INTERNATIONAL STANDARD

ISO/IEC 11179-3

Fourth edition
2023-01

Information technology — Metadata registries (MDR) —

Part 3: Metamodel for registry common facilities

Reference number
ISO/IEC 11179-3:2023(E)

COPYRIGHT PROTECTED DOCUMENT

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

List of Figures

List of Tables

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

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 document 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 or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC 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) or the IEC list of patent declarations received (see https://patents.iec.ch).

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. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC/JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

This fourth edition cancels and replaces the third edition (ISO/IEC 11179-3:2013), which has been technically revised. It also incorporates the Amendment ISO/IEC 11179-3:2013/Amd.1:2020.

The main changes are as follows:

— this fourth edition presents a metamodel for the Common Facilities of a Basic Registry which has potential use for more than just metadata;

— the previous edition has been split into multiple parts to make it more manageable;

  — the Basic Attributes (formerly Clause 12) have been moved to ISO/IEC 11179-30: Basic attributes of metadata;

  — the Data Description region (formerly Clause 11) has been moved to ISO/IEC 11179-31: Metamodel for data specification registration;

  — the Concepts region (formerly Clause 9) and the Binary Relations region (formerly Clause 10) have been moved to ISO/IEC 11179-32: Metamodel for concept system registration;

— simplification of the UML used to describe the metamodels, such as:

  — elimination of use of stereotypes;

  — addition of an explicit 'Item' class as the superclass of all types of registry items;

  — removal of role names on associations;

  - removal of redundant specification of attributes and associations in the text;

- refactoring of some of the packages to reduce dependencies, including:

  - moving the Concept class to the Basic and Core package where it is referenced from multiple metamodel regions, including: the Data Specification package in ISO/IEC 11179-31, the Concept System package in ISO/IEC 11179-32, the Data Set package in ISO/IEC 11179-33 and the Model package in ISO/IEC 11179-35;

  - moving the Context class to the Basic and Core package where it is referenced from the Designation and Definition package in this document, the Data Specification package in ISO/IEC 11179-31 and the Data Set package in ISO/IEC 11179-33;

  - moving the Slot class to the Basic and Core package, a more appropriate location than the Identification package;

  - a Classification region has been restored, based on the style of ISO/IEC 11179-3:2003[12], to remove dependency on the Concept System region for classification;

- adding a generic mapping facility among registry items;

- a change to the formatting of the text in Clauses 5 through 11 and Annexes B and C, to better align with ISO Directives and ISO House Style, see 5.3.2.

A list of all parts in the ISO/IEC 11179 series can be found on the ISO and IEC websites.

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 and www.iec.ch/national-committees.

Introduction

Data processing and electronic data interchange rely heavily on accurate, reliable, controllable and verifiable data recorded in databases. A prerequisite for correct and proper use and interpretation of data is that both users and owners of data have a common understanding of the meaning and representation of the data. To facilitate this common understanding, a number of characteristics, or attributes, of the data have to be defined. These characteristics of data are known as "metadata", that is, "data that describes data".

The attributes of data elements and associated metadata can be specified and recorded as registry items in a metadata registry (MDR). The metadata registry is used to keep information about data elements and associated concepts, such as "data element concepts", "conceptual domains" and "value domains" (see ISO/IEC 11179-31). Generically, these are all referred to as "metadata items". Such metadata are necessary to clearly describe, record, analyse, classify and administer data.

ISO/IEC 11179 addresses the semantics of data, the representation of data and the registration of the descriptions of that data. It is through these descriptions that an accurate understanding of the semantics and a useful depiction of the data are found.

The purposes of the ISO/IEC 11179 series are to promote the following:

— standard description of data;

— common understanding of data across organizational elements and between organizations;

— re-use and standardization of data over time, space and applications;

— harmonization and standardization of data within an organization and across organizations;

— management of the components of descriptions of data;

— re-use of the components of descriptions of data.

Each part of ISO/IEC 11179 is devoted to addressing a different aspect of these needs, as described in ISO/IEC 11179-1:2023, Clause 7. This document specifies the information to be recorded in a metadata registry in the form of a conceptual data model. It also specifies common facilities for dealing with identification, designation, definition and registration of any type of registry item. Thus, this document applies to registries other than metadata registries. Other parts of ISO/IEC 11179 extend this model to support specific types of metadata items, such as: data elements, data element concepts, data set specifications, concept systems, etc. (See 1.2.)

NOTE ISO/IEC 11179-30[16] describes the basic attributes of registry items for purposes where a complete metadata registry is not appropriate.

This document is of interest to information developers, information managers, data administrators, standards developers, application developers, business modellers and others who are responsible for making data understandable and shareable. ISO/IEC 11179 has broad applicability across subject areas and information technologies.

ISO/IEC 11179 applies to activities including:

a) the definition, specification and registration of contents of metadata registries, including interchanging or referencing among various collections of data elements[17], including data sets[19] and models[21][29];

b) interchange or reference among various collections of metadata, including models[21][29];

c) the registration and management of semantic artifacts that are useful for data management, data administration and data analysis;

d) the interrelation of concept systems with data held in relational databases, XML databases, knowledgebases, text, and possibly graph databases deriving from natural language text understanding systems;

e) the provision of services for semantic computing: Semantics Service Oriented Architecture, Semantic Grids, semantics-based workflows, Semantic Web, etc;

f) support for addressing semantic web considerations such as AAA (anyone can say anything about anything), non-unique names and open world assumption.

In Clauses 5 through 11 and Annexes B and C, this document uses:

— **bold** font to highlight terms which represent metadata objects specified by the metamodel;

— normal font for terms which represent concepts defined in Clause 3.

EXAMPLE **Concept** (6.4.2.2) is a class each instance of which models a concept.

Information technology — Metadata registries (MDR) —

Part 3:
Metamodel for registry common facilities

1 Scope

1.1 Structure of a metadata registry

This document specifies the information to be recorded in a metadata registry in the form of a conceptual data model:

— Clause 5 specifies the approach used to model a metadata registry;

— Clause 6 specifies the Core Model of the registry, including basic types and classes to be reused in extending the model. The core model defines a generic “registry item”, from which any type of item that needs to registered can be sub-classed;

— Clause 7 specifies the metamodel for Identification of registry items;

— Clause 8 specifies the metamodel for Designation and Definition of registry items;

— Clause 9 specifies the metamodel for Registration of registry items;

— Clause 10 specifies the metamodel for Classification of registry items;

— Clause 11 specifies the metamodel for Mapping among registry items.

1.2 Model extensions

Other parts of ISO/IEC 11179 extend the core model to support additional functionality, including the following:

— ISO/IEC 11179-31[17] provides a metamodel for data specification registration, including support for data elements, data element concepts, conceptual domains and value domains;

— ISO/IEC 11179-32[18] provides a metamodel for concept system registration , including support for concept systems and ontologies;

— ISO/IEC 11179-33[19] provides a metamodel for data set registration;

— ISO/IEC 11179-34[20] (under development) provides a metamodel for computable data registration;

— ISO/IEC 11179-35[21] provides a metamodel for model registration.

2 Normative references

The following referenced documents are indispensable for the application 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/IEC 11179-6:2023, *Information technology — Metadata registries (MDR) — Part 6: Registration*

ISO/IEC 11404:2007, *Information technology — General-Purpose Datatypes (GPD)*

3 Terms, definitions and abbreviated terms

For the purposes of this document, the following terms, definitions and abbreviated terms 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 https://www.electropedia.org/

3.1 Terms related to metamodel constructs

3.1.1
class
<metamodel> description of a set of *objects* (3.2.1) that share the same *attributes* (3.1.11), operations, methods, *relationships* (3.1.4) and semantics

Note 1 to entry: Adapted from ISO/IEC 19505-2:2012, 7.3.7[27].

3.1.2
abstract class
<metamodel> *class* (3.1.1) that does not provide a complete declaration and typically cannot be instantiated

Note 1 to entry: An abstract class is intended to be used by other classes as a general class for *specialization* (3.1.8).

Note 2 to entry: Cf. *concrete class* (3.1.3).

Note 3 to entry: Adapted from ISO/IEC 19505-1:2012, 9.19.1[26] Classifier.isAbstract.

3.1.3
concrete class
<metamodel> *class* (3.1.1) that can be instantiated

Note 1 to entry: Cf. *abstract class* (3.1.2).

3.1.4
relationship
<metamodel> connection among model elements

Note 1 to entry: In this document, a relationship is one of: an *association* (3.1.5), a *generalization* (3.1.7) or a *specialization* (3.1.8).

3.1.5
association
<metamodel> semantic *relationship* (3.1.4) between two *classes* (3.1.1) or between two or more instances of the same class

Note 1 to entry: An association is a type of relationship.

Note 2 to entry: Adapted from ISO/IEC 19505-2:2012, 7.3.3[27] Association.

3.1.6
association class
<metamodel> *association* (3.1.5) that is also a *class* (3.1.1)

Note 1 to entry: An association class not only connects a set of classes, but also defines a set of features that belong to the association itself.

Note 2 to entry: Adapted from ISO/IEC 19505-2:2012, 7.3.4[27] AssociationClass.

3.1.7
generalization
<metamodel> *relationship* (3.1.4) between a more general class (the parent) and a more specific class (the child) that is fully consistent with the general class and that adds additional information

Note 1 to entry: A generalization is a type of *relationship* (3.1.4).

Note 2 to entry: The more general class is referred to as the *superclass* (3.1.9).

Note 3 to entry: The more specific class is referred to as a *subclass* (3.1.10).

Note 4 to entry: A generalization is directed from the *subclass* to the *superclass*.

Note 5 to entry: 'Fully consistent' means that the *subclass* has all of the *attributes* (3.1.11) and *relationships* (3.1.4) of the *superclass*.

Note 6 to entry: cf. *specialization* (3.1.8); generalization is the inverse of specialization.

Note 7 to entry: Adapted from ISO/IEC 19505-2:2012, 7.3.20[27].

3.1.8
specialization
<metamodel> *relationship* (3.1.4) between a more general class (the parent) and a more specific class (the child) that is fully consistent with the general class and that adds additional information

Note 1 to entry: A specialization is a type of *relationship* (3.1.4).

Note 2 to entry: The more general class is referred to as the *superclass* (3.1.9).

Note 3 to entry: The more specific class is referred to as a *subclass* (3.1.10).

Note 4 to entry: A specialization is directed from the *superclass* to the *subclass*.

Note 5 to entry: 'Fully consistent' means that the *subclass* has all of the *attributes* (3.1.11) and *relationships* (3.1.4) of the *superclass*.

Note 6 to entry: cf. *generalization* (3.1.7); *specialization* is the inverse of *generalization*.

Note 7 to entry: Adapted from ISO/IEC 19505-2:2012, 7.3.20[27].

3.1.9
superclass
<metamodel> *class* (3.1.1) that is a *generalization* (3.1.7) of one or more other classes

Note 1 to entry: The classes being generalized are known as *subclasses* (3.1.10).

Note 2 to entry: In UML, **s**ubclasses of a superclass are by default not *disjoint* (3.2.39). This document specifies when subclasses are required to be disjoint. Further, when the superclass is not intended to be instantiated alone, this document shows the superclass as abstract, thus preventing the superclass being instantiated other than through one or more of its subclasses. An *abstract class* (3.1.2) is indicated by showing the class name in *italics* in any class diagram that uses it.

Note 3 to entry: A particular class can be a superclass with respect to one relationship, but a subclass with respect to another relationship.

3.1.10
subclass
<metamodel> *class* (3.1.1) that is a *specialization* (3.1.8) of another *class*

Note 1 to entry: The class being specialized is known as the *superclass* (3.1.9).

Note 2 to entry: In UML, subclasses of a superclass are by default not *disjoint* (3.2.39). This document specifies when subclasses are required to be disjoint. Further, when the superclass is not intended to be instantiated alone, this document shows the superclass as abstract, thus preventing the superclass being instantiated other than through one or more of its subclasses. An *abstract class* (3.1.2) is indicated by showing the class name in *italics* in any class diagram that uses it.

Note 3 to entry: A particular class can be a subclass with respect to one relationship, but a superclass with respect to another relationship.

3.1.11
attribute
<metamodel> *characteristic* (3.2.4) of an *object* (3.2.1) or set of objects

3.1.12
composite attribute
<metamodel> *attribute* (3.1.11) whose *datatype* (3.1.13) is non-atomic

EXAMPLE See Administered_Item.registration_state (9.4.2.4), where **registration_state** is a composite attribute with Registration_State (9.4.4) as its *composite datatype* (3.1.15).

3.1.13
datatype
set of distinct values, characterized by properties of those values and by operations on those values

[SOURCE: ISO/IEC 11404:2007, 3.12]

3.1.14
primitive datatype
datatype (3.1.13) that cannot be decomposed into other datatypes without loss of associated semantics

[SOURCE: ISO/IEC 11404:2007, 3.44, modified — Deleted “identifiable” as a qualifier for datatype and deleted “all” as a qualifier for associated semantics.]

3.1.15
composite datatype
<metamodel> *datatype* (3.1.13) that is also a *class* (3.1.1)

Note 1 to entry: A composite datatype is used as a datatype for a *composite attribute* (3.1.12).

EXAMPLE See Administered_Item.registration_state (9.4.2.4) where Registration_State (9.4.4) serves as the composite datatype for the composite attribute **registration_state**.

3.1.16
identifier
sequence of characters, capable of uniquely identifying that with which it is associated, within a specified *context* (3.2.15)

Note 1 to entry: Unlike a *name* (3.2.12), an identifier is linguistically neutral.

3.1.17
package
<metamodel> grouping of *metadata objects* (3.2.31) that provides a *namespace* (3.2.13) for the grouped objects and allows them to be referenced as a group

Note 1 to entry: Adapted from ISO/IEC 19505-2:2012, 7.3.38[27].

3.1.18
metamodel region
<metamodel> sub-division of a *package* (3.1.17) used to organize *metadata objects* (3.2.31) for ease of explanation

3.2 Terms related to concepts

3.2.1
object
anything perceivable or conceivable

Note 1 to entry: Objects can be material (e.g. 'engine', 'sheet of paper', 'diamond'), immaterial (e.g. 'conversion ratio', 'project plan') or imagined (e.g. 'unicorn', 'scientific hypothesis').

[SOURCE: ISO 1087:2019, 3.1.1]

3.2.2
object class
set of ideas, abstractions or things in the real world that are identified with explicit boundaries and meaning and whose properties and behaviour follow the same rules

[SOURCE: ISO/IEC 11179-31:2023, 3.1]

3.2.3
property
quality common to all members of an *object class* (3.2.2)

[SOURCE: ISO/IEC 11179-31:2023, 3.2]

3.2.4
characteristic
abstraction of a *property* (3.2.3)

EXAMPLE 'Having a cable for connecting with a computer' as a characteristic of the concept 'cord mouse'.

Note 1 to entry: Characteristics are used for describing *concepts* (3.2.7).

[SOURCE: ISO 1087:2019, 3.2.1]

3.2.5
classification scheme
descriptive information for an arrangement or division of *objects* (3.2.1) into groups based on criteria such as *characteristics* (3.2.4), which the objects have in common

EXAMPLE Origin, composition, structure, application, function, etc.

3.2.6
classification scheme item
item of content in a *classification scheme* (3.2.5)

Note 1 to entry: This can be a node in a taxonomy or ontology, a term in a thesaurus, etc.

3.2.7
concept
unit of knowledge created by a unique combination of *characteristics* (3.2.4)

Note 1 to entry: Concepts are not necessarily bound to particular natural languages. They are, however, influenced by the social or cultural background which often leads to different categorizations.

Note 2 to entry: A concept is independent of its representation.

[SOURCE: ISO 1087:2019, 3.2.7, modified — Note 2 to entry changed.]

3.2.8
concept relation
relation between *concepts* (3.2.7)

[SOURCE: ISO 1087:2019, 3.2.11]

3.2.9
concept system
set of *concepts* (3.2.7) structured in one or more related domains according to the *concept relations* (3.2.8) among its concepts

Note 1 to entry: In this definition, 'domain' is used in the sense of 'field of special knowledge' as defined in ISO 1087:2019, 3.1.4.

[SOURCE: ISO 1087:2019, 3.2.28, modified — Note 1 to entry added.]

3.2.10
sign (noun)
textual string or symbol that can be used to denote a *concept* (3.2.7)

3.2.11
designation
<registry> representation of a *registry item* (3.2.54) by a *sign* (3.2.10) which denotes it

3.2.12
name
designation of an *object* (3.2.1) by a linguistic expression

[SOURCE: ISO/IEC 15944-1:2011, 3.35]

3.2.13
namespace
set of *designations* (3.2.11), *scoped identifiers* (3.2.62) or both for a particular business need

Note 1 to entry: The term *namespace* is used in this document because it is in common use, even though the concept is being applied to identifiers as well as names.

3.2.14
acceptability rating
scale of acceptability

Note 1 to entry: Subclause 8.7.1 specifies the values to be used.

3.2.15
context
circumstance, purpose, and perspective under which an *object* (3.2.1) is defined or used

[SOURCE: ISO/IEC 11179-1:2023, 3.3.3]

3.2.16
language
systematic use of sounds, characters, symbols or *signs* (3.2.10) to express meaning or communicate meaning or a message

[SOURCE: ISO 5127:2017, 3.1.5.01]

3.2.17
designation acceptability
acceptability rating (3.2.14) of the *designation* (3.2.11) in the specified *designation context* (3.2.18)

3.2.18
designation context
context (3.2.15) in which a *designation* (3.2.11) is applicable

3.2.19
designation language
language (3.2.16) or dialect in which a *sign* (3.2.10), usually a *name* (3.2.12), is expressed

3.2.20
designation namespace
namespace (3.2.13) to which a *designation* (3.2.11) is bound

3.2.21
binding
mapping from one framework or specification to another, enabling *data* (3.2.29), commands or both to be passed between them

3.2.22
naming convention
specification of how *signs* (3.2.10) of *designations* (3.2.11), *scoped identifiers* (3.2.62) or both are formulated

Note 1 to entry: A naming convention can apply to scoped identifiers when they are included in the associated *namespace* (3.2.13)

3.2.23
definition
<registry> representation of a *registry item* (3.2.54) by a descriptive statement which, in a given *language* (3.2.16) and *context(s)* (3.2.15), serves to differentiate it from other registry items

3.2.24
definition text
text of the *definition* (3.2.23)

[SOURCE: ISO 1087:2019, 3.4.1]

3.2.25
definition acceptability
acceptability rating (3.2.14) of the *definition* (3.2.23) in the specified *definition context* (3.2.26)

3.2.26
definition context
context (3.2.15) in which the *definition* (3.2.23) is applicable

3.2.27
definition language
language (3.2.16) used to write the *definition text* (3.2.24)

3.2.28
definition source reference
reference to the source from which the *definition* (3.2.23) is taken

3.2.29
data
re-interpretable representation of information in a formalized manner suitable for communication, interpretation, or processing

Note 1 to entry: Data can be processed by humans or by automatic means.

[SOURCE: ISO/IEC 2382:2015, 2121272, modified — Notes to entry 2 and 3 deleted.]

3.2.30
metadata
data (3.2.29) that defines and describe other data

[SOURCE: ISO/IEC 11179-1:2023, 3.2.26]

3.2.31
metadata object
object type defined by a *metamodel* (3.2.34)

Note 1 to entry: In all parts of ISO/IEC 11179, this term is applied only to metadata objects described by the metamodels in those parts of ISO/IEC 11179 which specify potential registry content, including *data elements* (3.2.43) in ISO/IEC 11179-31[17], *concept systems* (3.2.9) in ISO/IEC 11179-32[18], data sets in ISO/IEC 11179-33[19], computable data in ISO/IEC 11179-34[20], and models in ISO/IEC 11179-35[21].

Note 2 to entry: The term also applies to metadata objects described by the metamodels in the various parts of ISO/IEC 19763[29], which build upon the metamodel in this document.

3.2.32
data model
graphical, lexical or combined representation of *data* (3.2.29), specifying their *properties* (3.2.3), structure and inter-relationships

[SOURCE: ISO/IEC 11179-1:2023, 3.2.24]

3.2.33
conceptual data model
conceptual model
data model (3.2.32) that represents an abstract view of the real world

Note 1 to entry: A conceptual model represents the human understanding of a system, which can be anywhere from a paper-based system to a complex database in an IT system.

[SOURCE: ISO/IEC 11179-1:2023, 3.2.25]

3.2.34
metamodel
data model (3.2.32) that specifies one or more other models, such as data models, process models, ontologies, etc

Note 1 to entry: The *metamodel* in this document is expressed as a *conceptual data model* (3.2.33) using UML class diagram *notation* (3.2.36).

[SOURCE: ISO/IEC 11179-1:2023, 3.2.27, modified — Note 1 to entry added.]

3.2.35
metamodel construct
unit of *notation* (3.2.36) for modelling

Note 1 to entry: The metamodel constructs used in this document are defined in 3.1.

3.2.36
notation
formal syntax and associated semantics for the representation of information

EXAMPLE UML, MOF, OCL, OWL/RDF, SKOS, CGIF, XCL, XTM or ISO/IEC 11404.

Note 1 to entry: A formal syntax consists of a set of symbols and the rules for their use.

Note 2 to entry: A formal syntax is often intended for machine processing.

3.2.37
cardinality
number of elements in a set

Note 1 to entry: cf. *multiplicity* (3.2.38).

Note 2 to entry: Adapted from ISO/IEC 19501:2005[25], Glossary.

3.2.38
multiplicity
specification of the range of allowable *cardinalities* (3.2.37) that a set can assume

Note 1 to entry: Multiplicity specifications can be given for roles within *associations* (3.1.5) or *attributes* (3.1.11) within a *class* (3.1.1).

Note 2 to entry: A multiplicity is a (possibly infinite) subset of the non-negative integers.

Note 3 to entry: Adapted from ISO/IEC 19501:2005, Glossary.

3.2.39
disjoint
having no elements in common

Note 1 to entry: Disjoint subclasses are mutually exclusive.

3.2.40
extension
feature not defined by this document

3.2.41
model extension
class (3.1.1), *attribute* (3.1.11) or *relationship* (3.1.4) that an implementation of a *registry* (3.2.49) provides that is not defined by this document or related standards

Note 1 to entry: This document specifies *slots* (3.2.42) as a mechanism for extending *registry items* (3.2.54).

