) version(1)}
```

The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

Annex B
(informative)

EXPRESS/Information modelling constructs and the equivalent SysML modelling constructs

Table B.1 below lists the information modelling concepts (that are common with EXPRESS information modelling language constructs) and the equivalent SysML constructs and canonical XMI constructs. It is ordered using the clause numbering from ISO 10303-11:2004.

NOTE 1 Greyed rows in Table B.1 are concept title with following rows describing the details of the concept. These reflect the clauses in ISO 10303-11:2004 which are listed in the first column.

NOTE 2 Table B.1 is based on Table 3 in N3424[10] with CXMI constructs added.

Table B.1 — EXPRESS/Information modelling constructs and the equivalent SysML modelling constructs with CXMI constructs

Clause in ISO 10303-11:2004	EXPRESS/Information modelling concepts	SysML Construct	CXMI constructs	Rules and comments
7	Basic language elements			NOTE ISO 10303-11:2014, Clause 7, details how the basic language elements are to be parsed according to the syntax rules defined in ISO 10303-11:2014, Annex A. Since SysML is not represented as a parsed language there is no need to identify correspondence between the two languages except for the concept of remarks. OCL is a parsed language so the OCL equivalent concepts are identified.
7.1	Character set			The character set used by EXPRESS and both SysML and OCL are equivalent.
7.1.6	Remarks	comment	`<ownedComment … xmi:type="uml:Comment">…</ownedComment>`	All SysML elements (Blocks, Properties etc.) that need to have textual descriptions should be added as comments within those elements. Developer problems and issues should be identified in the Packages (via the diagrams) and be suitably stereotypes so as not to be included in the publication mechanism.
7.2	Reserved words			There are no reserved words in SysML. There are reserved words OCL.
7.3	Symbols			OCL provides a list of symbols.
7.4	Identifiers	name		
7.5	Literals			OCL includes corresponding literals
8	Data types			SysML provides its own Data types, however, to use the EXPRESS inheritance structure the following data types have been declared in SysML in the package DataTypes which should be used instead of the built-in SysML data types.
8.1	Simple data types	ValueType		
8.1.1	Number data type	Number <<ValueType>>	`packagedElement xmi:type="uml:PrimitiveType"`	
8.1.2	Real data type	Real <<ValueType>>	`packagedElement xmi:type="uml:PrimitiveType"`	
8.1.3	Integer data type	Integer <<ValueType>>	`packagedElement xmi:type="uml:PrimitiveType"`	
8.1.4	Logical data type	Logical <<ValueType>>	`packagedElement xmi:type="uml:Enumeration"`	
8.1.5	Boolean data type	Boolean <<ValueType>>	`packagedElement xmi:type="uml:Enumeration"`	
8.1.6	String data type	String <<ValueType>>	`packagedElement xmi:type="uml:PrimitiveType"`	

Table B.1 *(continued)*

Clause in ISO 10303-11:2004	EXPRESS/Information modelling concepts	SysML Construct	CXMI constructs	Rules and comments
8.1.7	Binary data type	Binary <<ValueType>>	`packagedElement xmi:type="uml:PrimitiveType"`	
8.2	Aggregation data types			All aggregations types are defined to have a multiplicity of "a..b" where a = 0 (i.e. optional) or 1 and b is an integer. They may be specified to have unique members and may be ordered.

8.2.1	Array data type	Block stereotyped as <<Type>> with elements property	`<ownedAttribute ...>`
			`<isOrdered>true</isOrdered>`
			`</ownedAttribute>`
				At present, the indexing capability of Array has not been represented in SysML. This can be done but there has been no specified need for this level of complexity yet. The intent of the numbers in brackets in the EXPRESS specification is to identify the indexing mechanism used. This means that all Arrays are of fixed multiplicity, this is reflected into the resulting SysML.