3.2.42
slot
container for an *extension* (3.2.40) to a *registry item* (3.2.54)

3.2.43
data element
<organization of data> unit of *data* (3.2.29) that is considered in context to be indivisible

EXAMPLE The data element "age of a person" with values consisting of all combinations of 3 decimal digits.

Note 1 to entry: The definition states that a data element is "indivisible" in some context. This means that it is possible that a data element considered indivisible in one context (e.g. telephone number) can be divisible in another context, (e.g. country code, area code, local number).

[SOURCE: ISO/IEC 2382:2015, 2121599, modified — Note 1 to entry label removed from the example. Note 2 to entry replaced and renumbered as Note 1, Note 3 to entry deleted.]

3.2.44
data element concept
DEC
concept (3.2.7) that can be represented in the form of a *data element* (3.2.43), described independently of any particular representation

Note 1 to entry: A data element concept is implicitly associated with both the property and the object class whose combination it expresses.

[SOURCE: ISO/IEC 11179-31:2023, 3.25, modified — Notes 2 and 3 to entry deleted.]

3.2.45
conceptual domain
CD
concept (3.2.7) whose meaning is expressed as an enumerated set, a description or both of subordinate concepts, which are *value meanings* (3.2.46)

[SOURCE: ISO/IEC 11179-31:2023, 3.5]

3.2.46
value meaning
semantic content of a value

Note 1 to entry: The representation of value meanings in a *registry* (3.2.49) are independent of (and do not constrain) their representation in any corresponding *value domain* (3.2.47).

[SOURCE: ISO/IEC 11179-31:2023, 3.10]

3.2.47
value domain
VD
set of *permissible values* (3.2.48)

Note 1 to entry: The *value domain* provides representation but has no implication as to what *data element concept* (3.2.44) the values are associated with nor what the values mean.

Note 2 to entry: The *permissible values* can either be enumerated, expressed via a description, or a combination of the two.

[SOURCE: ISO/IEC 11179-31:2023, 3.13]

3.2.48
permissible value
designation (3.2.11) of a *value meaning* (3.2.46)

Note 1 to entry: *Permissible values* may be specified either as part of a *value domain* (3.2.47) or only associated with a *value meaning* (3.2.46).

Note 2 to entry: Within a *value domain*, *permissible values* can either be enumerated, expressed via a description, or a combination of the two.

Note 3 to entry: Explicit mapping of a single permissible value to a single value meaning is possible only when both the value meaning and permissible value are enumerated, e.g. for code sets. For described permissible values, it is possible for the described meaning to be associated with a range of values, e.g. weight in kilograms.

[SOURCE: ISO/IEC 11179-31:2023, 3.19]

3.2.49
registry
information system for *registration* (3.2.64)

[SOURCE: ISO/IEC 11179-1:2023, 3.2.34]

3.2.50
register
information store or database maintained by a *registry* (3.2.49)

3.2.51
registry product
particular information system for implementing a *registry* (3.2.49)

3.2.52
registry instance
implementation of a *registry product* (3.2.51) and instance of a *registry* (3.2.49)

3.2.53
registry metamodel
metamodel (3.2.34) specifying the model for a *registry* (3.2.49)

3.2.54
registry item
item recorded in a *registry* (3.2.49)

Note 1 to entry: A *registry item* is recorded in the *registry*, but is not necessarily identified, named, defined, classified, registered or administered. Specific information needs to be provided for each of these categories which can be provided when the item is initially recorded, or later. See also *classified item* (3.2.59), *identified item* (3.2.61), *registered item* (3.2.65), *administered item* (3.2.66) and *attached item* (3.2.68).

3.2.55
metadata registry
MDR
information system for registering *metadata* (3.2.30)

Note 1 to entry: The associated information store or database is known as a *metadata register* (3.2.56).

[SOURCE: ISO/IEC 11179-1:2023, 3.2.31, modified — Note 1 to entry added.]

3.2.56
metadata register
information store or database maintained by a *metadata registry* (3.2.55)

3.2.57
metadata registry product
particular information system for implementing a *metadata registry* (3.2.55)

3.2.58
metadata item
instance of a *metadata object* (3.2.31) in a *metadata registry* (3.2.55)

Note 1 to entry: In all parts of ISO/IEC 11179, this term is applied only to instances of metadata objects described by the metamodels in parts of ISO/IEC 11179 which specify potential registry content, including *data elements* (3.2.43) in ISO/IEC 11179-31[17], *concept systems* (3.2.9) in ISO/IEC 11179-32[18], data sets in ISO/IEC 11179-33[19], computable objects in ISO/IEC 11179-34[20], and models in ISO/IEC 11179-35[21].

Note 2 to entry: The term also applies to instances of metadata objects described by the metamodels in the various parts of ISO/IEC 19763[29], which build upon the metamodel in this document.

Note 3 to entry: A metadata item has associated *attributes* (3.1.11), as appropriate for the *metadata object* it instantiates.

3.2.59
classified item
registry item (3.2.54) classified in a *classification scheme* (3.2.5)

3.2.60
identification scheme
system for allocating *identifiers* (3.1.16) to registered *objects* (3.2.1)

[SOURCE: ISO/IEC 6523-1:1998, 3.6]

3.2.61
identified item
registry item (3.2.54) identified in a *registry* (3.2.49)

3.2.62
scoped identifier
identifier (3.1.16) of an *identified item* (3.2.61) within a specified *namespace* (3.2.13)

Note 1 to entry: A *namespace* provides the scope within which the *scoped identifier* uniquely identifies the *identified item*.

3.2.63
version
unique form differing in certain aspects from an earlier or later form

[SOURCE: ISO/IEC 11179-1:2023, 3.3.25]

3.2.64
registration
set of rules, operations and procedures for the management of an item in a *registry* (3.2.49)

Note 1 to entry: A detailed description of registration as it applies in ISO/IEC 11179 is found in ISO/IEC 11179-6.

Note 2 to entry: In this document, registration requires that a minimum set of *administrative information* (3.2.67) about the *registry item* (3.2.54) be specified, such that it becomes a *registered item* (3.2.65).

Note 3 to entry: A *registry item* is recorded in the *registry*, but is not necessarily identified, named, defined, classified, registered or administered. Specific information needs to be provided for each of these categories which can be provided when the item is initially recorded, or later. See also *classified item* (3.2.55), *identified item* (3.2.61), *registered item* (3.2.62), *administered item* (3.2.66) and *attached item* (3.2.68).

[SOURCE: ISO/IEC 11179-1:2023, 3.3.33, modified — Notes 2 and 3 to entry added.]

3.2.65
registered item
identified item (3.2.61) that is recorded and managed in a *registry* (3.2.49)

Note 1 to entry: In this document, registration requires that a minimum set of *administrative information* (3.2.67) about the *registry item* (3.2.54) be specified, such that it becomes a *registered item*.

Note 2 to entry: A *registry item* is recorded in the *registry*, but is not necessarily identified, named, defined, classified, registered or administered. Specific information needs to be provided for each of these categories which can be provided when the item is initially recorded, or later. See also *classified item* (3.2.59), *identified item* (3.2.61), *registered item* (3.2.65), *administered item* (3.2.66) and *attached item* (3.2.68).

3.2.66
administered item
registered item (3.2.65) for which *administrative information* (3.2.67) is recorded

3.2.67
administrative information
<registry> information about the administration of an item in a *registry* (3.2.49)

EXAMPLE Creation date, last change date, origin, change description, explanatory comment.

3.2.68
attached item
registered item (3.2.65) for which *administrative information* (3.2.67) is recorded in another registered item

Note 1 to entry: This is often a member of a group of registered items that is managed as a whole.

3.2.69
registration state
information about the *registration* (3.2.64) of an *administered item* (3.2.66)

3.2.70
registration status
designation (3.2.11) of the status in the *registration* (3.2.64) life-cycle of an *administered item* (3.2.66)

Note 1 to entry: Designation values are specified in 9.6.1 and described more fully in ISO/IEC 11179-6:2023, 4.3.2.

3.2.71
administrative status
designation (3.2.11) of the status in the administrative process of a *registration authority* (3.2.92)

Note 1 to entry: The values and associated meanings of the *administrative status* are determined by each *registration authority*. Cf. *registration status* (3.2.70).

3.2.72
basic attribute
<registry> *attribute* (3.1.11) of a *registry item* (3.2.54) commonly needed in its specification

3.2.73
attribute instance
specific instance of an *attribute* (3.1.11)

Note 1 to entry: Adapted from ISO/IEC 2382:2015, 2121441[5] attribute value, to distinguish an instance of an attribute from its value.

3.2.74
attribute value
value associated with an *attribute instance* (3.2.73)

Note 1 to entry: Adapted from ISO/IEC 2382:2015, 2121441[5] attribute value, to distinguish an instance of an attribute from its value.

3.2.75
mandatory
always required

Note 1 to entry: One of three obligation statuses applied to the *attributes* (3.1.11) of *registry items* (3.2.54), indicating the conditions under which the attribute is required; see also *conditional* (3.2.77) and *optional* (3.2.76).

Note 2 to entry: Obligation statuses apply to *registry items* with a *registration status* (3.2.70) of "recorded" or higher.

3.2.76
optional
permitted but not required

Note 1 to entry: One of three obligation statuses applied to the *attributes* (3.1.11) of *registry items* (3.2.54), indicating the conditions under which the attribute is required. See also *conditional* (3.2.77) and *mandatory* (3.2.75).

Note 2 to entry: Obligation statuses apply to *registry items* with a *registration status* (3.2.70) of "recorded" or higher.

3.2.77
conditional
required under certain specified conditions

Note 1 to entry: One of three obligation statuses applied to the *attributes* (3.1.11) of *registry items* (3.2.54), indicating the conditions under which the attribute is required. See also *mandatory* (3.2.75) and *optional* (3.2.76).

Note 2 to entry: Obligation statuses apply to *registry items* with a *registration status* (3.2.70) of "Recorded" or higher.

3.2.78
contact
instance of a *role* (3.2.83), an *individual* (3.2.82) or both within an *organization* (3.2.84) to or from whom an information item(s), a material object(s), person(s) or some combination can be sent in a specified context

3.2.79
contact information
information to enable a *contact* (3.2.78) to be located or communicated with

3.2.80
phone number
telephone number

Note 1 to entry: Specified by ITU-T Recommendation E.164 (2005-02)[35], the international public telecommunications numbering plan.

3.2.81
postal address
set of information which, for a postal item, allows the unambiguous determination of an actual or potential delivery point, usually combined with the specification of an addressee or a mailee

[SOURCE: UPU S42,[38] modified — Replaced “and/or” by “or”]

3.2.82
individual
single human being

3.2.83
role
specified responsibilities

3.2.84
organization
unique framework of authority within which *individuals* (3.2.82) act, or are designated to act, towards some purpose

Note 1 to entry: The kinds of organizations covered by ISO/IEC 6523-1[8] include the following examples:

a) an organization incorporated under law;

b) an unincorporated organization or activity providing goods, services or both including:

   1) partnerships;

   2) social or other non-profit organizations or similar bodies in which ownership or control is vested in a group of individuals;

   3) sole proprietorships;

   4) governmental bodies.

c) groupings of the above types of organizations where there is a need to identify these in information interchange.

[SOURCE: ISO/IEC 6523-1:1998, 3.1, modified — ‘goods and/or services’ changed to ‘goods, services or both’, 'person or persons' changed to 'individuals', leading article deleted, trailing fullstop deleted, NOTE converted to Note 1 to entry.]

3.2.85
organization identification scheme
identification scheme (3.2.60) dedicated to the unique identification of *organizations* (3.2.84)

[SOURCE: ISO/IEC 6523-1:1998, 3.7]

3.2.86
international code designator
identifier (3.1.16) of an *organization identification scheme* (3.2.85)

Note 1 to entry: See also ISO/IEC 11179-6:2023, Annex A, A.4.

[SOURCE: ISO/IEC 6523-1:1998, 3.8, modified — Used 'identifier of' instead of 'the data element used to uniquely identify', Note 1 to entry added.]

3.2.87
organization identifier
identifier (3.1.16) assigned to an *organization* (3.2.84) within an *organization identification scheme* (3.2.85) and unique within that scheme

[SOURCE: ISO/IEC 6523-1:1998, 3.10]

3.2.88
organization part
any department, service or other entity within an *organization* (3.2.84) which needs to be identified for information exchange

[SOURCE: ISO/IEC 6523-1:1998, 3.2]

3.2.89
organization part identifier
opi
identifier (3.1.16) allocated to a particular *organization part* (3.2.88)

Note 1 to entry: See also ISO/IEC 11179-6.

[SOURCE: ISO/IEC 6523-1:1998, 3.11, modified — Note 1 to entry added.]

3.2.90
registrar
representative of a *registration authority* (3.2.92)

3.2.91
registrar identifier
identifier (3.1.16) for the *registrar* (3.2.90)

3.2.92
registration authority
RA
organization (3.2.84) responsible for maintaining a *register* (3.2.50)

3.2.93
registration authority identifier
identifier (3.1.16) assigned to a *registration authority* (3.2.92)

3.2.94
boolean
mathematical *datatype* (3.1.13) associated with two-valued logic

Note 1 to entry: See also 6.2.2 Boolean datatype.

Note 2 to entry: Adapted from ISO/IEC 11404:2007, 8.1.1 Boolean.

3.2.95
date
datatype (3.1.13) whose values are points in time to the resolution: year, month, day

Note 1 to entry: See also 6.2.4 Date datatype.

Note 2 to entry: Adapted from ISO/IEC 11404:2007, 8.1.6 date and time.

3.2.96
datetime
datatype (3.1.13) whose values are points in time to the resolution: year, month, day, hour, minute, second and optionally fractions thereof and optionally a time-zone specification

Note 1 to entry: See also 6.2.3 Datetime datatype.

Note 2 to entry: Adapted from ISO/IEC 11404:2007, 8.1.6 date and time.

3.2.97
datetime period
combination of an optional start *datetime* (3.2.96) and an optional end *datetime*

Note 1 to entry: At least one of the start or end datetimes shall be specified.

Note 2 to entry: A period can be open ended, meaning that the end datetime is not yet known, and is therefore not specified.

Note 3 to entry: A period can be missing a start datetime if that is unknown.

3.2.98
integer
mathematical *datatype* (3.1.13) comprising the exact integral values

Note 1 to entry: See also 6.2.5 Integer datatype.

Note 2 to entry: Adapted from ISO/IEC 11404:2007, 8.1.7.

3.2.99
string
family of *datatypes* (3.1.13) which represent strings of symbols from standard character-sets

Note 1 to entry: See also 6.2.11 String datatype.

Note 2 to entry: Adapted from ISO/IEC 11404:2007, 10.1.5 Character String.

3.2.100
text
data (3.2.29) in the form of characters, symbols, words, phrases, paragraphs, sentences, tables, or other character arrangements, intended to convey a meaning, and whose interpretation is essentially based upon the reader's knowledge of some natural language or artificial language

Note 1 to entry: See also the **Text** datatype (6.2.12).

[SOURCE: ISO/IEC 2382:2015(en), 2121273, modified — Note 1 to entry added.]

3.2.101
reference document
document that provides pertinent details for consultation about a subject

3.2.102
stewardship
responsibility for the maintenance of *administrative information* (3.2.67) applicable to one or more *administered items* (3.2.66)

Note 1 to entry: The responsibility for the *registration* (3.2.64) of an *administered item* (3.2.66) can be different from the responsibility for *stewardship* of the *administered item*.

3.2.103
stewardship contact
contact information (3.2.79) associated with a *stewardship* (3.2.102)

3.2.104
stewardship organization
organization (3.2.84) that maintains *stewardship* (3.2.102) of an *administered item* (3.2.66)

3.2.105
stewardship record
record of a *stewardship organization* (3.2.104) and a *stewardship contact* (3.2.103) involved in the *stewardship* (3.2.102) of an *administered item* (3.2.66)

3.2.106
submission
act of submitting a *registry item* (3.2.54) for *registration* (3.2.55) in a *registry* (3.2.49)

3.2.107
submission contact
submitter
contact information (3.2.79) associated with a *submission* (3.2.106)

3.2.108
submission organization
submitting organization
organization (3.2.84) that submits a *registry item* (3.2.54) for *registration* (3.2.55)

3.2.109
submission record
record of a *submission organization* (3.2.108) and a *submission contact* (3.2.107) involved in the *submission* (3.2.106) of a *registry item* (3.2.54) for *registration* (3.2.55)

3.3 Abbreviated terms

CD	conceptual domain
DOI	digital object identifier (see Reference [32])
EBNF	extended Backus-Naur form (see Reference [22])
MDR	metadata registry
OPI	organization part identifier
RA	registration authority
RDF	resource description framework (see Reference [37])
UML	unified modeling language (see References [26] and [27])
UPU	Universal Postal Union (see Reference [38])
URI	uniform resource identifier
URL	uniform resource locator
VD	value domain
W3C	World Wide Web Consortium
XCL	extended Common Logic markup language (see Reference [31])
XML	extensible markup language

4 Conformance

4.1 Overview of conformance

This document prescribes a conceptual model, not a physical implementation. Therefore, the metamodel need not be physically implemented exactly as specified. However, it shall be possible to unambiguously map between the implementation and the metamodel in both directions.

Conformance claims shall specify a "degree of conformance" (4.2) and the features to which conformance is claimed (4.3). Conformance statements with respect to this document shall also be explicit as to which portions of this document conformity is being claimed. This may be done in some cases simply by reference to one or more of the clauses. In other cases, conformance may instead be claimed to one or more of the standard profiles, which specify combinations of multiple clauses, and how they are to be fitted together. Two simple standard profiles are specified in 4.4.2.

When a registry product makes a conformance claim, the product shall support all the associated functionality, and shall enable the enforcement of the associated constraints. When a registry instance makes a conformance claim, it shall actually enforce the associated constraints.

4.2 Degree of conformance

4.2.1 General

The distinction between "strictly conforming" and "conforming" implementations is necessary to address the simultaneous needs for interoperability and extensions. This document describes specifications that promote interoperability. Extensions are motivated by needs of users, vendors, institutions, and industries, and:

a) are not explicitly specified by this document;

b) are specified and agreed to outside this document;

c) can serve as trial usage for future editions of this document.

A strictly conforming implementation can be limited in usefulness but is maximally interoperable with respect to this document. A conforming implementation can be more useful but can be less interoperable with respect to this document.

4.2.2 Strictly conforming implementations

A strictly conforming implementation:

a) shall support all mandatory, optional and conditional classes, attributes, datatypes and associations;

b) shall not use, test, access or probe for any extension features nor extensions to classes, attributes, datatypes, associations or any combination thereof;

c) shall not recognize, nor act on, nor allow the production of classes, attributes, datatypes, associations or any combination thereof that are dependent on any unspecified, undefined or implementation-defined behaviour.

NOTE The use of extensions to the metamodel can cause undefined behaviour.

4.2.3 Conforming implementations

A conforming implementation:

a) shall support all mandatory, optional and conditional classes, attributes, datatypes and associations;

b) as permitted by the implementation, may use, test, access or probe for extension features or extensions to classes, attributes, datatypes, associations or any combination thereof;

c) may recognize, act on or allow the production of classes, attributes, datatypes, associations or any combination thereof that are dependent on implementation-defined behaviour.

NOTE 1 All strictly conforming implementations are also conforming implementations.

NOTE 2 The use of extensions to the metamodel can cause undefined behaviour.

4.3 Conformance by feature

Conformance claims may be limited to individual clauses of this document or to specific features within these clauses. Some clauses are dependent upon one or more other clauses of this document (see Figure 1), so conformance to any of these clauses shall be understood to imply conformance also to relevant provisions specified in one or more other clauses.

Conformance may be claimed for a set of data structures, datatypes or both for clauses:

— Clause 6 Basic and Core package;

— Clause 7 Identification package;

— Clause 8 Designation and Definition package;

— Clause 10 Classification package;

— Clause 11 Item Mapping package.

Conformance may be claimed for a register (a set of administered metadata) or for a registry software system for:

— Clause 9 Registration package.

A conformance statement shall specify exactly the features supported and not supported.

4.4 Registry conformance

4.4.1 Overview

Registries shall conform to Clause 9. Registries conforming to Clause 9 may additionally claim conformance to one of the standard profiles specified in other parts of ISO/IEC 11179 or ISO/IEC 19763, each of which extends the core model in this document.

4.4.2 Standard profiles for registries

Implementers of either a generic registry platform (software system) which is customizable for a range of registered content types, or of a register or registry software system supporting some specific range of registered content types not specified in this document, can claim conformance to one of the following standard profiles:

— **Basic Registry**: Implements all clauses in this document except Clause 11 Item Mapping package;

— **Basic Registry with mapping**: Implements all clauses in this document including Clause 11 Item Mapping package.

Additional standard profiles are defined in other parts of ISO/IEC 11179 (e.g. ISO/IEC 11179-31[17], ISO/IEC 11179-32[18], ISO/IEC 11179-33[19], ISO/IEC 11179-34[20] and ISO/IEC 11179-35[21]), specifying how additional clauses should be combined with this document.

4.4.3 Conformance labels

Conformance to the profiles specified in 4.4.2 may be claimed using the following labels, respectively:

— ISO/IEC 11179-3:2023 Basic Registry;

— ISO/IEC 11179-3:2023 Basic Registry with mapping.

4.5 Obligation

Attributes and associations specified in this document are one of: Mandatory, Conditional or Optional. The obligation is not explicitly stated but is to be inferred from the multiplicity of the attribute or association, and the presence or absence of a condition. In addition, a registration authority can specify additional constraints to be applied to particular instances of the **Administered_Item** class (9.4.2) by using the facilities provided by the **Constraint_Set** class (9.4.5).

For the purpose of conformance:

a) Mandatory attributes and associations shall exist and shall conform to the provisions of this document.

b) Anything specified as Conditional within this document shall be treated as Mandatory if the associated condition is satisfied and shall otherwise be not present.

c) Optional attributes and associations are not required to exist, but if they do exist, they shall conform to the provisions of this document.

Such obligation is enforced if and only if the Registration Status of the associated registry items is "Recorded" or higher (see 9.4.4.3 and ISO/IEC 11179-6:2023, 4.3.2).

4.6 Implementation conformance statement (ICS)

An implementation claiming conformance to this document shall include an Implementation Conformance Statement stating:

a) whether it conforms or strictly conforms;

b) which clauses are or are not supported;

c) what extensions, if any, are supported or used.

A standard profile may be referenced, if applicable.

EXAMPLE 1 Product X strictly conforms to ISO/IEC 11179-3:2023 Basic Registry, with the exception of Clause 10 which is not supported, and without extensions.

EXAMPLE 2 Product Y conforms to ISO/IEC 11179-31:2023 Data Specification Registry with mapping, except that it relies on the Item Mapping facility of ISO/IEC 11179-3:2023 in place of implementing the ISO/IEC 11179-31:2023 Data Element Derivation feature (ISO/IEC 11179-31:2023, 7.6.2.5, 7.6.2.6, 7.6.3.4, 7.6.3.5, 7.6.3.6).

4.7 Roles and responsibilities for registration

Conformance shall be considered in the context of the roles and responsibilities of registration authorities, in accordance with ISO/IEC 11179-6.

Extended conformance of systems requires formalisation of procedures, agreement of roles and responsibilities between parties, and guidelines addressing use of software products and conversions from other systems. The formalisation of these aspects shall be consistent with the conformance requirements in 4.2 through 4.6, and the roles of registration authorities in accordance with ISO/IEC 11179-6.

5 Approach to modelling a metadata registry

5.1 Metamodel for a metadata registry

A metamodel is a model that describes other models, e.g. a data model defining metadata that describes other data. A metamodel provides a mechanism for understanding the precise structure and

components of the specified models, which are needed for the successful sharing of the models by users, software facilities or both.

This document uses a metamodel to describe the information to be recorded in a metadata registry. The registry in turn will be used to describe and model other data, for example about enterprise, public administration or business applications. The registry metamodel is specified as a conceptual data model, i.e., one that describes how relevant information is structured in the natural world. In other words, it is how the human mind is accustomed to thinking of the information.

As a conceptual data model, there need be no one-to-one match between the attributes in the model and fields, columns, objects, et cetera in a database. There may be more than one field per attribute and some object classes, associations or both may be implemented as fields. There is no intent that an implementation should have a table for each association or object class. The metamodel need not be physically implemented as specified.

The structure described by this metamodel can be distributed over several implementations. These implementations can be databases, data repositories, metadata registers, metadata registries, dictionaries, etc. No particular technology is implied. Implementations may utilise technologies including, but not limited to: relational database, XML database, object-oriented systems or RDF/OWL.

The model shows constraints on minimum and maximum occurrences (the obligation) of attributes. The constraints on maximum occurrences are to be enforced at all times. The constraints on minimum occurrences are to be enforced when the registration status for the registry item is “Recorded” or higher. In other words, a registration status of “Recorded” indicates that all mandatory attributes have been documented.

5.2 Application of the metamodel

Some of the objectives of the metamodel for a metadata registry are:

— to provide a unified view of concepts, terms, value domains and value meanings;

— to promote a common understanding of the data described;

— to provide the specification at a conceptual level to facilitate the sharing and reuse of the contents of implementations.

A metamodel is necessary for coordination of data representation between persons, systems or both that store, manipulate and exchange data. The metamodel will assist registrars in maintaining consistency among different registries. The metamodel enables systems tools and information registries to store, manipulate and exchange the metadata for data attribution, classification, definition, naming, identification and registration. In this manner, consistency of data content supports interoperability among systems, tools and information registries.

Using the metamodel, mappings to the schema of particular metadata management tool sets can be developed. The metamodel constructs can be translated into the language of each tool set, preserving the concepts represented in the metamodel.

An implementer may use this conceptual data model to develop a more specific logical data model of the identical sphere of interest. A logical data model describes the same data, but as structured in an information system. It is often referred to as a Model of the Information System. A logical data model can be directly used for database design.

5.3 Specification of the metamodel

5.3.1 Terminology used in specifying the metamodel

When using a model to specify another model, it is easy for the reader to become confused about which model is being referred to at any particular point. To minimise this confusion, this document deliberately uses different terms in the model being specified from those used to do the specification.

The registry metamodel is specified using a subset of the Unified Modelling Language (UML) Version 2.4.1[26][27]. This document uses the term metamodel construct for the UML model constructs it uses, but metadata object for the model constructs it specifies. The metamodel constructs used are: classes, associations, association classes, attributes, composite attributes and composite datatypes, and these are defined in 3.1. The specified metadata objects are defined in Clauses 6 through 11. The concepts that the metadata objects represent are defined in 3.2.

5.3.2 Choice of fonts

In this Clause 5 through Clause 11 and Annexes B and C, this document uses:

— **bold** font to highlight terms which represent metadata objects specified by the metamodel;

— normal font for terms which represent concepts defined in Clause 3.

EXAMPLE **Concept** (6.4.2.2) is a class each instance of which models a concept.

5.3.3 Use of UML Packages

For descriptive and conformance purposes, the metamodel is organized into packages.

— Basic and Core package (see Clause 6) contains simple classes that are reused by other packages.

— Identification package (see Clause 7) contains classes that enable the contents of a registry to be identified.

— Designation and Definition package (see Clause 8) contains classes that enable the contents of a registry to be named, or otherwise designated, and defined.

— Registration package (see Clause 9) contains classes that enable registry items to be registered.

— Classification package (see Clause 10) contains classes that enable registry items to be classified in a classification scheme.

— Item mapping package (see Clause 11) contains classes that enable mappings among registry items.

5.3.4 Package dependencies

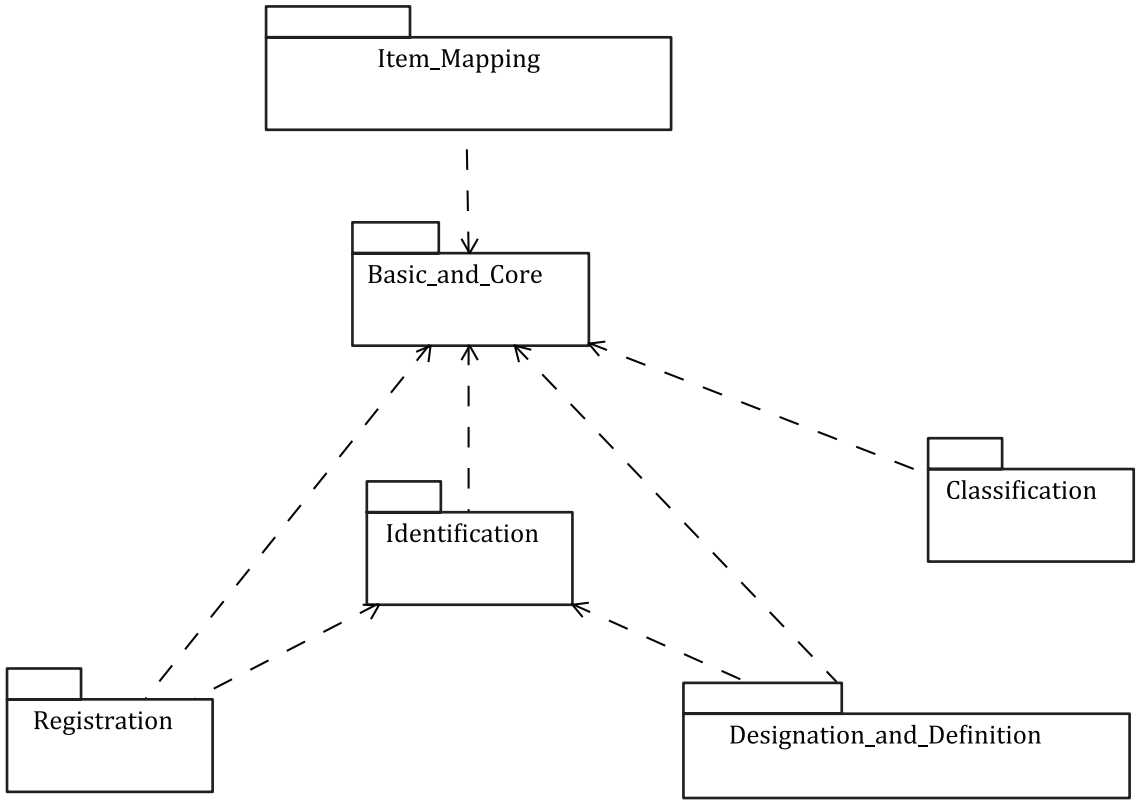

Figure 1 — Package dependencies

Figure 1 illustrates the dependencies among the packages. The lines in Figure 1 illustrate dependencies in the direction of the arrow. In order to implement a package that has dependencies, the packages on which it is dependent shall also be implemented.

Other parts extend the core model with additional packages supporting additional functionality:

— ISO/IEC 11179-31 contains classes that enable metadata objects to be described and registered;

— ISO/IEC 11179-32 contains classes that enable concepts and relations among concepts to be described and registered;

— ISO/IEC 11179-33 contains classes that enable data sets to be described and registered;

— ISO/IEC 11179-34 (under development) contains classes that enable computable data to be described and registered;

— ISO/IEC 11179-35 contains classes that enable models to be described and registered.

Conformance options are specified in Clause 4.

5.3.5 Use of UML Class diagrams and textual description

This document uses both text and UML class diagrams to describe the metamodel. Both are normative and are intended to be complementary. However, if a conflict exists between the text and the UML notation, the text takes precedence; if a conflict exists between a formal definition and other normative text, the formal definition takes precedence.

A consolidated UML class hierarchy is included as Annex A.

While the model diagrams are presented in UML notation, this document does not assume nor endorse any specific system environment, database management system, database design paradigm, system development methodology, data definition language, command language, system interface, user interface, computing platform or any technology required for implementation.

5.3.6 Use of dot notation

This document uses 'dot notation' when referring to an attribute within a class:

— **Classname.attributename**, e.g. **Designation.sign**.

5.4 Types, instances and values

When considering data and metadata, it is important to distinguish between types of data or metadata, and instances of these types and their associated values. The metamodel specifies types of classes, attributes and associations. Any particular instance of one of these will be of a specific type and, at any point in time, that instance will have a specific value (possibly null). As examples, this document defines "attribute instance" and "attribute value", but the same principle applies to classes, relationships and all other metamodel constructs defined in 3.1.

NOTE In UML, subclasses of a superclass are by default not disjoint. This document specifies when subclasses are required to be disjoint. Further, when the superclass is not intended to be instantiated alone, this document shows the superclass as abstract, thus preventing the superclass being instantiated other than through one or more of its subclasses. An abstract class is indicated by showing the class name in *italics* in any class diagram that uses it.

Clauses 6 through 11 specify the common facilities that form the core of a metadata registry and which apply to potentially any type of registry content. Other parts of ISO/IEC 11179 (e.g. ISO/IEC 11179.31, ISO/IEC 11179-32, ISO/IEC 11179-33, ISO/IEC 11179-34 and ISO/IEC 11179-35) and ISO/IEC 19763 (e.g. ISO/IEC 19763-3, ISO/IEC 19763-5, ISO/IEC 19763-7, ISO/IEC 19763-8, ISO/IEC 19763-10, ISO/IEC 19763-12, ISO/IEC 19763-13 and ISO/IEC 19763-16) specify the types of metadata objects that form the potential content of a metadata registry. A metadata registry will be populated with instances of these metadata objects (referred to as "metadata items"), which in turn define types of data, e.g. in an application database. In other words, instances of metadata specify types of application-level data. In turn, the application database will be populated by the real-world data as instances of those defined metadata object types.

NOTE ISO/IEC TR 19583-1[28] explains the concepts of various levels of modelling.

5.5 Extensibility

It is not expected that this metamodel will completely accommodate all users. Particular sectors, such as document management, scientific data and statistical data, require metadata attributes not addressed in this standard. This standard provides a **Slot** class (6.4.2.4) as a mechanism to extend registry items with custom attributes. Classes, relationships, attributes, datatypes or any combination thereof may be added as extensions to existing packages in this conceptual data model, or completely new packages may be added.

Implementers of this document may include extensions as part of an implementation, or they may provide facilities to enable a registry user to define their own extensions, such as classes, packages, etc. or both. An implementation with such extensions shall be considered conformant if it does not violate any of the rules inherent in the structure and content as specified by the metamodel in this document.

5.6 Date references

In this document, dates are important attributes of administered items, represented by instances of the **Administered_Item** class (9.4.2), and of operations of a registry. For the purpose of this document, "date" refers to Gregorian calendar date (see ISO 8601[9]). (See also 6.2.4 for the specification of the associated datatypes.)

6 Basic_and_Core package

6.1 Overview of Basic_and_Core package

The Basic_and_Core package is partitioned into three regions:

— "Predefined Types" which specifies common datatypes for use elsewhere in the metamodel (6.2);

— "Basic Classes" which specifies common classes for use elsewhere in the metamodel (6.3);

— "Core metamodel" which specifies features that are fundamental to a metadata registry, or which can be referenced from multiple other metamodel regions, or both (6.4).

6.2 Predefined types metamodel region

6.2.1 Overview of predefined types

The predefined types metamodel region specifies the datatypes used in the specification of the metamodel. A datatype is a set of distinct values, characterized by properties of those values and by operations on those values (ISO/IEC 11404). All of the other types used in the model are based on this core set of types and any compliant implementation of a metadata registry should include an implementation of the semantics specified in these core types.

NOTE The datatypes that are described in this subclause are used in specification of the metamodel itself and are not intended to constrain the datatypes that can be used in the content of a registry.

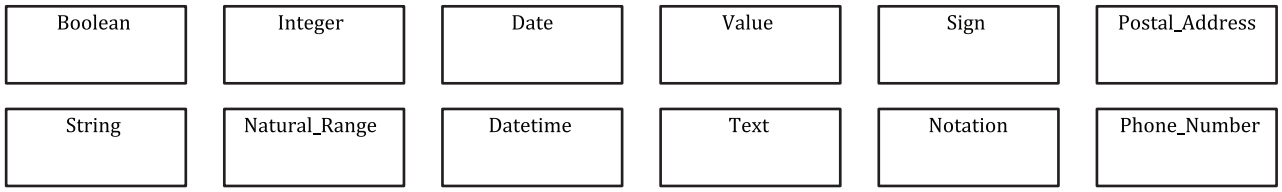

Figure 2 — Predefined types metamodel region

6.2.2 Boolean

Boolean is the mathematical datatype associated with two-valued logic.

The notation and semantics for Boolean shall be in accordance with ISO/IEC 11404:2007, 8.1.1 Boolean.

6.2.3 Datetime

Datetime is a datatype whose values are points in time to the resolution: year, month, day, hour, minute, second and optionally fractions of seconds.

The notation and semantics for **Datetime** shall be in accordance with ISO/IEC 11404:2007, 8.1.6 Date-and-Time, with the addition of a timezone specification as per ISO 8601 basic format.

6.2.4 Date

Date is a datatype whose values are points in time to the resolution: year, month and day.

The notation and semantics for **Date** shall be in accordance with ISO/IEC 11404:2007, 8.1.6 Date-and-Time.

6.2.5 Integer

Integer is a mathematical datatype comprising the exact integral values.

The notation and semantics for **Integer** shall be in accordance with ISO/IEC 11404:2007, 8.1.7 Integer.

6.2.6 Natural_Range

Natural_Range is a datatype comprising a range of "natural numbers", i.e. the positive integers, including zero. Any instance of **Natural_Range** is one of:

— a constant non-negative integer;

— a bounded range of non-negative integers defined by a minimum and a (strictly larger) maximum value;

— an unbounded range defined by only a non-negative minimum (e.g. 0..*, 1..*, 2..*).

6.2.7 Notation

Notation is a datatype that denotes a notation defined elsewhere but used by an item within the registry. A notation defines a formal syntax and semantics, meant for machine processing. In this metamodel, **Notation** is used by **Reference_Document** (6.3.8).

EXAMPLES XCL Common Logic (ISO/IEC 24707[31]) or OWL-DL XML notation.

6.2.8 Phone_Number

Phone_Number is a datatype that denotes a phone number. A phone number uniquely identifies a telephone line within a telephone network. The data structure of the **Phone_Number** datatype shall conform to ITU-T Recommendation E 164[35] and may conform to ISO/IEC 19773[30] Module 17: Data structure for ITU-T E.164 phone number data.

NOTE ISO/IEC 19773 is referenced but not required.

6.2.9 Postal_Address

Postal_Address is a datatype that denotes a postal address. A postal address enables the unambiguous determination of an actual or potential delivery point, usually combined with the specification of an addressee or a mailee. The data structure of **Postal_Address** may conform to ISO/IEC 19773[30] Module 16: Data Structure for UPU postal data[38].

NOTE ISO/IEC 19773 is referenced but not required.

6.2.10 Sign

Sign is a datatype that denotes a sign. A sign can be a character string, graphic image, sound clip or other symbol that can be used to denote or designate a concept. The **Sign** datatype can be implemented using the Reflit(of_type) data structure of ISO/IEC 19773:2011[30], 10.4.2, where the list of supported types is implementation defined. At a minimum, datatype **String** (6.2.11) shall be supported.

NOTE ISO/IEC 19773 is referenced but not required.

6.2.11 String

String is a family of datatypes which represent strings of symbols from standard character-sets. The syntax and semantics of the **String** datatype are as defined in ISO/IEC 11404:2007, 10.1.5 Character String.

6.2.12 Text

Text is a datatype that denotes text, data in the form of characters, symbols, words, phrases, paragraphs, sentences, tables, or other character arrangements, intended to convey a meaning, and whose interpretation is essentially based upon the reader's knowledge of some natural language or artificial language.

Where multilingual text is required, the **Text** datatype can be implemented using the multitext data structure of ISO/IEC 19773:2011, 12.3.3[30].

EXAMPLE A business letter printed on paper or displayed on a screen.

NOTE ISO/IEC 19773 is referenced but not required.

6.2.13 Value

Value is a datatype that represents any instance of any other datatype.

6.3 Basic classes metamodel region

6.3.1 Overview of basic classes

The basic classes metamodel region specifies classes which are used as datatypes within the metamodel. See Figure 3.

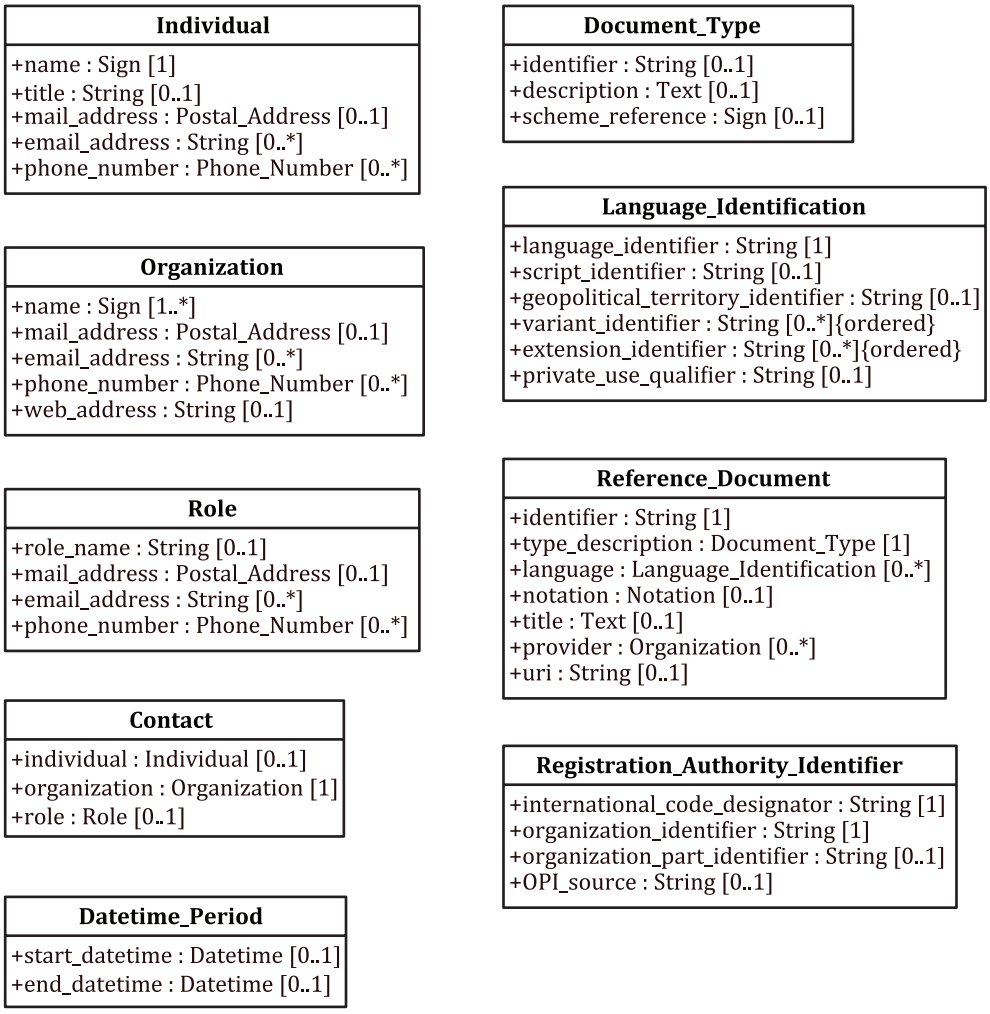

Figure 3 — Basic classes metamodel region

6.3.2 Individual class

6.3.2.1 Description of Individual

Individual is a class each instance of which models an individual.

6.3.2.2 Attributes of Individual

See Table 1.

Table 1 — Attributes of Individual class

Attribute name	Multiplicity	Datatype	Description
name	1	**Sign** (6.2.10)	Definition: sign that designates the individual.
title	0..1	**String** (6.2.11)	Definition: name of the position held by the individual (e.g. database administrator)
mail_address	0..1	**Postal_Address** (6.2.9)	Definition: postal address for the individual
email_address	0..*	**String** (6.2.11)	Definition: email addresses of the individual
phone_number	0..*	**Phone_Number** (6.2.8)	Definition: phone numbers for the individual

6.3.3 Organization class

6.3.3.1 Description of Organization

Organization is a class each instance of which models an organization.

Organization is the superclass of **Registration_Authority** (9.4.6) in the Registration package.

6.3.3.2 Attributes of Organization

See Table 2.