				At present, an Array is deemed to be equivalent to a fixed size List so just uses.
				isOrdered='true'

8.2.2	List data type	Block stereotyped as <<Type>> with elements property	`<ownedAttribute ...>`
			`<isOrdered>true</isOrdered>`
			`</ownedAttribute>`
				isOrdered='true'

8.2.3	Bag data type	Block stereotyped as <<Type>> with elements property	`<ownedAttribute ...>`
			`<isUnique>false</isUnique>`
			`</ownedAttribute>`
				isOrdered='false'

8.2.4	Set data type	Block stereotyped as <<Type>> with elements property	`<ownedAttribute ...>`
			`...`
			`</ownedAttribute>`
				isOrdered='false'
				isUnique='true'

8.2.5	Value uniqueness on aggregates	OCL constraint	`<ownedRule ... xmi:type="uml:Constraint">`
			`<specification>`
			`<body>...</body>`
			`<language>OCL2.0</language>`
			`</specification>`
			`...`
			`</ownedRule>`
				SysML and OCL uniqueness (like EXPRESS) is based on instance comparison. Value uniqueness would have to be specified by a more complex OCL constraint.

8.3	Named data types			

Table B.1 *(continued)*

Clause in ISO 10303-11:2004	EXPRESS/Information modelling concepts	SysML Construct	CXMI constructs	Rules and comments
8.3.1	Entity data type	Block with part, reference or value Property	`<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>`	
8.3.2	Defined data type	ValueType	`<sysml:ValueType...>   <base_DataType xmi:idref="..."/> </sysml:ValueType>`	
8.4	Constructed data types			
8.4.1	Enumeration data type	Enumeration <<ValueType>> EnumerationLiteral Generalization	`<packagedElement ... xmi:type="uml:Enumeration">   <ownedLiteral ... xmi:type="uml:EnumerationLiteral">...</ownedLiteral> ... </packagedElement> <sysml:ValueType ...>   <base_DataType xmi:idref="..."/> </sysml:ValueType>`	The enumeration ValueType owns the EnumerationLiterals. Extension uses Generalization. This behaves differently from the EXPRESS extensible enumerations, in that the generalization subtype does not change the supertype. The equivalent final enumerations may be generated.

Table B.1 *(continued)*

Clause in ISO 10303-11:2004	EXPRESS/Information modelling concepts	SysML Construct	CXMI constructs	Rules and comments
E	Select data type	Block with stereotype <<Auxiliary>> isAbstract='true' Generalization	<packagedElement xmi:id="{xmi id of the Select}" xmi:type="uml:Class"> <isAbstract>true</isAbstract> ... </packagedElement> <StandardProfile:Auxiliary ...> <base_Class xmi:idref="{xmi id of the Select}"/> </StandardProfile:Auxiliary>	The <<Auxiliary>> stereotype is found in the UML L2 Extensions Generalization relationship from the select elements to the Auxiliary. This is the equivalent to the EXPRESS SELECT list. If all the elements of the select list are Entities, then a Block is used. If all the elements of the select list are defined data types, then a ValueType is used. If mixed types are used (entity and defined data types), this is an Auxiliary Block, with the data types represented as Blocks (called "<name>Proxy") stereotyped as <<Type>> (to indicate it is a supporting type) with a single value property of the ValueType. The auxiliary is used as the type for a property
8.5	Generalized data types	OCL isTypeOf	<ownedRule ... xmi:type="uml:Constraint"> <specification> <body>...oclIsTypeOf...</body> <language>OCL2.0</language> </specification> ... </ownedRule>	
8.6	Data type usage classification			This is explicit to EXPRESS parsing rules so not required.
9	Declarations			

Table B.1 *(continued)*

Clause in ISO 10303-11:2004	EXPRESS/Information modelling concepts	SysML Construct	CXMI constructs	Rules and comments
9.1	Type declaration	ValueType or Block	<sysml:ValueType...> <base_DataType xmi:idref="{umlid}"/> </sysml:ValueType> Or <sysml:Block ...> <base_Class xmi:idref="{umlid}"/> </sysml:Block> <packagedElement xmi:type='uml:Class' xmi:id='{umlid}' xmi:uuid='...'> <name>StepEntityName</name> ... </packagedElement>	For user-defined types that are specializations of simple types.
9.2	Entity declaration	Block	<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>	An entity and an application domain model block are commonly named domain object. An entity shall not be stereotyped as auxiliary. An entity may have isAbstract set to “true”