Table 2 — Attributes of Organization class

Attribute name	Multiplicity	Datatype	Description
name	1..*	**Sign** (6.2.10)	Definition: sign that designates the organization.
mail_address	0..1	**Postal_Address** (6.2.9)	Definition: postal address for the organization.
email_address	0..*	**String** (6.2.11)	Definition: contact email addresses of the organization that are not tied to a specific individual or role.
phone_number	0..*	**Phone_Number** (6.2.8)	Definition: phone numbers for the organization.
web_address	0..1	**String** (6.2.11)	Definition: URL for the organization's website.

6.3.4 Role class

6.3.4.1 Description of Role

Role is a class each instance of which models a role which an individual can play as a contact within an organization.

6.3.4.2 Attributes of Role

See Table 3.

Table 3 — Attributes of Role class

Attribute name	Multiplicity	Datatype	Description
role_name	0..1	**String** (6.2.11)	Definition: name of the position that fulfils the role
mail_address	0..1	**Postal_Address** (6.2.9)	Definition: postal address for the role
email_address	0..*	**String** (6.2.11)	Definition: email addresses of the role
phone_number	0..*	**Phone_Number** (6.2.8)	Definition: phone numbers for the role

6.3.5 Contact class

6.3.5.1 Description of Contact

Contact is a class each instance of which models a contact, which specifies a role, an individual or both within an organization to whom information item(s), material object(s), person(s) or some combination can be sent to or from. **Registrar** (9.4.7) is a subclass of **Contact**.

6.3.5.2 Attributes of Contact

See Table 4.

Table 4 — Attributes of Contact class

Attribute name	Multiplicity	Datatype	Description
organization	1	**Organization** (6.3.3)	Definition: organisation for which the contact, represented by this instance of the **Contact** class, acts as a representative.
individual	0..1	**Individual** (6.3.2)	Definition: individual that is the contact, represented by this instance of the **Contact** class, within the organisation.
role	0..1	**Role** (6.3.4)	Definition: role undertaken by the contact, represented by this instance of the **Contact** class, within the organisation.

6.3.5.3 Constraints on Contact

For each instance of the **Contact** class there shall be a value specified for the **individual** attribute of type **Individual**, or there shall be a value specified for the **role** attribute of type **Role**, or there shall be values specified for both these attributes.

6.3.6 Document_Type class

6.3.6.1 Description of Document_Type

Document_Type is a class used to specify the document type of a **Reference_Document** (6.3.8). The document type can be specified using an **identifier** or a **description**.

6.3.6.2 Attributes of Document_Type

See Table 5.

Table 5 — Attributes of Document_Type class

Attribute name	Multiplicity	Datatype	Description
identifier	0..1	**String** (6.2.11)	Definition: identifies the type of document.
description	0..1	**Text** *(6.2.12)*	Definition: describes the type of document.
scheme_reference	0..1	**Sign** (6.2.10)	Definition: identification scheme from which the identifier is drawn.

6.3.6.3 Constraints on Document_Type

6.3.6.3.1 Constraint 1

For each instance of the **Document_Type** class there shall be a value specified for the **identifier** attribute of type **String**, or there shall be a value specified for the **description** attribute of type **Text**, or there shall be values specified for both these attributes.

6.3.6.3.2 Constraint 2

For each instance of the **Document_Type** class, a value shall be specified for the **scheme_reference** attribute only if a value is specified for the **identifier** attribute in the same instance.

6.3.7 Language_Identification class

6.3.7.1 Description of Language_Identification

Language_Identification is a class which serves as an identifier for a language. Each instance of the **Language_Identification** class represents a language as spoken (or written, signed or otherwise signalled) by human beings for communication of information to other human beings. Computer languages such as programming languages are explicitly excluded.

The identifier is comprised of the following parts, or attributes, which are based on IETF RFC 5646[33]. IETF RFC 5646 refers to these attributes as 'language subtags':

NOTE 1 The W3C has a description of the use of the IETF language subtags at: http://www.w3.org/International/articles/language-tags/Overview.en.php.

NOTE 2 IANA maintains a registry of language subtags at: http://www.iana.org/assignments/language-subtag-registry.

NOTE 3 RFC 5646 requires the extension identifiers to be prefixed by a single character that identifies the registration authority that has registered the extension.

6.3.7.2 Attributes of Language_Identification

See Table 6.

Table 6 — Attributes of Language_Identification class

Attribute name	Multiplicity	Datatype	Description
language_identifier	1	**String** (6.2.11)	Definition: identifier for the language Recommendation: Use of the three-character alphabetic codes from ISO 639-2/Terminology[1], with extensions if needed, is recommended but not required.

Table 6 *(continued)*

Attribute name	Multiplicity	Datatype	Description
script_identifier	0..1	**String** (6.2.11)	Definition: identifier for the set of graphic characters used for the written form of one or more languages Recommendation: Use of the four-character codes from ISO 15924[23] codes for the representation of the names of scripts is recommended but not required.
geopolitical_territory_identifier	0..1	**String** (6.2.11)	Definition: identifier for a specific country, territory or region whose linguistic variations apply Recommendation: Use of the three-digit numeric codes from ISO 3166-1[6], with extensions if needed, is recommended but not required.
variant_identifier	0..*	**String** (6.2.11) [ordered]	Definition: identifier for a language variant, which indicates additional, well-recognized variations that define a language or its dialects that are not covered by other available identifiers Recommendation: Use of RFC 5646[33] is recommended but not required. In RFC 5646, variant identifiers are typically represented as dates and are used to distinguish events such as spelling reforms. Variant identifiers can be order dependent. String Numeric variant identifiers are interpreted to be Gregorian calendar year numbers. Alphanumeric tags reference IANA variant subtags.
extension_identifier	0..*	**String** (6.2.11) [ordered]	Definition: identifier for an extension to a language_identifier Recommendation: Use of RFC 5646[33] is recommended but not required. Constraint: In RFC 5646, extension identifiers are ordered and consist of key-value pairs, separated by the EQUALS SIGN (=). The values shall be alphanumeric with no embedded whitespace. Whitespace separates the identifiers. NOTE The u- extension is defined in RFC 6067[34], which points to the Unicode Consortium's Common Locale Data Repository (CLDR) for details on the subtags that follow it.
private_use_qualifer	0..1	**String** (6.2.11)	Definition: qualifier whose meaning is defined solely by private agreement. Recommendation: Use of RFC 5646[33] is recommended but not required. NOTE The definition has been derived from IETF RFC 5646.

6.3.8 Reference_Document class

6.3.8.1 Description of Reference_Document

Reference_Document is a class each instance of which models a reference document that provides pertinent details for consultation about a subject.

6.3.8.2 Attributes of Reference_Document

See Table 7.

Table 7 — Attributes of Reference_Document class

Attribute name	Multiplicity	Datatype	Description
identifier	1	**String** (6.2.11)	Definition: identifier for the reference document
type_description	1	**Document_Type** (6.3.6)	Definition: description of the type of reference document
language	0..*	**Language_Identification** (6.3.7)	Definition: identifier of the natural language used in the reference document Conditions: (1) If a value has been specified for the **default_language** attribute of the **Registry_Specification** class (9.4.10) the absence of a value for this attribute implies that the language for this reference document is the language specified by the **default_language** attribute of the **Registry_Specification** class. (2) If no value has been specified for the **default_language** attribute of the **Registry_Specification** class, then a value for this attribute shall be specified.
notation	0..1	**Notation** (6.2.7)	Definition: notation used within the reference document
title	0..1	**Text** (6.2.12)	Definition: title of the reference document.
provider	0..*	**Organization** (6.3.3)	Definition: organization that maintains or carries an official copy of the reference document.
uri	0..1	**String** (6.2.11)	Definition: URI where the reference document is available.

6.3.9 Registration_Authority_Identifier class

6.3.9.1 Description of Registration_Authority_Identifier

The **Registration_Authority_Identifier** is a class used to uniquely identify a **Registration_Authority** (9.4.6). The sources of values for each part of the identifier are specified in ISO/IEC 6523-1[8] and further explained for the metadata registry in ISO/IEC 11179-6.

6.3.9.2 Attributes of Registration_Authority_Identifier

See Table 8.

Table 8 — Attributes of Registration_Authority_Identifier class

Attribute name	Multiplicity	Datatype	Description
international_code_designator	1	**String** (6.2.11)	Definition: identifier of the organization identification scheme used in the identification of the organization providing this registration authority
organization_identifier	1	**String** (6.2.11)	Definition: identifier within the organization identification scheme used in the identification of the organization providing this registration authority that uniquely identifies this organization within this organization identification scheme.
organization_part_identifier	0..1	**String** (6.2.11)	Definition: identifier allocated to the organization part providing this registration authority within this organization organization part identifier is often abbreviated as OPI.
OPI_source	0..1	**String** (6.2.11)	Definition: source for the **organization_part_identifier** Condition: If **organization_part_identifier** is present, then **OPI_source** shall be present.

6.3.10 Datetime_Period class

6.3.10.1 Description of Datetime_Period

Datetime_Period is a class each instance of which models a datetime period, a combination of an optional start datetime and an optional end datetime.

6.3.10.2 Attributes of Datetime_Period

See Table 9.

Table 9 — Attributes of Datetime_Period class

Attribute name	Multiplicity	Datatype	Description
start_datetime	0..1	**Datetime** (6.2.3)	Definition: the datetime that the datetime period starts, if known.
end_datetime	0..1	**Datetime** (6.2.3)	Definition: the datetime that the datetime period ends, if known.

6.3.10.3 Constraint on Datetime_Period

At least one of the attributes shall be specified.

6.4 Core metamodel region

6.4.1 Overview of Core metamodel region

The Core metamodel region specifies features that are fundamental to a metadata registry, or which can be referenced from multiple other metamodel regions, or both. Figure 4 shows the Core metamodel region.

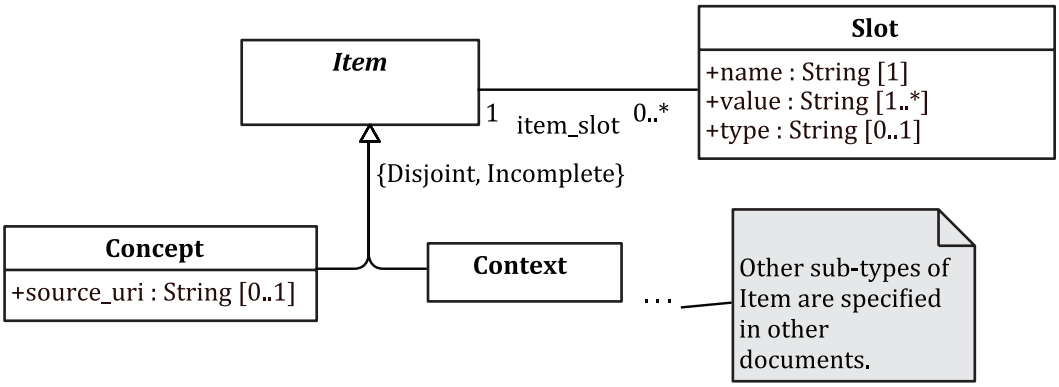

Figure 4 — Core metamodel region

6.4.2 Classes in the Core metamodel region

6.4.2.1 Item class

6.4.2.1.1 Description of Item

Item is an abstract superclass, each instance of which models a registry item, but which, as an abstract class, shall also be instantiated as a specific subclass [e.g. **Concept** (6.4.2.2), **Data_Element** (ISO/IEC 11179-31:2023, 7.6.2.1), etc.]. **Item** provides the basis for all registry items entered into a metadata registry that conforms to the specifications contained in the appropriate parts of ISO/IEC 11179. Additional facilities defined later allow any **Item** to be: identified (see Clause 7), named and defined (see Clause 8), registered and administered (see Clause 9), classified (see Clause 10) and mapped to other registry items (see Clause 11).

6.4.2.1.2 Associations of Item

This metamodel region specifies the following associations:

— **item_slot** (6.4.3.1).

Other associations are added in other metamodel regions where the **Item** class is extended:

— the Identification metamodel region (7.2.1.2);

— the Designation and Definition metamodel region (8.2.1.2);

— the Classification metamodel region (10.2.1.2);

— the Item Mapping metamodel region (11.2.1.2);

— and in ISO/IEC 11179-31[17], ISO/IEC 11179-32[18], ISO/IEC 11179-33[19] and ISO/IEC 11179-35[21].

6.4.2.1.3 Attributes of Item

No attributes are specified in this metamodel region. Attributes are defined by most subclasses specified in other metamodel regions specified in this document and in other parts of ISO/IEC 11179.

6.4.2.2 Concept class

6.4.2.2.1 Direct superclass

Concept is a subclass of **Item** (6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

6.4.2.2.2 Description of Concept

Concept is a class each instance of which models a concept, a unit of knowledge created by a unique combination of characteristics. A concept is independent of representation.

NOTE **Concept** is included in the Core metamodel so it can be used in other metamodel regions without creating dependencies among them, e.g. the Data Specification region in ISO/IEC 11179-31[17], the Concept System region in ISO/IEC 11179-32[18], the Data Set region in ISO/IEC 11179-33[19], and the Model region in ISO/IEC 11179-35[21].

6.4.2.2.3 Associations of Concept

As a subclass of **Item**, **Concept** inherits **Item**'s associations (6.4.2.1.2). Associations are added in other metamodel regions where the class is extended, e.g. 8.2.2.3, and in ISO/IEC 11179-31[17], ISO/IEC 11179-32[18], ISO/IEC 11179-33[19] and ISO/IEC 11179-35[21].

6.4.2.2.4 Attributes of Concept

See Table 10.

Table 10 — Attributes of Concept class

Attribute name	Multiplicity	Datatype	Description
source_uri	0..1	**String** (6.2.11)	Definition: URI that enables access to the concept system which includes this concept To be used when the **Concept_System** is external to the registry. If the **Concept_System** is in the registry, use the association **concept_source** (ISO/IEC 11179-32:2023, 7.1.3.2) instead.

6.4.2.3 Context class

6.4.2.3.1 Direct superclass

Context is a subclass of **Item** (6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

6.4.2.3.2 Description of Context

Context is a class each instance of which models a context, which is used to constrain the applicability of some other registry item.

NOTE 1 **Context** is included in the Core metamodel so it can be used by multiple packages without creating a dependency among them. **Context** is currently used by the Designation and Definition region (Clause 8), the Data Specification region in ISO/IEC 11179-31[17] and the Data Set region in ISO/IEC 11179-33[19].

NOTE 2 **Context** is further described in 8.2.2.

6.4.2.3.3 Associations of Context

As a subclass of **Item**, **Context** inherits **Item**'s associations (6.4.2.1.2). Associations are added in other model regions where the class is extended, e.g. 8.2.2.3 and in ISO/IEC 11179-31[17], ISO/IEC 11179-32[18], ISO/IEC 11179-33[19] and ISO/IEC 11179-35[21].

6.4.2.3.4 Attributes of Context

No attributes are specified in this metamodel region.

6.4.2.4 Slot class

6.4.2.4.1 Description of Slot

Slot is a class each instance of which models a slot, a container for an extension to a registry item. Instances of the **Slot** class provide a dynamic way to add arbitrary attributes to instances of **Item** (6.4.2.1).

EXAMPLE If a company wants to add a "copyright" attribute to each **Item** instance that it submits, it can do so by adding a slot with name "copyright" and value containing the copyrights statement.

NOTE **Slot** is modelled after the Slot class of ebXML RIM[36], but it also maps directly to the 'slot tuple' of ISO/IEC 19773:2011[30] Module 13, where:

— **Slot.name** maps to slot_tuple.identifier;

— **Slot.type** maps to slot_tuple.kind;

— **Slot.value** maps to slot_tuple.value.

6.4.2.4.2 Associations of Slot

See **item_slot** association (6.4.3.1).

6.4.2.4.3 Attributes of Slot

See Table 11.

Table 11 — Attributes of Slot class

Attribute name	Multiplicity	Datatype	Description
name	1	**String** (6.2.11)	Definition: name of the slot Constraint: All **Slot** instances associated with a particular **Item** instance shall have a distinct name to allow each **Slot** instance to be unambiguously identified.
Value	1..*	**String** (6.2.11)	Definition: value assigned to the slot
type	0..1	**String** (6.2.11)	Definition: categorization of the value of the slot **type** can be used to categorize slot values in some way, including but not limited to specifying the datatype of the value.

6.4.3 Associations in the Core metamodel region

6.4.3.1 item_slot

The **item_slot** association records the binding of exactly one instance of the **Item** (6.4.2.1) class with zero, one or more instances of the **Slot** (6.4.2.4) class.

This association registers the slots that are used to extend the registry item.

7 Identification package

7.1 Overview of Identification metamodel region

The Identification package consists of one region, the Identification metamodel region, which specifies how registry items are identified in the metadata registry.

NOTE A description of the distinction between identification and designation is available in ISO/IEC 15944-1:2011[24], C.6.

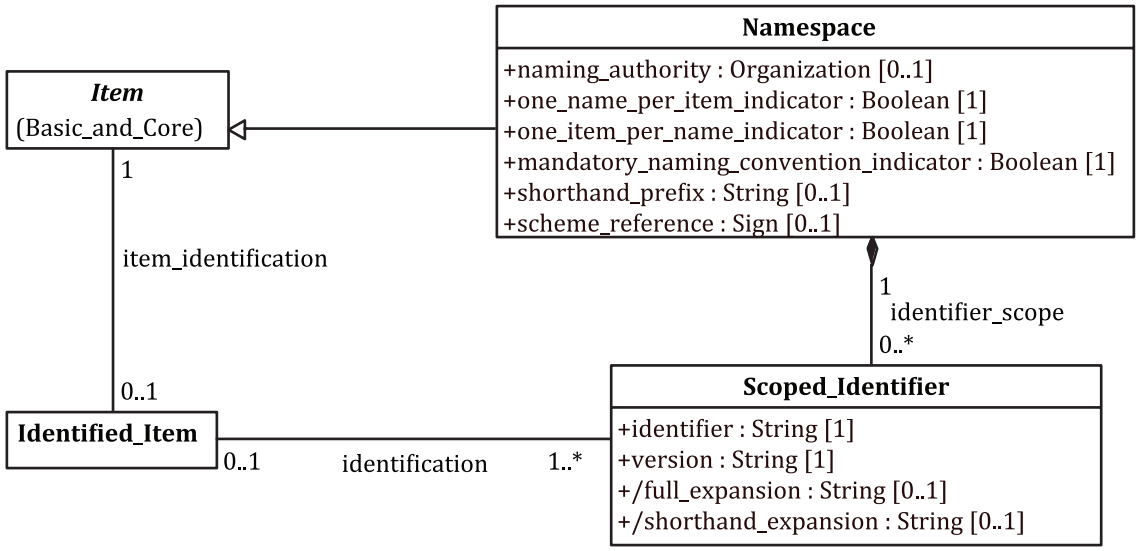

Figure 5 — Identification metamodel region

Identified_Item (7.3.1) is made a separate class, so it can subsequently be subclassed for **Registered_Items** (9.4.1), thus ensuring that **Registered_Items** shall be identified.

7.2 Classes referenced from the Basic_and_Core package

7.2.1 Item class

7.2.1.1 Description of Item

Item is part of the Core metamodel region and is specified in 6.4.2.1. Additional associations and subclasses are specified in this metamodel region.

7.2.1.2 Associations of Item

This metamodel region specifies the following associations:

— **item_identification** (7.4.1).

Other associations are added in other metamodel regions where the **Item** class is extended:

— the Core metamodel region (6.4.2.1.2);

— the Designation and Definition metamodel region (8.2.1.2);

— the Classification metamodel region (10.2.1.2);

— the Item Mapping metamodel region (11.2.1.2);

— and in ISO/IEC 11179-31[17], ISO/IEC 11179-32[18], ISO/IEC 11179-33[19], ISO/IEC 11179-34[20] and ISO/IEC 11179-35[21].

7.2.1.3 Attributes of Item

No attributes are specified in this metamodel region.

7.3 Classes in the Identification metamodel region

7.3.1 Identified_Item class

7.3.1.1 Description of Identified_Item

Identified_Item is a class each instance of which models an identified item, the identification of a registry item in a registry.

Registered_Item (9.4.1) is a subclass of **Identified_Item**.

7.3.1.2 Associations of Identified_Item

Identified_Item has the following associations:

— **item_identification** (7.4.1);

— **identification** (7.4.2).

7.3.1.3 Attributes of Identified_Item

No attributes are specified in this metamodel region.

7.3.2 Scoped_Identifier class

7.3.2.1 Description of Scoped_Identifier

Scoped_Identifier is a class each instance of which models a scoped identifier, an identifier with a particular scope provided by a namespace.

7.3.2.2 Associations of Scoped_Identifier

Scoped_Identifier has the following associations:

— **identification** (7.4.1);

— **identifier_scope** (7.4.3).

7.3.2.3 Attributes of Scoped_Identifier

See Table 12.

Table 12 — Attributes of Scoped_Identifier class

Attribute name	Multiplicity	Datatype	Description
identifier	1	**String** (6.2.11)	Definition: string used to unambiguously identify this scoped identifier which has been used to denote the linked identified item within the scope of the linked namespace.

Table 12 *(continued)*

Attribute name	Multiplicity	Datatype	Description
Version	1	**String** (6.2.11)	Definition: string used to uniquely identify the version of this scoped identifier which has been used to identify the linked identified item.
Full_expansion	0..1	**String** (6.2.11)	Conditional, derived. Definition: string providing a representation of this scoped identifier, in which the unique identifier of the linked namespace is combined in some way with the identifier of this scoped identifier to fully specify the scope. Condition: **full_expansion** is defined only when **Namespace** has exactly one identifier. Derivation: The precise manner of derivation can vary depending on the type of namespace.
Shorthand_expansion	0..1	**String** (6.2.11)	Conditional, derived. Definition: string providing a representation of this scoped identifier in which the shorthand prefix of the linked namespace, represented by the value assigned to the **Namespace.shorthand_prefix** attribute, is prepended to the identifier of this scoped identifier to indicate the scope. Condition: **shorthand_expansion** shall exist if and only if the corresponding **shorthand_prefix** exists. Derivation: The precise manner of derivation can vary depending on the type of namespace.

7.3.2.4 Constraint on Scoped_Identifier

If an instance of the **Scoped_Identifier** class is linked via the **identifier_scope** (7.4.3) association to an instance of the **Namespace** (7.3.3) class which is subsequently deleted, then the instance of **Scoped_Identifier** shall also be deleted.

7.3.3 Namespace class

7.3.3.1 Direct superclass

Namespace is a subclass of **Item** (6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

7.3.3.2 Description of Namespace

Namespace is a class each instance of which represents a namespace. A namespace is a scoping construct used to group sets of designations, represented by instances of the **Designation** class (8.4.1), scoped identifiers, represented by instances of the **Scoped_Identifier** class (7.3.2), or both, used in a metadata registry. Distinct namespaces permit independent development of metadata collections and ontologies. They permit enforcement of uniqueness constraints on the identifiers of scoped identifiers or signs of designations within a specific namespace without central coordination.

A namespace may contain a set of signs for designations, a set of identifiers for scoped identifiers or a combination of both.

NOTE 1 The term “namespace” is used in this document because it is in common use, even though the concept is applied to identifiers as well as names.

NOTE 2 These are not XML Namespaces. However, it is possible to add additional subclasses of **Namespace** to model XML Namespaces.

NOTE 3 See also 8.3.1 for further description of the **Namespace** class in the Designation and Definition metamodel region, and 9.3.1 for further description of the **Namespace** class in the Registration metamodel region.

7.3.3.3 Associations of Namespace

As a subclass of **Item** (7.3.1), **Namespace** inherits **Item**'s associations (7.2.1.2). This metamodel region specifies the following association:

— **identifier_scope** (7.4.3).

Additional associations of **Namespace** are specified as part of the Designation and Definition metamodel region (8.3.1.3) and as part of the Registration metamodel region (9.3.2.3).

7.3.3.4 Attributes of Namespace

See Table 13.

Table 13 — Attributes of Namespace class

Attribute name	Multiplicity	Datatype	Description
naming_authority	0..1	**Organization** (6.3.3)	Definition: organization that has the authority for naming in this namespace.
one_name_per_item_ indicator	1	**Boolean** (6.2.2)	Definition: indicator that denotes whether more than one designation or scoped identifier within this namespace may be associated with any single registry item, represented by an instance of the **Item** class (6.4.2.1). If the indicator is TRUE, then at most one designation or scoped identifier within this namespace may be associated with any single registry item. Rule: If the indicator is TRUE, then the registry shall enforce the rule for the namespace.
one_item_per_name_ indicator	1	**Boolean** (6.2.2)	Definition: indicator that denotes whether this namespace may contain more than one designation or scoped identifier having the same sign or identifier. If the indicator is TRUE, then at most one designation or scoped identifier having the same sign or identifier is permitted within the namespace. Rule: If the indicator is TRUE, then the registry shall enforce the rule for the namespace. Constraints: The metamodel constrains each scoped identifier and designation to be associated with only one item. This indicator can be used to require that the value of the associated sign be unique within the namespace.

Table 13 *(continued)*

Attribute name	Multiplicity	Datatype	Description
mandatory_naming_ convention_indicator	1	**Boolean** (6.2.2)	Definition: indicator specifying whether all designations in this namespace shall conform to a naming convention. Constraint: If the value of this attribute is TRUE: (a) there shall be at least one instance of the **Naming_Convention** class (8.4.4) linked to this instance of the **Namespace** class (7.3.3) via the **naming_convention_utilization** association (8.6.6); (b) every instance of the **Designation** class (8.4.1) shall be linked via the **naming_ convention_conformance** association (8.6.5) with one of the same instances of the **Naming_Convention** class used in part (a) above. If the value of this attribute is FALSE, it is possible for a namespace to be associated with zero, one or more naming conventions, for a designation to conform to more than one convention, or both.
shorthand_prefix	0..1	**String** (6.2.11)	Definition: prefix conventionally used as shorthand for a namespace, for greater readability, in text for human consumption. Rule: In the case of URL prefixes as defined in XML, a final colon (:) should be included here as part of the shorthand prefix.
scheme_reference	0..1	**Sign** (6.2.10)	Definition: Internationalized Resource Identifier (IRI) identifying the type of the specification for this namespace. EXAMPLE 1: For XML Namespaces, specify: http://www.w3.org/TR/1999/REC-xml-names-19990114/ EXAMPLE 2: For UML Namespaces, specify: http://www.omg.org/spec/UML/2.4.1/Core/Namespace EXAMPLE 3: For Digital Object Identifiers use the form: doi: https://www.doi.org/10.1000/182[32]

7.4 Associations in the Identification metamodel region

7.4.1 item_identification association

The **item_identification** association records the binding of zero or one instance of the **Identified_Item** (7.3.1) class with the instance of the **Item** (7.2.1) class that is to be identified. The association records the identified item that provides identification for a registry item.

If an instance of the **Item** (7.2.1) class is to be identified, then it shall be linked with exactly one instance of the **Identified_Item** (7.3.1) class through an instance the **item_identification** association.

NOTE The association is optional, allowing an item to be entered into a registry without being identified, but the use case for this scenario is not obvious. Best practice is to always provide a unique identifier for an item.

7.4.2 identification association

The **identification** association records the binding of one or more instances of the **Scoped_Identifier** (7.3.2) class with zero or one instance of the **Identified_Item** (7.3.1) class that they identify. The association records all the scoped identifiers for an identified item.

Every instance of the **Identified_Item** class shall have links via one or more instances of the **identification** association with the instance of the **Scoped_identifier** class that provides an **identifier** for the instance of the **Identified_Item** class.

7.4.3 identifier_scope association

The **identifier_scope** association records the binding of zero, one or more instances of the **Scoped_Identifier** (7.3.2) class with exactly one instance of the **Namespace** (7.3.3) class which provides the scope for the instance of the **Scoped_Identifier**. The association records the namespace that provides the scope for the scoped identifiers.

NOTE See Constraint on **Scoped_Identifier** (7.3.2.4).

8 Designation_and_Definition package

8.1 Overview of Designation and Definition metamodel region

The Designation_and_Definition package consists of one region, the Designation and Definition metamodel region which contains classes that are used to manage the designations and definitions of registry items and the contexts for the designations and definitions. The relevant classes are **Designation** (8.4.1), **Definition** (8.4.2), **Item** (8.2.1) and **Context** (8.2.2). A registry item may have many designations with different signs that will vary depending on discipline, locality, technology, etc. The following subclauses describe the classes, associations and association classes of this region.

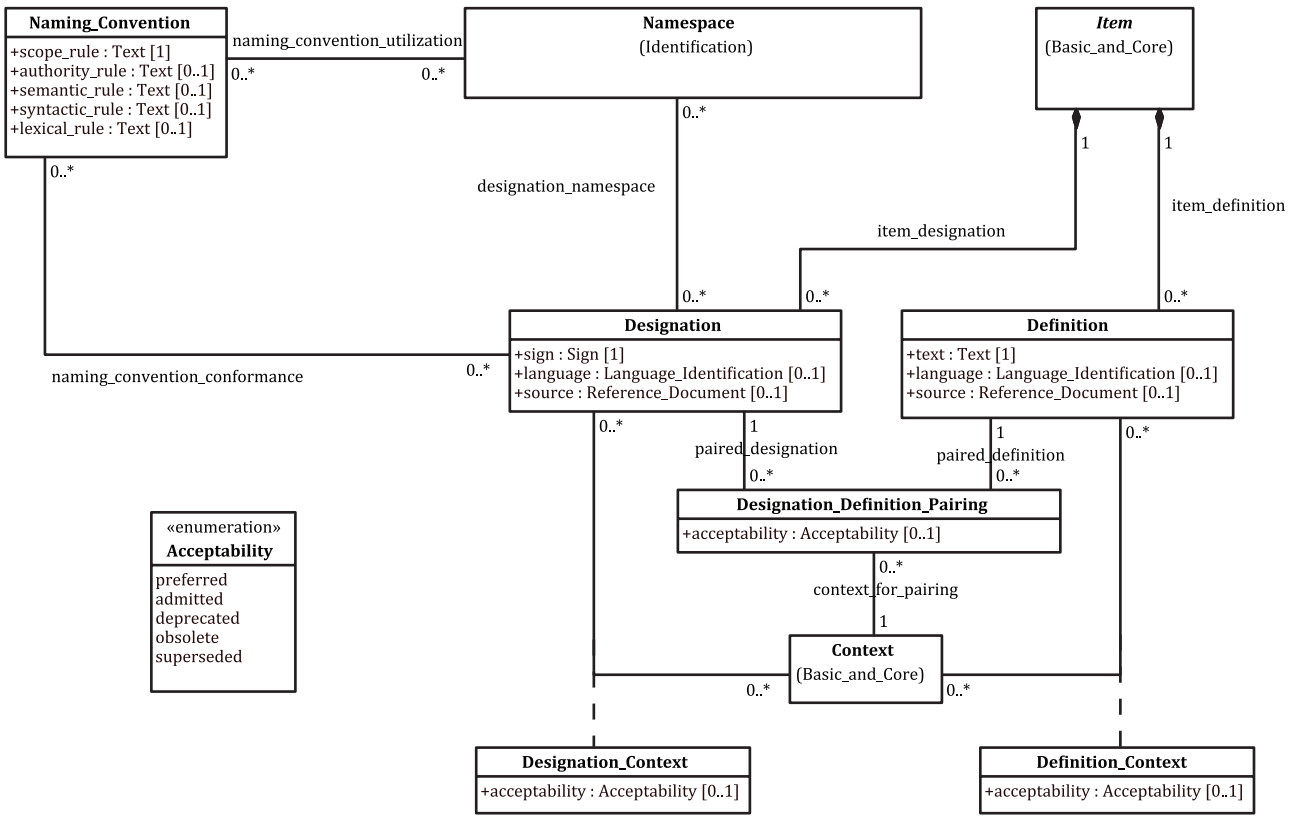

Figure 6 — Designation and Definition metamodel region

Figure 6 represents the Designation and Definition region. This region of the metamodel is based on, and is consistent with, terminological models[2][1] developed by ISO/TC 37 Language and terminology.

ISO/IEC 11179-4[14] provides rules and guidelines for the formulation of data definitions.

ISO/IEC 11179-5[15] provides naming and identification principles for **Item**s within a **Context**.

NOTE A description of the distinction between identification and designation is available in ISO/IEC 15944-1:2011[24], C.6.

8.2 Classes referenced from the Basic_and_Core package

8.2.1 Item class

8.2.1.1 Description of Item

Item is part of the Core metamodel region and is specified in 6.4.2.1. Additional associations and subclasses are specified in this metamodel region.

8.2.1.2 Associations of Item

This metamodel region specifies the following associations:

— **item_definition** (8.6.3);

— **item_designation** (8.6.4).

Other associations are added in other metamodel regions where the **Item** class is extended:

— the Core metamodel region (6.4.2.1.2);

— the Identification metamodel region (7.2.1.2);

— the Classification metamodel region (10.2.1.2);

— the Item Mapping metamodel region (11.2.1.2);

— and in ISO/IEC 11179-31[17], ISO/IEC 11179-32[18], ISO/IEC 11179-33[19], ISO/IEC 11179-34[20] and ISO/IEC 11179-35[21].

8.2.1.3 Attributes of Item

No attributes are specified in this metamodel region.

8.2.2 Context class

8.2.2.1 Direct superclass

Context is a subclass of **Item** (8.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

NOTE The subclassing of **Context** from **Item** is shown in Figure 4. The subclassing is omitted from Figure 6 to simplify the figure.

8.2.2.2 Description of Context

Context is part of the Core metamodel and is specified in 6.4.2.3. This clause describes its use in the Designation and Definition metamodel region.

Context is a class each instance of which models a context, the setting in which designations are used to designate and definitions are used to define registry items. Each registry item may be designated, defined or both within one or more contexts.

A context defines the setting within which the subject data has meaning. A context can be a business domain, an information subject area, an information system, a database, file, data model, standard document or any other environment determined by the stewardship organization responsible for the context, or the registration authority responsible for the registry. The different types of contexts can be distinguished by classifying them in a classification scheme.

NOTE 1 Usage of the **Context** class in the Data Specification metamodel region is described in ISO/IEC 11179-31:2023[17], 7.7.2.1 and 7.8.2.1.

NOTE 2 Usage of the **Context** class in the Data Set metamodel region is described in ISO/IEC 11179-33:2023[19], 7.2.2.2.

8.2.2.3 Associations of Context

As a subclass of **Item**, **Context** inherits **Item**'s associations (8.2.1.2). This metamodel region specifies the following association classes and association:

— **Definition_Context** (8.5.1) (association class);

— **Designation_Context** (8.5.2) (association class);

— **context_for_pairing** (8.6.1).

Additional associations of **Context** are specified in ISO/IEC 11179-31[17] and ISO/IEC 11179-33[19].

8.2.2.4 Attributes of Context

No attributes are specified in this metamodel region.

8.3 Classes referenced from the Identification package

8.3.1 Namespace class

8.3.1.1 Direct superclass

Namespace is a subclass of **Item** (6.4.2.1, 8.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

NOTE The subclassing of **Namespace** from **Item** is shown in Figure 5. The subclassing is omitted from Figure 6 to simplify the figure.

8.3.1.2 Description of Namespace

Namespace is described in 7.3.3. Additional constraints are specified in 8.3.1.5.

One use of **Namespace**s is to permit the **Item** represented by a **Designation.sign** to be uniquely determined for a **sign** within a particular **Namespace**.

8.3.1.3 Associations of Namespace

As a subclass of **Item**, **Namespace** inherits **Item**'s associations (8.2.1.2). This metamodel region specifies the following associations:

— **designation_namespace** (8.6.2);

— **naming_convention_utilization** (8.6.6).

Other associations of **Namespace** are specified as part of the Identification region (7.3.3.3) and as part of the Registration metamodel region (9.3.2.3).

8.3.1.4 Attributes of Namespace

See 7.3.3.4.

8.3.1.5 Constraints on Namespace

8.3.1.5.1 Constraints specified by mandatory_naming_convention_indicator

If the **mandatory_naming_convention_indicator** attribute (in **Namespace**) is TRUE:

a) there shall be exactly one naming convention associated with this namespace. That is, one instance of the **Naming_Convention** (8.4.4) class linked with this instance of the **Namespace** class via an instance of the **naming_convention_utilization** (8.6.6) association;

b) all designations that are associated with this namespace shall also be associated with the same naming convention as the namespace. That is, every instance of the **Designation** (8.4.1) class linked via an instance of the **designation_namespace** (8.6.2) association with this instance of the **Namespace** class shall also be linked via an instance of the **naming_convention_conformance** (8.6.5) association with the same instance of the **Naming_Convention** (8.4.4) class used in a) above.

If **mandatory_naming_convention_indicator** is FALSE, it is possible for this namespace to be associated with zero, one or more naming convention or for a designation to conform to more than one naming convention, or both. That is, an instance of the **Namespace** class may be linked with zero, one or more instances of the **Naming_Convention** class or for an instance of the **Designation** class may conform to more than one instance of the **Naming_Convention class**, or both.

8.3.1.5.2 Constraints specified by one_name_per_item_indicator

If the **one_name_per_item_indicator** attribute (in **Namespace**) is TRUE (a.k.a. unique names), then each registry item associated with the designations of this namespace has exactly one designation within this namespace. That is, each instance of the **Item** (8.2.1) class linked with an instance of the **Designation** (8.4.1) class that is also linked with this instance of the **Namespace** (8.3.1) class has exactly one instance of the **Designation** class linked with this instance of the **Namespace** class.

"Unique names" implies a functional mapping from **Item**s to **Designation.sign**s. In common parlance, no possibility of aliases exists. Thus, two distinct **Designation.sign**s (names) within a **Namespace** shall refer to separate **Item**s.

If the **one_name_per_item_indicator** is FALSE, then each registry item associated with the designations of this namespace may have more than one designation within this namespace. That is, each instance of the **Item** class linked with an instance of the **Designation** class that is also linked with this instance of the **Namespace** (8.3.1) class may be linked to more than one instance of the **Designation** class within this **Namespace**.

8.3.1.5.3 Constraints specified by one_item_per_name_indicator

If the **one_item_per_name_indicator** attribute (in **Namespace**) is TRUE (a.k.a. unambiguous names), then there exists at most one registry item associated with each designation in this namespace. That is, there exists at most one instance of the **Item** (8.2.1) class linked with an instance of the **Designation** (8.4.1) class that is also linked with this instance of the **Namespace** (8.3.1) class.

"Unambiguous names" implies a functional mapping from **Designation.sign**s to **Item**s.

8.4 Classes in the Designation and Definition metamodel region

8.4.1 Designation class

8.4.1.1 Description of Designation

Designation is a class each instance of which models a designation which uses a sign to denote a registry item. Each instance of **Designation** is linked with the instance of **Item** (8.2.1) that it denotes, via the **item_designation** association (8.6.4). Each instance of **Designation** is situated with respect to an instance of **Context** (8.2.2), **Naming_Convention** (8.4.4), **Namespace** (8.3.1), and may be paired with an instance of **Definition** (8.4.2).

NOTE 1 It is best practice to pair a designation with a definition (see 8.4.3) which occurs within a particular context.

NOTE 2 The requirement that a **Designation** be associated with an **Item**, means that it is not possible to use the registry simply to record terms as designations with associated definitions without reference to some item. It is anticipated that such a requirement would be met by associating each term with a **Concept** (6.4.2.2).

8.4.1.2 Associations of Designation

Designation has the following association class and associations:

— **Designation_Context** (8.5.2) (association class);

— **designation_namespace** (8.6.2);

— **item_designation** (8.6.4);

— **naming_convention_conformance** (8.6.5);

— **paired_designation** (8.6.8).

8.4.1.3 Attributes of Designation

See Table 14.

Table 14 — Attributes of Designation class

Attribute name	Multiplicity	Datatype	Description
sign	1	**Sign** (6.2.10)	Definition: sign of the designation
language	0..1	**Language_Identification** (6.3.7)	Conditional. Definition: language or dialect in which the sign (usually a name) is expressed This attribute is conditional because it is not always applicable (e.g. if the value of the **sign** attribute is an icon). Conditions: 1) If a language is applicable, a value for this attribute may be omitted if a value is specified for the **default_language** attribute of the **Registry_Specification** class (9.4.10). 2) If a language is applicable and a value has not been specified for the **default_language** attribute of the **Registry_Specification** class, a value shall be specified for this attribute.

Table 14 *(continued)*

Attribute name	Multiplicity	Datatype	Description
source	0..1	**Reference_Document** (6.3.8)	Definition: reference to the source from which the sign of the designation is taken

8.4.1.4 Constraint on Designation

8.4.1.4.1 Constraint 1

Each instance of **Designation** shall participate in either the **paired_designation** association (8.6.8) or the **Designation_Context** association class (8.5.2) or both.

8.4.1.4.2 Constraint 2

If an instance of the **Designation** class is linked via the **item_designation** (8.6.4) association to an instance of the **Item** (8.2.1) class which is subsequently deleted, then the instance of **Designation** shall also be deleted.

8.4.2 Definition class

8.4.2.1 Description of Definition

Definition is a class each instance of which models a definition, a representation of a registry item by a descriptive statement which, in a given language and context(s), serves to differentiate it from other registry items. Each registry item may be associated with zero, one or more definitions, each definition being specified in a particular language. Each instance of **Definition** is linked with the instance of **Item** (8.2.1) that it describes, via the **item_definition** association (8.6.3). Each instance of **Definition** may be paired with an instance of **Designation** (8.4.1).

8.4.2.2 Associations of Definition

Definition has the following association class and associations:

— **Definition_Context** (8.5.1) (association class);

— **item_definition** (8.6.3);

— **paired_definition** (8.6.7).

8.4.2.3 Attributes of Definition

See Table 15.

Table 15 — Attributes of Definition class

Attribute name	Multiplicity	Datatype	Description
text	1	**Text** (6.2.12)	Definition: text of the definition
language	0..1	**Language_Identification** (6.3.6)	Definition: language used to write the text of the definition, that is the language used for the value of the **text** attribute Conditions: 1) If a language is applicable, a value for this attribute may be omitted if a value is specified for the **default_language** attribute of the **Registry_Specification** class (9.4.10). 2) If a language is applicable and a value has not been specified for the **default_language** attribute of the **Registry_Specification** class, a value shall be specified for this attribute.
source	0..1	**Reference_Document** (6.3.8)	Definition: reference to the source from which the text of the definition is taken

8.4.2.4 Constraints on Definition

8.4.2.4.1 Constraint 1

Each instance of **Definition** shall participate in either the **paired_definition** association (8.6.7) or the **Definition_Context** association class (8.5.1) or both.

8.4.2.4.2 Constraint 2

If an instance of the **Definition** class is linked via the **item_definition** (8.6.3) association to an instance of the **Item** (8.2.1) class which is deleted, then the instance of **Definition** shall also be deleted.

8.4.3 Designation_Definition_Pairing class

8.4.3.1 Description of Designation_Definition_Pairing

Designation_Definition_Pairing is a class, each instance of which models the pairing of a designation with a definition in a particular context.

8.4.3.2 Associations of Designation_Definition_Pairing

Designation_Definition_Pairing has the following associations:

— **context_for_pairing** (8.6.1);

— **paired_definition** (8.6.7);

— **paired_designation** (8.6.8).

8.4.3.3 Attributes of Designation_Definition_Pairing

See Table 16.

Table 16 — Attributes of *Designation_Definition_Pairing* class

Attribute name	Multiplicity	Datatype	Description
acceptability	0..1	**Acceptability** (8.7.1)	Definition: acceptability rating of the designation and definition in the specified context.

8.4.4 Naming_Convention class

8.4.4.1 Direct superclass

Naming_Convention is a subclass of **Item** (6.4.2.1, 8.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

NOTE The subclassing of **Naming_Convention** from **Item** is shown in Figure A.1. The subclassing is omitted from Figure 6 to simplify the figure.

8.4.4.2 Description of Naming_Convention

Naming_Convention is a class, each instance of which models a naming convention, which provides a set of rules for constructing the sign of a designation of a registry item. A naming convention can range in complexity from simple to complex. The semantic, syntactic and lexical rules may each have their own complexity.

NOTE ISO/IEC 11179-5[15] has a more elaborate discussion of naming conventions.

8.4.4.3 Associations of Naming_Convention

As a subclass of **Item**, **Naming_Convention** inherits **Item**'s associations (8.2.1.2). **Naming_Convention** has the following additional associations:

— **naming_convention_conformance** (8.6.5);

— **naming_convention_utilization** (8.6.6).

8.4.4.4 Attributes of Naming_Convention

See Table 17.

Table 17 — Attributes of Naming_Convention class

Attribute name	Multiplicity	Datatype	Description
scope_rule	1	**Text** (6.2.12)	Definition: rule specifying the range within which the naming convention is in effect RULE: In terms of the metadata registry, the scope of a naming convention may be as broad or narrow as the registration authority, or other authority, determines is appropriate. The scope should document whether the naming convention is descriptive or prescriptive.
authority_rule	0..1	**Text** (6.2.12)	Definition: rule identifying the authority that assigns names or enforces naming conventions, or both Examples of authorities include information technology standards committees or nomenclature standardization bodies (e.g. in biology).

Table 17 *(continued)*

Attribute name	Multiplicity	Datatype	Description
semantic_rule	0..1	**Text** (6.2.12)	Definition: rule specifying the meanings of parts of a name and possibly separators that delimit them in a naming convention The rule should specify whether or not names convey meaning and if so how.
syntactic_rule	0..1	**Text** (6.2.12)	Definition: rule specifying the arrangement of parts of a name and the separators that delimit them in a naming convention The arrangement may be specified as relative or absolute, or some combination of the two. Relative arrangement specifies parts in terms of other parts, e.g. a rule within a convention that requires a qualifier term always appear before the part being qualified appears. Absolute arrangement specifies a fixed occurrence of the part, e.g. a rule that requires the property term be always the last part of a name. The syntactic principle can also specify the syntactic forms of the name (noun phrase or verb phrase) and the parts of speech used to construct a name.
lexical_rule	0..1	**Text** (6.2.12)	Definition: rule specifying the appearance of names: preferred and non-preferred terms, synonyms, abbreviations, part length, spelling, permissible character set, case sensitivity, etc. [Derived from ISO/IEC 11179-5[15]] The result of applying lexical rules should be that all names governed by a specific naming convention have a consistent appearance. An example lexical principle is the specification of the use of camelCase capitalization of words in a phrase which are concatenated together.

8.5 Association classes in the Designation and Definition metamodel region

8.5.1 Definition_Context association class

8.5.1.1 Description of Definition_Context

Definition_Context is an association which records the bindings of zero, one or more instances of the **Definition** (8.4.2) class with zero, one or more instances of the **Context**s (8.2.2) class. **Definition_Context** is also a class with one attribute as specified in 8.5.1.2.

This association registers the context(s) which provide the scope for each of the definitions.

NOTE The requirement that all definitions be associated with at least one context applies even when the item being defined is a context, and this does not create a problem of infinite regress. For example, one straightforward way to satisfy this requirement is to include, within each context, the definition(s) for itself; another is to place all definitions for contexts in a "registry context" [including the definition(s) for the registry context itself].

8.5.1.2 Attributes of Definition_Context

See Table 18.

Table 18 — Attributes of Definition_Context association class

Attribute name	Multiplicity	Datatype	Description
acceptability	0..1	**Acceptability** (8.7.1)	Definition: acceptability rating of the definition in the specified context.

8.5.2 Designation_Context association class

8.5.2.1 Description of Designation_Context

Designation_Context is an association which records the bindings of zero, one or more instances of the **Designation** (8.4.1) class with one or more instances of the **Context** (8.2.2) class. **Designation_Context** is also a class with one attribute as specified in 8.5.2.2.

This association registers the context(s) which provide the scope for each of the designations.

NOTE The requirement that all designations be associated with at least one context applies even when the item being designated is a context, and this does not create a problem of infinite regress. For example, one straightforward way to satisfy this requirement is to include, within each context, the designation(s) for itself; another is to place all designations for contexts in a "registry context" (including the designation(s) for the registry context itself).

8.5.2.2 Attributes of Designation_Context

See Table 19.

Table 19 — Attributes of Designation_Context association class

Attribute name	Multiplicity	Datatype	Description
acceptability	0..1	**Acceptability** (8.7.1)	Definition: acceptability rating of the designation in the specified context.

8.6 Associations in the Designation and Definition metamodel region

8.6.1 context_for_pairing association

The **context_for_pairing** association records the binding of exactly one instance of the **Context** (8.2.2) class to zero, one or more instances of the **Designation_Definition_Pairings** (8.4.3) class. The association records the context in which a designation and definition are paired.

8.6.2 designation_namespace association

The **designation_namespace** association records the bindings of zero, one or more instances of the **Designation** (8.4.1) class to zero, one or more instances of the **Namespace** (8.3.1) class. The association records the namespaces in which a designation is valid.

NOTE Unlike the **identifier_scope** (7.4.3) association, this association is optional in both directions, so instances of **Designation** are not deleted when the linked instance of **Namespace** is deleted.

8.6.3 item_definition association

The **item_definition** association records the binding of exactly one instance of the **Item** (8.2.1) class to zero, one or more instances of the **Definition** (8.4.2) class. The association records all of the definitions for a specific registry item.

A definition shall be used for exactly one registry item. Definitions shall not be reused across multiple registry items because each such item should be distinct and distinguishable, and this should be reflected in the definition, if not in the text itself, then in the associated context.

8.6.4 item_designation association

The **item_designation** association records the binding of exactly one instance of the **Item** (8.2.1) class to zero, one or more instances of the **Designation**s (8.4.1) class.

This association records that the registry item is designated by the designation(s) by recording all the designations (sign plus language pairs) for a registry item.

A designation shall be used for exactly one registry item. Designations shall not be reused across multiple registry items because each such registry item should be distinct and distinguishable, and this should be reflected in the designations, if not in sign of the designation itself, then in the associated context or namespace.

8.6.5 naming_convention_conformance association

The **naming_convention_conformance** association records the binding of zero, one or more instances of the **Designation** (8.4.1) class to zero, one or more instances of the **Naming_Convention** (8.4.4)class.

This association records the naming convention to which a designation conforms.

8.6.6 naming_convention_utilization association

The **naming_convention_utilization** association records the binding of zero, one or more instances of the **Namespace** (8.3.1) class to zero, one or more instances of the **Naming_Convention** (8.4.4) class.

This association records the naming convention used by the namespace.

8.6.7 paired_definition association

The **paired_definition** association records the binding of exactly one instance of the **Definition** (8.4.2) class to zero, one or more instances of the **Designation_Definition_Pairing** (8.4.3) class.

This association records that the definition is paired with a particular designation within a particular context.

8.6.8 paired_designation association

The **paired_designation** association records the binding of exactly one instance of the **Designation** (8.4.1) class to zero, one or more instances of the **Designation_Definition_Pairing**s (8.4.3) class.

This association records that the designation is paired with a particular definition within a particular context.

8.7 Datatypes in the Designation and Definition metamodel region

8.7.1 Acceptability enumeration

8.7.1.1 Description of Acceptability

Acceptability models a scale of acceptability ratings. See Table 20 for possible ratings and their meanings. This enumeration is used as a datatype for the attributes **Designation.acceptability** (8.4.1.3) and **Definition.acceptability** (8.4.2.3).

8.7.1.2 Enumeration of *Acceptability* ratings

Table 20 lists the possible ratings and their meanings.

Table 20 — Enumeration of Acceptability ratings

Rating	Meaning
preferred	The best option.
admitted	An acceptable option but not as good as the preferred option.
deprecated	No longer considered acceptable but is still in use.
superseded	Has been replaced by another option which is used instead.
obsolete	No longer in use but has not been superseded.
NOTE 'option' refers to members of a set of designations or definitions.	

The sequence in which the ratings are applied to any particular instance may vary. **Designation. acceptability** (8.4.1.3) and **Definition.acceptability** (8.4.2.3) should be considered in conjunction with **Registration_State. registration_status** (9.4.4.3).

The rating "obsolete" probably only makes sense if the **Registration_State.registration_status** (9.4.4.3) of the **Administered_Item** (9.4.2) is "Retired".

8.7.1.3 Constraint on Acceptability

Only one option (member of a set of designations or definitions) may be specified as "preferred".

9 Registration package

9.1 Overview of Registration metamodel region

The Registration package consists of one region, the Registration metamodel region, which supports the registration of items in a registry. ISO/IEC 11179-6 further describes the registration of **Registered_ Items** (9.4.1) as either **Administered_Items** (9.4.2) or **Attached_Items** (9.4.3).

Figure 7 shows the classes, relationships, attributes and composite attributes that support Registration.

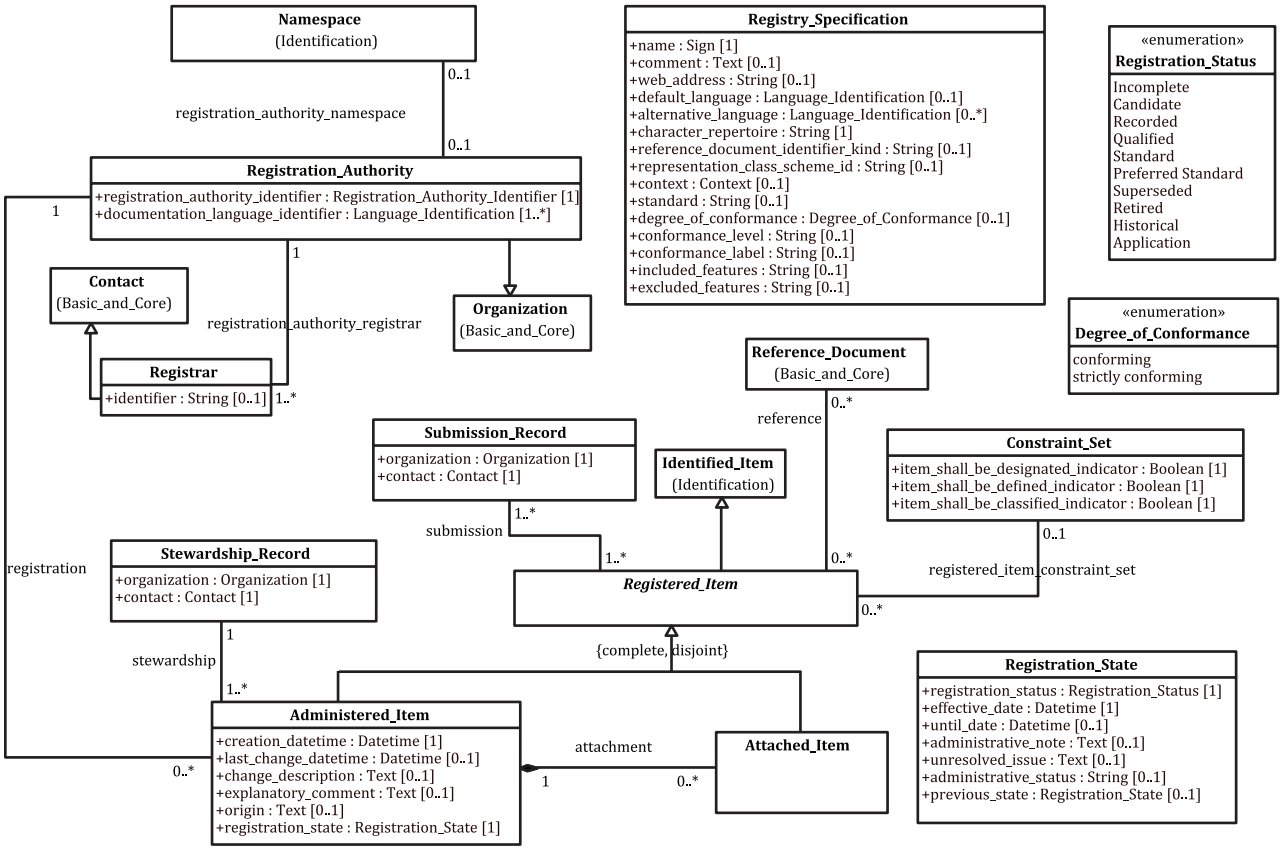

Figure 7 — Registration metamodel region

Although the Figure does not show it, there is an implicit association between **Registry_Specification** and **Item** in that the specification applies to all items in the registry. If an implementation were to implement multiple registries within a single system, then the association between each registry specification and its corresponding items would need to be made explicit.

Annex C provides an example of registering a conceptual domain.

9.2 Classes referenced from the Basic and core package

9.2.1 Contact class

Contact is a class each instance of which models a contact, which specifies a role, an individual or both within an organization or an organization part to whom information item(s), material object(s), person(s) or some combination can be sent to or from.

Registrar (9.4.7) is a subclass of **Contact**. **Contact** is further described in 6.3.5.

9.2.2 Organization class

Organization is a class each instance of which models an organization, a unique framework of authority within which individuals act, or are designated to act, towards some purpose.

An organization can play one or more roles with respect to a metadata registry.

— Each registration authority is a specialization of an organization, so, in this region, **Registration_Authority** (9.4.6) is a subclass of **Organization**.

— An organization can be a submission organization within a submission record, each of which may be the submission record for many submitted registered items, so, in this region, a submission

organization can be recorded by the appropriate use of an instance of the **Organization** class as the value for the **organization** attribute within the appropriate instance of the **Submission_Record** class (9.4.9), with one or more instances of the **Registered_Item** class (9.4.1) linked through an instance of the **submission** association (9.5.8).

— An organization can be a stewardship organization within a stewardship record, each of which may be the stewardship record for many administered items, so, in this region, a stewardship organization can be recorded by the use of the appropriate instance of the **Organization** class as the value for **organization** attribute within the appropriate instance of the **Stewardship_Record** class (9.4.8), with one or more instances of the **Administered_Item** class (9.4.2) linked through an instance of the **stewardship** association (9.5.7).

Organization is further described in 6.3.3.

9.2.3 Reference_Document class

Reference_Document is a class each instance of which models a reference document, a document that provides pertinent details for consultation about a subject. **Reference_Document** is specified in 6.3.8.

Reference_Document participates in the **reference** association (9.5.2).

9.3 Classes referenced from the Identification package

9.3.1 Identified_Item

Identified_Item is specified in 7.3.1. **Identified_Item** is the superclass of **Registered_Item** (9.4.1).

9.3.2 Namespace class

9.3.2.1 Direct superclass

Namespace is a subclass of **Item** (6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

NOTE The subclassing of **Namespace** from **Item** is shown in Figure 5. The subclassing is omitted from Figure 7 to simplify the figure.

9.3.2.2 Description of Namespace

Namespace is described in 7.3.3.

9.3.2.3 Associations of Namespace

As a subclass of **Item**, a **Namespace** inherits **Item**'s associations (7.2.1.2, 8.2.1.2). This metamodel region specifies the following additional association:

— **registration_authority_namespace** (9.5.5).

Other associations of **Namespace** are specified as part of the Identification region (7.3.3.3) and as part of the Designation and Definition metamodel region (8.3.1.3).

9.4 Classes in the Registration region

9.4.1 Registered_Item class

9.4.1.1 Direct superclass

Identified_Item (7.3.1).

9.4.1.2 Description of Registered_Item

Registered_Item is a class each instance of which models a registered item. A registered item is an identified item that is registered and managed in a metadata registry.

As an abstract superclass, **Registered_Item** shall be instantiated as either an **Administered_Item** (9.4.2) or an **Attached_Item** (9.4.3) but not both.

9.4.1.3 Associations of Registered_Item

As a subclass of **Identified_Item**, **Registered_Item** inherits **Identified_Item**'s associations (7.3.1.2). **Registered_Item** has the following additional associations:

— **reference** association (9.5.2);

— **registered_item_constraint_set** association (9.5.3);

— **submission** association (9.5.8).

A registration authority may specify that a registered item is required to have at least one designation and definition. See 9.4.5.4.

9.4.1.4 Attributes of Registered_Item

No attributes are specified in this metamodel region.

9.4.2 Administered_Item class

9.4.2.1 Direct superclass

Registered_Item (9.4.1).

9.4.2.2 Description of Administered_Item

Administered_Item is a class each instance of which models an administered item. An administered item is a registered item for which administrative information is recorded by a registration authority.

9.4.2.3 Associations of Administered_Item

As a subclass of **Registered_Item**, **Administered_Item** inherits **Registered_Item**'s associations (9.4.1.3). **Administered_Item** has the following additional associations:

— **attachment** association (9.5.1);

— **registration** association (9.5.4);

— **stewardship** association (9.5.7).

9.4.2.4 Attributes of Administered_Item

See Table 21.

Table 21 — Attributes of Administered_Item class

Attribute name	Multiplicity	Datatype	Description
creation_datetime	1	**Datetime** (6.2.3)	Definition: date and time the administered item was created
last_change_datetime	0..1	**Datetime** (6.2.3)	Definition: date and time the administered item was last changed

Table 21 *(continued)*

Attribute name	Multiplicity	Datatype	Description
change_description	0..1	**Text** (6.2.12)	Definition: description of what has changed since the prior version of the administered item Previous versions of the administered item can be maintained in the registry, or not.
explanatory_comment	0..1	**Text** (6.2.12)	Definition: descriptive comments about the administered item
origin	0..1	**Text** (6.2.12)	Definition: the source (e.g. document, project, discipline or model) for the administered item
registration_state	1	**Registration_State** (9.4.4)	Definition: current collection of administrative information about the registration of this administered item.

9.4.3 Attached_Item class

9.4.3.1 Direct superclass

Registered_Item (9.4.1).

9.4.3.2 Description of Attached_Item

Attached_Item is a class each instance of which models an attached item. An attached item is a registered item for which administrative information is recorded in another registered item (an administered item). Every attached item has an owning administered item which supplies the administrative information for both the administered item and its attached items. In this metamodel region, therefore, **Attached_Item** has an **attachment** (9.5.1) association with **Administered_Item** (9.4.2).

The attached item concept provides a mechanism by which to administer a set of registered items together, as a group, rather than maintaining separate administrative information for every individual registry item.

EXAMPLE 1 In ISO/IEC 11179-31[17], in the Conceptual and Value Domain metamodel region, the value meanings, represented by instances of the **Value_Meaning** class, within a conceptual domain, represented by the instance of the **Conceptual_Domain** class that is linked to those instances of the **Value_Meaning** class, can be attached to, and thus administered with, the containing conceptual domain. The instance of the **Conceptual_Domain** class is linked to an instance of the **Administered_Item** class, while the instances of the **Value_Meaning** class are linked to instances of the **Attached_Item** class.

EXAMPLE 2 In ISO/IEC 11179-32[18], in the Concept System metamodel region, all the assertions, represented by instances of the **Assertion** class, and the concepts, represented by instances of the **Concept** class, within a concept system, represented by the instance of the **Concept_System** class that is linked to those instances of the Assertion and Concept classes, can be attached to, and thus administered with, the containing concept system. The instance of the **Concept_System** class is linked to an instance of the **Administered_Item** class, while the instances of the **Assertion** and **Concept** classes are linked to instances of the **Attached_Item** class.

EXAMPLE 3 Similarly, in ISO/IEC 11179-33[19] all the distributions, assessments. specifications, etc., can be attached to, and thus administered with, the data set to which they refer.

EXAMPLE 4 Similarly, in ISO/IEC 11179-35[21] all the model elements can be attached to, and thus administered with, the model of which they are a part.

9.4.3.3 Associations of Attached_Item

As a subclass of **Registered_Item**, **Attached_Item** inherits **Registered_Item**'s associations (9.4.1.3). **Attached_Item** has the following additional association:

— **attachment** (9.5.1).

9.4.3.4 Attributes of Attached_Item

No attributes are specified in this metamodel region.

9.4.3.5 Constraint on Attached_Item

If an instance of the **Attached_Item** class is linked via the **attachment** (9.5.1) association with an instance of the **Administered_Item** (9.4.2) class which is subsequently deleted, then the instance of **Attached_Item** shall also be deleted.

9.4.4 Registration_State class

9.4.4.1 Description of Registration_State

Registration_State is a class, each instance of which models a registration state, a collection of information about the registration of an administered item. The **Registration_State** class is used as a datatype for the **registration_state** attribute of the **Administered_Item** class (9.4.2).

9.4.4.2 Associations of Registration_State

No associations are specified in this metamodel region.

9.4.4.3 Attributes of Registration_State

See Table 22.

Table 22 — Attributes of Registration_State class

Attribute name	Multiplicity	Datatype	Description
registration_status	1	**Registration_Status** (9.6.2)	Definition: designation of the status in the registration life-cycle of the administered item that is the subject of the registration with this registration status Permitted values for registration_status are specified in 9.6.1.
effective_date	1	**Datetime** (6.2.3)	Definition: date and time the administered item that is the subject of the registration with this registration status became or becomes available to registry users
until_date	0..1	**Datetime** (6.2.3)	Definition: date and time the registration by the registration authority in the registry of the administered item that is the subject of the registration with this registration status was, or will be, no longer effective
administrative_note	0..1	**Text** (6.2.12)	Definition: general note(s) about the registration of the administered item that is the subject of the registration with this registration status

Table 22 *(continued)*

Attribute name	Multiplicity	Datatype	Description
unresolved_issue	0..1	**Text** (6.2.12)	Definition: any problem(s) that remains unresolved regarding proper documentation of the administered item that is the subject of the registration with this registration status
administrative_status	0..1	**String** (6.2.11)	Definition: designation of the status in the administrative process of the registration authority responsible for the administered item that is the subject of the registration with this registration status The values, and their associated meanings, to be applied to this attribute are determined by each registration authority. Cf. registration status.
previous_state	0..1	**Registration_State** (9.4.4)	Definition: immediately prior collection of administrative information about the registration of the administered item that is the subject of the registration with this registration status.

9.4.5 Constraint_Set class

9.4.5.1 Direct superclass

Constraint_Set is a subclass of **Item** (6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

9.4.5.2 Description of Constraint_Set

A **Constraint_Set** is a set of constraints about zero, one or more registered items. A registration authority can apply a **Constraint_Set** to a **Registered_Item** (9.4.1) to specify the constraints that should be enforced when its **Administered_Item.registration_state.registration_status** (9.4.4.3) is "Recorded" or higher.

9.4.5.3 Associations of Constraint_Set

Constraint_Set has the following association:

— **registered_item_constraint_set** (9.5.3).

9.4.5.4 Attributes of Constraint_Set

See Table 23.

Table 23 — Attributes of Constraint_Set class

Attribute name	Multiplicity	Datatype	Description
item_shall_be_designated_indicator	1	**Boolean** (6.2.2)	Definition: indicator as to whether the linked registered item is required to have a designation when its registration status is 'Recorded' or higher. That is, whether the parent instance of the **Item** (8.2.1) class has a linked instance of the **Designation** (8.4.1) class via the **item_designation** (8.6.4) association when the value of the **Registration_State.registration_status** (9.4.4.3) is 'Recorded' or higher. Best practice is to set this to TRUE.
item_shall_be_defined_indicator	1	**Boolean** (6.2.2)	Definition: indicator as to whether the linked registered item is required to have a definition when its registration status is 'Recorded' or higher. That is, whether the parent instance of the **Item** (8.2.1) class has a linked instance of the **Definition** (8.4.2) class via the **item_definition** (8.6.3) association when the value of the **Registration_State.registration_status** (9.4.4.3) is 'Recorded' or higher. Best practice is to set this to TRUE.
item_shall_be_classified_indicator	1	**Boolean** (6.2.2)	Definition: indicator as to whether the linked registered item is required to be a classified item when its registration status is 'Recorded' or higher. That is, whether the parent instance of the **Item** (8.2.1) class has a linked instance of the **Classification_Scheme_Item** (10.3.2) class via the **item_classification** (10.4.1) association when the value of the **Registration_State.registration_status** (9.4.4.3) is 'Recorded' or higher.

9.4.5.5 Subclassing Constraint_Set

Other parts of ISO/IEC 11179 specify subclasses of **Constraint_Set** as a way of specifying constraints in other parts of the metamodel.

9.4.6 Registration_Authority class

9.4.6.1 Direct superclass

Organization (6.3.3).

9.4.6.2 Description of Registration_Authority

Registration_Authority is a class each instance of which models a registration authority, an organization responsible for maintaining a register.

9.4.6.3 Associations of Registration_Authority

As a subclass of **Organization**, a **Registration_Authority** inherits **Organization**'s associations. **Registration_Authority** has the following additional associations:

— **registration** association (9.5.4);

— **registration_authority_namespace** association (9.5.5);

— **registration_authority_registrar** association (9.5.6).

9.4.6.4 Attributes of Registration_Authority

See Table 24.

Table 24 — Attributes of Registration_Authority class

Attribute name	Multiplicity	Datatype	Description
registration_authority_identifier	1	**Registration_Authority_Identifier** (6.3.9)	Definition: identifier of a registration authority
documentation_language_identifier	1..*	**Language_Identification** (6.3.6)	Definition: identifier of the language used for documentation by the registration authority

9.4.7 Registrar class

9.4.7.1 Direct superclass

See **Contact** (9.2.1).

9.4.7.2 Description of Registrar

Registrar is a class each instance of which represents a registrar, a contact that is a representative of the registration authority. A registration authority is represented by one or more registrars.

The registrars are the persons who perform the administrative steps to register administered items in a metadata registry.

9.4.7.3 Associations of Registrar

See **registration_authority_registrar** association (9.5.6).

9.4.7.4 Attribute of Registrar

See Table 25.

Table 25 — Attributes of Registrar class

Attribute name	Multiplicity	Datatype	Description
identifier	0..1	**String** (6.2.11)	Definition: identifier for this registrar

9.4.8 Stewardship_Record class

9.4.8.1 Description of Stewardship_Record

Stewardship_Record is a class each instance of which models a stewardship record, which identifies both the stewardship organization, a specialization of an organization, and the stewardship contact,

a specialization of a contact, responsible for the stewardship of one or more administered items. Each instance of the **Stewardship_Record** class is linked to the relevant instance of **Administered_Item** class (9.4.2) via the **stewardship** association (9.5.7).

NOTE To allow for various types of outsourcing, the stewardship contact does not have to work directly for the stewardship organization. Thus, the value of the **organization** attribute of the instance of **Contact** that is the value of the **Stewardship_Record.contact** attribute can be different from that of the value of the **Stewardship_Record.organization** attribute.

9.4.8.2 Associations of Stewardship_Record

See **stewardship** association (9.5.7).

9.4.8.3 Attributes of Stewardship_Record

See Table 26.

Table 26 — Attributes of Stewardship_Record class

Attribute name	Multiplicity	Datatype	Description
organization	1	**Organization** (6.3.3)	Definition: information about the organization that is responsible for the stewardship of the administered items that are linked to this stewardship record.
contact	1	**Contact** (6.2.3)	Definition: information about the contact that is responsible for the stewardship of the administered items that are linked to this stewardship record.

9.4.9 Submission_Record class

9.4.9.1 Description of Submission_Record

Submission_Record is a class each instance of which models a submission record which identifies both the submission organization, a specialization of an organization, and the submission contact, a specialization of a contact, responsible for the submission of one or more registered items for addition, change or cancellation/withdrawal within a metadata registry. The submission contact is the contact for issues related to the submission. Each instance of the **Submission_Record** class is linked to the relevant instance of **Registered_Item** class (9.4.1) via the *submission* association (9.5.8).

NOTE 1 To allow for various types of outsourcing, the submission contact does not have to work directly for the submission organization. Thus, the value of the **organization** attribute of the instance of **Contact** that is the value of the **Submission_Record.contact** attribute can be different from that of the value of the **Submission_Record.organization** attribute.d

NOTE 2 A submission record is required for attached items as well as for administered items because they can be submitted separately. However, one submission record can be used for multiple administered items, attached items or both submitted together.

9.4.9.2 Associations of Submission_Record

See **submission** association (9.5.8).

9.4.9.3 Attributes of Submission_Record

See Table 27.

Table 27 — Attributes of Submission_Record class

Attribute name	Multiplicity	Datatype	Description
organization	1	**Organization** (6.3.3)	Definition: information about the organization that is responsible for the submission of the registered items that are linked to this submission record.
contact	1	**Contact** (6.2.3)	Definition: information about the contact that is responsible for the submission of the registered items that are linked to this submission record.

9.4.10 Registry_Specification class

9.4.10.1 Description of Registry_Specification

Registry_Specification describes the environment in which a registry operates. If this document is being applied in an environment that does not use a registry, then **Registry_Specification** is not required. In an environment with multiple registries, there should be one **Registry_Specification** per registry.

9.4.10.2 Associations of Registry_Specification

No associations are specified in this metamodel region.

9.4.10.3 Attributes of Registry_Specification

See Table 28.

Table 28 — Attributes of Registry_Specification class

Attribute name	Multiplicity	Datatype	Description
name	1	**Sign** (6.2.10)	Definition: name by which the registry is commonly known Example: Australian Institute of Health and Welfare (AIHW) Metadata Online Registry (METeOR)[39]
comment	0..1	**Text** (6.2.12)	Definition: any comment that should be noted about the registry
web_address	0..1	**String** (6.2.11)	Definition: The World Wide Web uniform resource locator (URL) for the registry Example: https://meteor.aihw.gov.au/
default_language	0..1	**Language_Identification** (6.3.7)	Definition: The **l**anguage that is used as the default in the registry for designations, definitions and reference documents. Examples: eng-840, en-US
alternative_language	0..*	**Language_Identification** (6.3.7)	Definition: Any other language that is used in the registry Examples: eng-840, en-US

Table 28 *(continued)*

Attribute name	Multiplicity	Datatype	Description
character_repertoire	1	**String** (6.2.11)	Definition: The character repertoire that is used by the registry for internal operation Example: ISO/IEC 646[3] or ISO/IEC 10646[10]
reference_document_identifier_kind	0..*	**String** (6.2.11)	Definition: Specification of the kind of identifier used for identifying reference documents in the registry Examples: Some registries use URIs. Other registries use an external document management system which provides unstructured, opaque identifiers. Yet other registries define a structured identifier form which embeds an identifier type within each identifier in the registry. This attribute specifies the form of identifiers used for reference documents in this particular registry.
representation_class_scheme_id	0..1	**String** (6.2.11)	Definition: identifier of the classification scheme used by the registry to capture representation classes Changes from previous editions: Edition 2 (ISO/IEC 11179-3:2003[12]) had a separate structure to record "representation classes". In edition 3 (ISO/IEC 11179-3:2013[13]) and edition 4 (this document), these are considered to be just another classification scheme, which in edition 3 was considered a kind of concept system. This document has restored a **Classification_Scheme** class independent of the **Concept_System** class specified in ISO/IEC 11179-32. This attribute allows the registration authority to specify which classification scheme is used for this purpose. This document uses just the identifier of the classification scheme for this attribute. Edition 3 embedded a whole concept system.

Table 28 *(continued)*

Attribute name	Multiplicity	Datatype	Description
context	0..1	**Context** (6.4.2.3)	Definition: context which represents the registry itself It is sometimes useful to reference a default context, rather than create a new one. This attribute is one possible choice for such a default.
standard	0..1	**String** (6.2.11)	Definition: Identifier for the standard to which this registry complies. Example: ISO/IEC 11179-3:2023
degree_of_conformance	0..1	**Degree_of_Conformance** (9.6.1)	Definition: 'conforming' or 'strictly conforming'. See 4.2. Condition: applies only if **standard** is specified.
conformance_level	0..1	**String** (6.2.11)	Definition: The conformance level of the registry as described in the standard Example: Conformance Level 2 (where standard is specified as ISO/IEC 11179-3:2003). Condition: applies only if **standard** is specified, and the specified standard uses conformance levels. Note: This document specifies conformance labels instead of conformance levels.
conformance_label	0..1	**String** (6.2.11)	Definition: The conformance label of the registry as described in the standard Example: ISO/IEC 11179-3:2023 Basic Registry (see 4.4.3). Condition: applies only if **standard** is specified, and the specified standard uses conformance labels.
included_features	0..1	**String** (6.2.11)	Definition: Freeform specification of the features of the standard that this implementation supports. Condition: applies only if **standard** is specified, and the specified standard allows conformance by feature (see 4.3).

Table 28 *(continued)*

Attribute name	Multiplicity	Datatype	Description
excluded_features	0..1	**String** (6.2.11)	Definition: Freeform specification of the features of the standard that this implementation does not support. Condition: applies only if **standard** is specified, and the specified standard allows conformance by feature (see 4.3).
default_time_zone	0..1	**Datetime** (6.2.3) (only hours and minutes are used)	Definition: time zone to be assumed for any datetime instance that does not explicitly specify its own time zone. Specified as the number of hours offset from UTC, with no separator between hours and minutes, with a leading zero if the hour is less then 9, and a leading minus sign if applicable. Example: For Newfoundland Daylight Time: -0230

9.5 Associations in the Registration region

9.5.1 attachment association

The **attachment** association records the binding of exactly one instance of the **Administered_Item** (9.4.2) class with zero, one or more instances of the **Attached_Item** (9.4.3) class. This association records that the attached items share all of the administrative information of the linked administered item, enabling collections of registered items to be administered collectively as a block.

NOTE Attached items can share the same administered item and still have different submission organizations.

9.5.2 reference association

The **reference** association records the binding of zero, one or more instances of the **Reference_Document** (9.2.3) class with zero, one or more instances of the **Registered_Item** (9.4.1) class.

This association records the reference documents that provide additional information about the linked registered items.

9.5.3 registered_item_constraint_set association

The **registered_item_constraint_set** association records the binding of zero, one or more instances of the **Registered_Item** (9.4.1) class with zero or one instance of the **Constraint_Set** (9.4.5) class.

This association records the set of constraints that the registration authority deems appropriate for the linked registered items.

9.5.4 registration association

The **registration** association records the binding of exactly one instance of the **Registration_Authority** (9.4.6) class with zero, one or more instances of the **Administered_Item** (9.4.2) class.

This association records the registration authority responsible for the register in which the linked administered items are registered.

9.5.5 registration_authority_namespace association

The **registration_authority_namespace** association records the binding of zero or one instances of the **Registration_Authority** (9.4.6) class with zero or one instance of the **Namespace** (9.3.1) class.

This association records the namespace that provides the scope for the registration authority identifier, represented by the instance of the **Registration_Authority_Identifier** (6.3.9) class used as the value for the **registration_authority_identifier** attribute, for the registration authority.

9.5.6 registration_authority_registrar association

The **registration_authority_registrar** association records the binding of exactly one instance of the **Registration_Authority** (9.4.6) class with one or more instances of the **Registrar** (9.4.7) class.

This association records the registrars that represent the registration authority.

9.5.7 stewardship association

The **stewardship** association records the binding of exactly one instance of the **Stewardship_Record** (9.4.8) class with one or more instances of the **Administered_Item** (9.4.2) class.

This association records the stewardship record for the administered item, where the stewardship record identifies the stewardship organization, represented by an instance of the **Organization** (6.3.3) class, and the stewardship contact, represented by an instance of the **Contact** (6.3.5) class, who are responsible for maintaining the administrative information for the administered item.

9.5.8 submission association

The **submission** association records the binding of one or more instances of the **Submission_Record** (9.4.9) class with one or more instances of the **Registered_Item** (9.4.1) class.

This association records the submission record for the registered item, where the submission record identifies the submission organization, represented by an instance of the **Organization** (6.3.3) class, and the submission contact, represented by an instance of the **Contact** (6.3.5) class.

The submission organization is the organization that submitted the registered item for addition, change, cancellation or withdrawal within the metadata registry.

The submission contact is the contact at the submission organization for issues related to the submission.

NOTE Attached items can share the same administered item and still have different submission organizations.

9.6 Datatypes in the Registration metamodel region

9.6.1 Degree_of_Conformance enumeration

Degree_of_Conformance models degree of conformance (4.2). See Table 29 for possible values.

Table 29 — Enumeration of Degree_of_Conformance

Value	Meaning
conforming	From 4.2.3: A conforming implementation: a) shall support all mandatory, optional and conditional classes, attributes, datatypes and associations; b) as permitted by the implementation, may use, test, access or probe for extension features or extensions to classes, attributes, datatypes, associations or any combination thereof; c) may recognize, act on or allow the production of classes, attributes, datatypes, associations or any combination thereof that are dependent on implementation-defined behaviour.
strictly conforming	From 4.2.2: A strictly conforming implementation: a) shall support all mandatory, optional and conditional classes, attributes, datatypes and associations; b) shall not use, test, access or probe for any extension features nor extensions to classes, attributes, datatypes, associations or any combination thereof; c) shall not recognize, nor act on, nor allow the production of classes, attributes, datatypes, associations or any combination thereof that are dependent on any unspecified, undefined or implementation-defined behaviour.

All strictly conforming implementations are also conforming implementations.

The use of extensions to the metamodel can cause undefined behaviour.

9.6.2 Registration_Status enumeration

Registration_Status models registration status. See Table 30 for possible statuses and their meanings. This enumeration is used as a datatype for the attribute **Registration_State.registration_status** (9.4.4.3).

Table 30 — Enumeration of Registration Statuses

Status	Meaning
Incomplete	The submitter wishes to make the community that uses this registry aware of the existence of an administered item, and any associated attached items, in their local domain.
Candidate	The administered item, and any associated attached items, has been proposed for progression through the registration levels.
Recorded	The registration authority has confirmed that for the administered item, and any associated attached items: — all mandatory metadata attributes have been completed; — all mandatory associations have been instantiated.
Qualified	The registration authority has confirmed that for the administered item, and any associated attached items: — the mandatory metadata attributes have been complete; — all mandatory associations have been instantiated; and — the mandatory metadata attributes conform to applicable quality requirements.
Standard	The registration authority confirms that the administered item, and any associated attached items are:

Table 30 *(continued)*

Status	Meaning
	— of sufficient quality and — of broad interest for use in the community that uses this registry.
Preferred Standard	The registration authority confirms that the administered item, and any associated attached items are: — preferred for use within the community that uses this registry.
Superseded	The registration authority determined that the administered item, and any associated attached items are: — no longer recommended for use by the community that uses this registry, and — a successor administered item is now preferred for use.
Retired	The registration authority has approved the administered item, and any associated attached items as: — no longer recommended for use in the community that uses this registry and — should no longer be used.
Historical	The submitter wishes to make the community that uses this registry aware of the existence of an administered item, and any associated attached items, that was used in the past.
Application	The registration authority wishes to make the community that uses this metadata register aware of the existence of an administered item, and any associated attached items, in their local domain that is in an application system and is not specified at the logical level. This item can be very well described.

The use of these statuses is described more fully in ISO/IEC 11179-6:2023, 4.3.2.

10 Classification package

10.1 Overview of Classification metamodel region

The Classification package consists of one region, the Classification metamodel region, which provides a facility to register and administer classification schemes and their constituent classification scheme items. The Classification region supports the classification of registry items in one or more classification schemes.

NOTE ISO/IEC 11179-32[18] specifies a metamodel for concept system registration. Also, Annex C of that document specifies a mapping of classification schemes as specified here to concept systems as specified there.

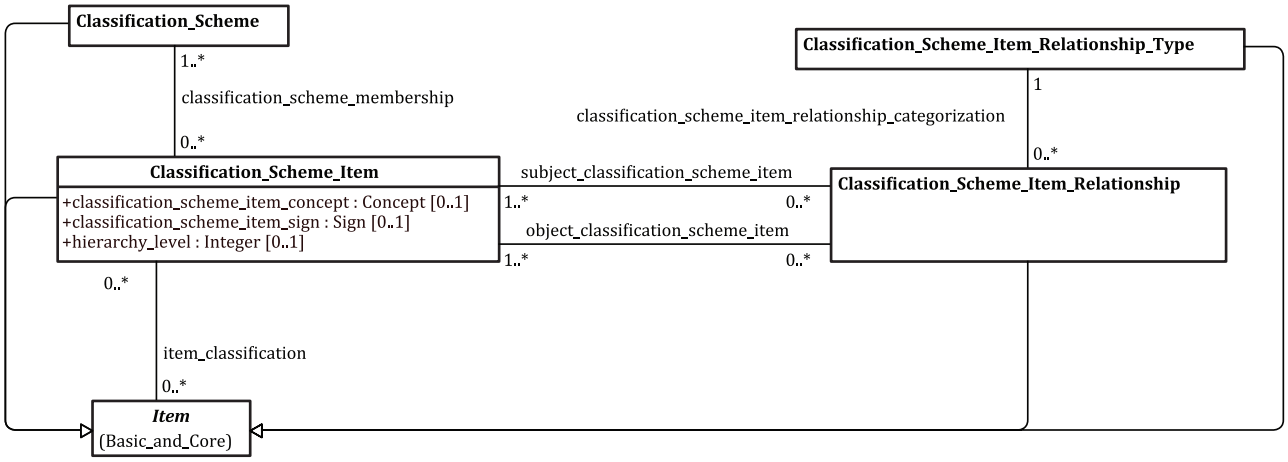

Figure 8 — Classification metamodel region

10.2 Classes referenced from the Basic and core package

10.2.1 Item class

10.2.1.1 Description of Item

Item is part of the Core metamodel and is specified in 6.4.2.1. Additional associations and subclasses are specified in this metamodel region.

10.2.1.2 Associations of Item

This metamodel region specifies the **item_classification** association (10.4.1).

Other associations are added in other metamodel regions where the **Item** class is extended:

— the Core metamodel region (6.4.2.1.2);

— the Identification metamodel region (7.2.1.2);

— the Designation and Definition metamodel region (8.2.1.2);

— the Classification metamodel region (10.2.1.2);

— the Item Mapping metamodel region (11.2.1.2);

— and in ISO/IEC 11179-31[17], ISO/IEC 11179-32[18], ISO/IEC 11179-33[19], ISO/IEC 11179-34[20] and ISO/IEC 11179-35[21].

10.2.1.3 Attributes of Item

No attributes are specified in this metamodel region.

10.3 Classes in the Classification metamodel region

10.3.1 Classification_Scheme class

10.3.1.1 Direct superclass

Classification_Scheme is a subclass of **Item** (10.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

10.3.1.2 Description of Classification_Scheme

Classification_Scheme is a class each instance of which models a classification scheme.

A classification scheme may be a taxonomy, a network, an ontology or any other terminological system. The classification may also be just a list of controlled vocabulary of property words (or terms). It is possible for the list to be taken from the “leaf level” of a taxonomy.

10.3.1.3 Associations of Classification_Scheme

Classification_Scheme has the following association:

— **classification_scheme_membership** association (10.4.2).

10.3.1.4 Attributes of Classification_Scheme

No attributes are specified in this metamodel region.

10.3.2 Classification_Scheme_Item class

10.3.2.1 Direct superclass

Classification_Scheme_Item is a subclass of **Item** (10.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

10.3.2.2 Description of Classification_Scheme_Item

Classification_Scheme_Item is a class each instance of which models a **classification scheme item**, which represents an individual item within a **Classification_Scheme** (10.3.1), as represented by the **classification_scheme_membership** association (10.4.2).

10.3.2.3 Associations of Classification_Scheme_Item

Classification_Scheme_Item has the following associations:

— **item_classification** association (10.4.1);

— **classification_scheme_membership** association (10.4.2);

— **subject_classification_scheme_item** association (10.4.3);

— **object_classification_scheme_item** association (10.4.4).

10.3.2.4 Attributes of Classification_Scheme_Item

See Table 31.

Table 31 — Attributes of Classification_Scheme_Item class

Attribute name	Multiplicity	Datatype	Description
classification_scheme_item_concept	0..1	**Concept** (6.4.2.2)	Definition: concept used as a classifier for this classification scheme item in this classification scheme
classification_scheme_item_sign	0..1	**Sign** (6.2.10)	Definition: sign used as a classifier for this classification scheme item in this classification scheme
hierarchy_level	0..1	**Integer** (6.2.5)	Definition: level of this classification scheme item in the hierarchy of the classification scheme, where the top level is level 1 and each lower level increments by 1 (2, 3, etc)”

10.3.2.5 Rules for Classification_Scheme_Item

a) **classification_scheme_item_concept** should be used when the classifier has meaning.

b) **classification_scheme_item_sign** should be used when the classifier is just a tag with no associated meaning.

c) A **Classification Scheme Item** shall have either a **classification_scheme_item_concept** or a **classification_scheme_item_sign** to be used as a classifier.

10.3.3 Classification_Scheme_Item_Relationship class

10.3.3.1 Direct superclass

Classification_Scheme_Item_Relationship is a subclass of **Item** (10.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

10.3.3.2 Description of Classification_Scheme_Item_Relationship

Classification_Scheme_Item_Relationship is a class each instance of which models a relationship among classification scheme items.

10.3.3.3 Associations of Classification_Scheme_Item_Relationship

Classification_Scheme_Item_Relationship has the following associations:

— **subject_classification_scheme_item** association (10.4.3);

— **object_classification_scheme_item** association (10.4.4);

— **classification_scheme_item_relationship_categorization** association (10.4.5).

All the above associations are required for each instance of **Classification_Scheme_Item_Relationship**.

10.3.3.4 Attributes of Classification_Scheme_Item_Relationship

No attributes are specified in this metamodel region.

10.3.4 Classification_Scheme_Item_Relationship_Type class

10.3.4.1 Direct superclass

Classification_Scheme_Item_Relationship_Type is a subclass of **Item** (10.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

10.3.4.2 Description of Classification_Scheme_Item_Relationship_Type

Classification_Scheme_Item_Relationship_Type is a class each instance of which models a type of relationship among classification_scheme_items.

EXAMPLES Parent, sibling, child.

10.3.4.3 Associations of Classification_Scheme_Item_Relationship_Type

See **classification_scheme_item_relationship_categorization** (10.4.5).

10.3.4.4 Attributes of Classification_Scheme_Item_Relationship_Type

No attributes are specified in this metamodel region.

10.3.4.5 Constraints on Classification_Scheme_Item_Relationship_Type

10.3.4.5.1 Constraint 1

Each instance of **Classification_Scheme_Item_Relationship_Type** shall be linked via the **item_designation** (8.6.4) association with an instance of **Designation** (8.4.1) which provides the name for the relationship type.

10.3.4.5.2 Constraint 2

Each instance of **Classification_Scheme_Item_Relationship_Type** shall be linked via the **item_definition** (8.6.3) association with an instance of **Definition** (8.4.2) which provides the definition for the relationship type.

10.4 Associations in the Classification metamodel region

10.4.1 item_classification association

The **item_classification** association records the binding of zero, one or more instances of the **Classification_Scheme_Item** (10.3.2) class with zero, one or more instances of the **Item** (10.2.1) class.

This association records the registry items that are to be classified as classification scheme items within a classification scheme.

10.4.2 classification_scheme_membership association

The **classification_scheme_membership** association records the binding of zero, one or more instances of the **Classification_Scheme_Item** (10.3.2) class with one or more instances of the **Classification_Scheme** (10.3.1) class.

This association records the classification scheme items that are members of a classification scheme.

10.4.3 subject_classification_scheme_item association

The **subject_classification_scheme_item** association records the bindings of zero, one or more instances of the **Classification_Scheme_Item_Relationship**s (10.3.3) class with one or more instances of the **Classification_Scheme_Item**s (10.3.2) class.

This association records that the set of classification scheme items are the subject of the relationship modelled by the instance of the **Classification_Scheme_Item_Relationship** class.

10.4.4 object_classification_scheme_item association

The **object_classification_scheme_item** association records the bindings of zero, one or more instances of the **Classification_Scheme_Item_Relationship**s (10.3.3) class with one or more instances of the **Classification_Scheme_Item**s (10.3.2) class.

This association records that the set of classification scheme items are the object of the relationship modelled by the instance of the **Classification_Scheme_Item_Relationship** class.

10.4.5 classification_scheme_item_relationship_categorization association

The **classification_scheme_item_relationship_categorization** association records the bindings of zero, one or more instances of the **Classification_Scheme_Item_Relationship**s (10.3.3) class with exactly one instance of the **Classification_Scheme_Item_Relationship_Type** class.

This association records the categorization, as modelled by the instance of the **Classification_Scheme_Item_Relationship_Type** class, of the relationship between the subject and object sets of classification scheme items that is modelled by the instance of the **Classification_Scheme_Item_Relationship** class.

11 Item_Mapping package

11.1 Overview of the Item_Mapping metamodel region

The Item_Mapping package consists of one region, the Item_Mapping metamodel region, which specifies a facility to map among registry items in a registry that uses the common facilities specified in this document.

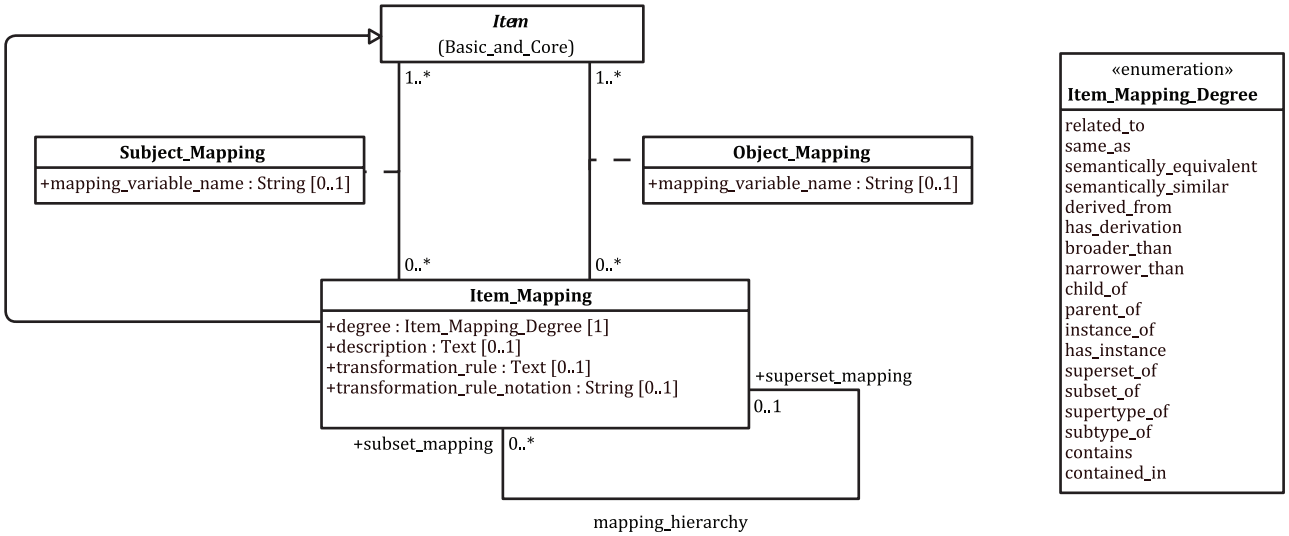

Figure 9 — Item mapping

11.2 Classes referenced from the Basic and core package

11.2.1 Item class

11.2.1.1 Description of Item

Item is part of the Core metamodel and is specified in 6.4.2.1. It is extended here in the Item Mapping metamodel. It is further extended in the Identification region (see Clause 7), the Designation and Definition region (see Clause 8), the Classification region (see Clause 10) and indirectly in the Registration region (see Clause 9).

11.2.1.2 Associations of Item

This metamodel region specifies the following association classes:

— **Subject_Mapping** association class (11.4.1);

— **Object_Mapping** association class (11.4.2).

Other associations are added in other metamodel regions where the **Item** class is extended:

— the Core metamodel region (6.4.2.1.2);

— the Identification metamodel region (7.2.1.2);

— the Designation and Definition metamodel region (8.2.1.2);

— the Classification metamodel region (10.2.1.2);

— and in ISO/IEC 11179-31[17], ISO/IEC 11179-32[18], ISO/IEC 11179-33[19], ISO/IEC 11179-34[20] and ISO/IEC 11179-35[21].

11.2.1.3 Attributes of Item

No attributes are specified in this metamodel region.

11.3 Classes in the Mapping metamodel region

11.3.1 Item_Mapping class

11.3.1.1 Direct superclass

Item (11.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

11.3.1.2 Description of Item_Mapping

Item_Mapping is a class each instance of which models some part of a mapping of registry items. Each instance of the **Item_Mapping** class comprises a **Subject_Mapping** (11.4.1) set of instances of the **Item** class (11.2.1) and an **Object_Mapping** (11.4.2) set of instances of the **Item** class that model similar real-world requirements.

11.3.1.3 Associations of Item_Mapping

See:

— **Subject_Mapping** association class (11.4.1);

— **Object_Mapping** association class (11.4.2);

— **mapping_hierarchy** association (11.5.1).

11.3.1.4 Attributes of Item_Mapping

See Table 32.

Table 32 — Attributes of *Item_Mapping* class

Attribute name	Multiplicity	Datatype	Description
degree	1	**Item_Mapping_Degree** (11.6.1)	Definition: specification of the kind of mapping between sets of registry items linked to this instance of **Item_Mapping**
description	0..1	**Text** (6.2.12)	Definition: additional explanation of the kind of mapping between sets of registry items linked to this instance of **Item_Mapping**, for use especially the value of the **degree** attribute is specified as 'related to'.
transformation_rule	0..1	**Text** (6.2.12)	Definition: rule for transforming the subject items into the object items

Table 32 *(continued)*

Attribute name	Multiplicity	Datatype	Description
transformation_rule_notation	0..1	**Text** (6.2.12)	Conditional. Definition: specification of the notation used to specify the transformation rule EXAMPLES EBNF[22], XCL[31] Condition: Required if **transformation_rule** is present.

Annex B lists examples of each kind of mapping degree and includes detailed examples of some of them.

11.4 Association Classes in the Mapping metamodel region

11.4.1 Subject_Mapping association class

11.4.1.1 Description of Subject_Mapping

Subject_Mapping is an association which records the binding of zero, one or more instances of the **Item_Mapping** class (11.3.1) with one or more instances of the **Item** class (6.4.2.1) which are the subject of the mapping.

Subject_Mapping is also a class with one attribute as specified in 11.4.1.2.

11.4.1.2 Attributes of Subject_Mapping

See Table 33.

Table 33 — Attributes of Subject_Mapping association class

Attribute name	Multiplicity	Datatype	Description
mapping_variable_name	0..1	**String** (6.2.11)	Definition: name of the variable used in the **Subject_Mapping** in the **transformation_rule**. EXAMPLE See B.3.3 which illustrates the variables: 'day', 'month' and 'year' in the mapping of a date to its component data elements.

11.4.2 Object_Mapping association class

11.4.2.1 Description of Object_Mapping

Object_Mapping is an association which records the binding of zero, one or more instances of the **Item_Mapping** class (11.3.1) with one or more instances of the **Item** class (6.4.2.1) which are the object of the mapping.

Object_Mapping is also a class with one attribute as specified in 11.4.2.2.

11.4.2.2 Attributes of Object_Mapping

See Table 34.

Table 34 — Attributes of Object_Mapping association class

Attribute name	Multiplicity	Datatype	Description
mapping_variable_name	0..1	**String** (6.2.11)	Definition: name of the variable used in the **Object_Mapping** in the **transformation_rule** EXAMPLE See B.3.3 which illustrates the variables: 'day', 'month' and 'year' in the mapping of a date to its component data elements.

11.5 Associations in the Item Mapping metamodel region

11.5.1 mapping_hierarchy association

The **mapping_hierarchy** association is a reflexive association which records the binding of zero or one instance of the **Item_Mapping** (11.3.1) class, each of which represents a superset mapping, to zero, one or more other instances of the **Item_Mapping** class, each of which represents a subset mapping. This association registers any superset-subset hierarchy that exists within a group of the mappings, e.g. when two models and their model elements are mapped together, the instance of the **Item_Mapping** class that represents the top-level mapping of the models will be at the superset end of this association and the instances of the **Item_Mapping** class that represent the lower-level mappings of sets of model elements be at the subset end of this association.

11.6 Datatypes in the Mapping metamodel region

11.6.1 Item_Mapping_Degree enumeration

Item_Mapping_Degree is an enumerated datatype with values as shown in Table 35.

Table 35 — Enumeration of Item_Mapping_Degree values

Value	Description
related to	Indicates that the creator of the mapping has asserted that the subject item(s) and the object item(s) share some form of relationship other than those explicitly defined by the other values of **Item_Mapping_Degree**, and as further explained in the **description**.
same as	Indicates that the creator of the mapping has asserted that the subject item(s) and the object item(s) have the same semantics and syntax. (I.e., they represent the same thing, though they necessarily have different identifiers, and may have different names.)
semantically equivalent	Indicates that the creator of the mapping has asserted that the subject item(s) and the object item(s) have the same semantics.
semantically similar	Indicates that the creator of the mapping has asserted that the subject item(s) and the object item(s) are semantically similar although not identical.
derived from	Indicates that the creator of the mapping has asserted that the subject item(s) have been used in the creation of the object item(s). Inverse of "has derivation".
has derivation	Indicates that the creator of the mapping has asserted that the object item(s) have been used in the creation of the subject item(s). Inverse of "derived from".
broader than	Indicates that the creator of the mapping has asserted that the subject item(s) and the object item(s) are linked, and that the object item(s) have a broader meaning than the subject item(s). Inverse of "narrower than".
narrower than	Indicates that the creator of the mapping has asserted that the subject item(s) and the object item(s) are linked, and that the object item(s) have a narrower meaning than the subject item(s). Inverse of "broader than".
child of	Indicates that the creator of the mapping has asserted that the subject item(s) and the object item(s) participate in a hierarchical relationship, and that the object item(s) are one level below the subject item(s) in the hierarchy. Inverse of "parent of".

Table 35 *(continued)*

Value	Description
parent of	Indicates that the creator of the mapping has asserted that the subject item(s) and the object item(s) participate in a hierarchical relationship, and that the object item(s) are one level above the subject item(s) in the hierarchy. Inverse of "child of".
instance of	Indicates that the creator of the mapping has asserted that the object item(s) are an instantiation of the subject item(s). Inverse of "has instance".
has instance	Indicates that the creator of the mapping has asserted that the object item(s) have an instantiation in the subject item(s). Inverse of "instance of".
superset of	Indicates that the creator of the mapping has asserted that the subject item(s) and the object item(s) participate in a superset/subset relationship, and that the object item(s) are a superset of the subject item(s). Inverse of "subset of".
subset of	Indicates that the creator of the mapping has asserted that the subject item(s) and the object item(s) participate in a superset/subset relationship, and that the object item(s) are a subset of the subject item(s). Inverse of "superset of".
supertype of	Indicates that the creator of the mapping has asserted the subject item(s) are an implementation of the object item(s) and additionally that the object item(s) satisfy requirements for participating in computing operation(s) in some framework which will also compute on the subject item(s). Inverse of "subtype of".
subtype of	Indicates that the creator of the mapping has asserted the object item(s) are an implementation of the subject item(s) and additionally that the subject item(s) satisfy requirements for participating in computing operation(s) in some framework which will also compute on the object item(s). Inverse of "supertype of".
contains	Indicates that the creator of the mapping has asserted that the subject item(s) and the object item(s) participate in a containment relationship, and that the object item(s) contain the subject item(s). Inverse of "contained in".
contained in	Indicates that the creator of the mapping has asserted that the subject item(s) and the object item(s) participate in a containment relationship, and that the object item(s) are a contained in the subject item(s). Inverse of "contains".

Item_Mapping_Degree is used as the datatype of the **degree** attribute of **Item_Mapping** (11.3.1).

A conforming implementation may extend the above list of values.

Annex A
(informative)

Consolidated class hierarchy

Figure A.1 shows all classes that participate in a class hierarchy in this document. Classes that do not participate in a class hierarchy are not shown.

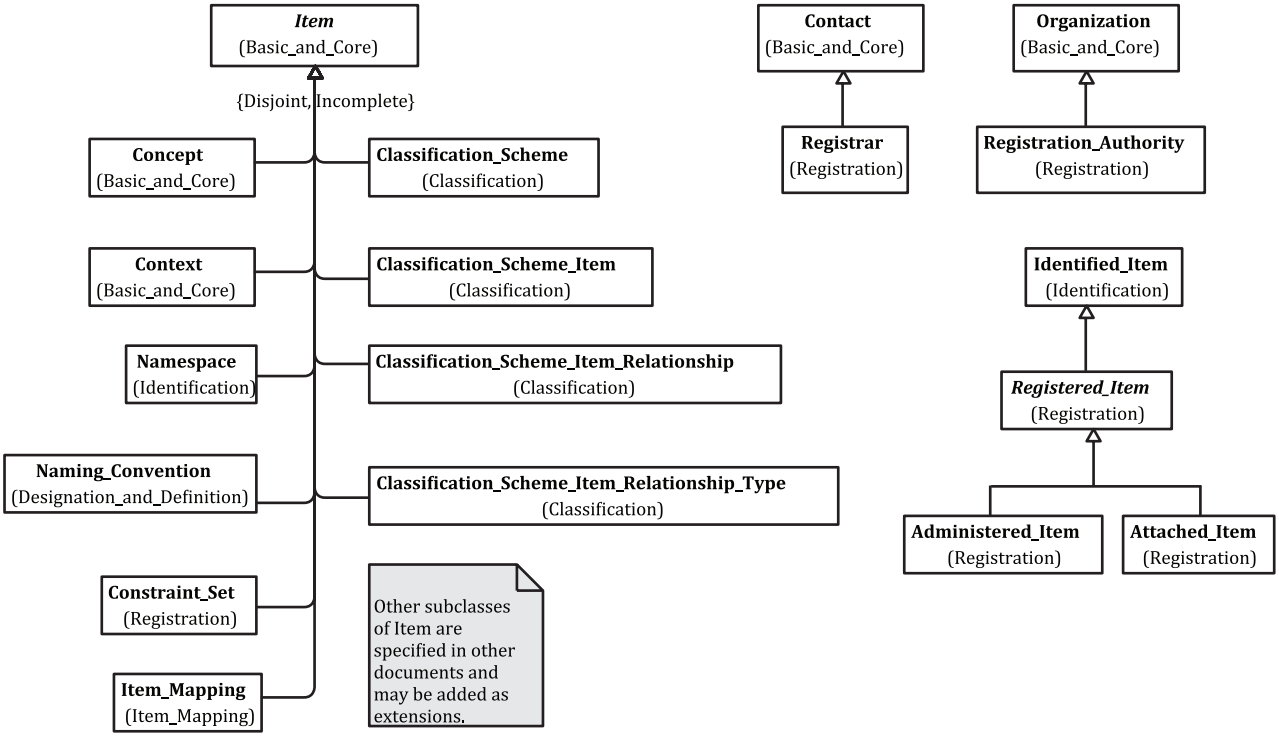

Figure A.1 — Consolidated class hierarchy

Annex B
(informative)

Illustrations of Item_Mapping

B.1 Examples of where various mapping degrees can be used

Table B.1 provides an example of each mapping degree.

Table B.1 — Examples of Mapping Degree usage

Mapping Degree	Example usage
related_to	Use for a relationship that is not covered by any of the more specific mapping degree types. Use the **description** attribute to explain the relationship further.
same_as	Use where an item has been registered twice, perhaps from different sources, but they are really the same thing.
semantically_ equivalent	Use where two items are distinct, but they mean the same. Perhaps data elements in different application systems.
semantically_similar	Use where two items are similar but not identical. E.g. Value domains
derived_from has_derivation	It is possible to use this for data elements, though ISO/IEC 11179-31[17] has specific derivation associations for that. Use the **transformation_rule** attribute to show how the derivation is made.
broader_than narrower_than	Typically used for concepts.
child_of parent_of	Example: Mapping of value domains to permissible values.
instance_of has_instance	Use for mapping across levels: type to instance.
superset_of subset_of	Use to relate sets, e.g. value domains, classification schemes.
supertype_of subtype_of	Use to relate classes.
contains contained_in	Example: Data elements contained in a data set specification.

B.2 Example of 'Same_as' mapping degree

B.2.1 Overview of example

This example shows the use of Item Mapping to record that two data elements represent the same thing.

Consider two data elements, "Individual Birthdate" in a HR system and "Person Date of Birth" in a Production system. Both of these are registered in an MDR as instances of the **Data_Element** class (see ISO/IEC 11179-31:2023, 7.6.2.1[17]), both of which are linked to instances of the **Administered_Item** class (9.4.2) via the **item_identification** (7.4.1) association.

B.2.2 Items used in this example

The view of the metamodel in Table B.2 has been flattened for simplicity. It includes attributes from the Identification region (**identifier**), the Designation and Definition region (**name**, **definition**) and **Item** sub-classing (**type**).

Table B.2 — Table of Items used in 'same as' example

identifier	type	name	definition
B_2_1	**Data_Element**	Individual Birthdate	Birth date of an individual.
B_2_2	**Data_Element**	Person Date of Birth	Birth date of a person.
B_2_3	**Item_Mapping**	Birth Date Mapping	Mapping of HR birthdate to production birthdate

NOTE 1 Since **Item_Mapping** is itself an **Item**, it can be administered, with a submission record to show its origin, and reference documents to provide additional detail.

NOTE 2 **Data_Element** is defined in ISO/IEC 11179-31:2023, 7.6.2.1[17].

B.2.3 Object diagram for this example

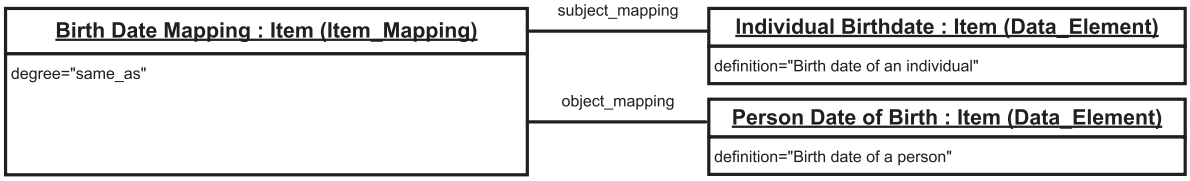

Figure B.1 — Object Diagram for the 'same as' mapping example

B.2.4 Details of Item_Mapping for this example

Table B.3 represents the instance of the **Item_Mapping** class for the birthdate_mapping and associated **Subject_Mapping** and **Object_Mapping** association classes.

NOTE Only those attributes with values are shown, and the associations are represented as references to Item.identifiers.

Table B.3 — Item_Mapping for 'Birth Date_Mapping'

Item_Mapping Item.identifier	degree	Subject_Mapping Item.identifier	Object_Mapping Item.identifier
B_2_3	same as	B_2_1	B_2_2

B.3 Example of 'derived_from' mapping degree

B.3.1 Overview of example

This example shows the use of **Item_Mapping** to record deriving a DATE data type from component DAY, MONTH, YEAR data types. Two mappings are shown, one using the ISO 8601 basic format (YYYYMMDD), the other using the ISO 8601 extended format (YYYY-MM-DD).

B.3.2 Items used in this example

The view of the metamodel in Table B.4 has been flattened for simplicity. It includes attributes from Identification, Designation, Definition and **Item** sub-classing.

Table B.4 — Table of Items used in 'derived_from' example

identifier	type	name	definition
B_3_1	**Datatype**	Day	Day of month as two digits.
B_3_2	**Datatype**	Month	Month as two digits.

Table B.4 *(continued)*

identifier	type	name	definition
B_3_3	**Datatype**	Year	Year as four digits.
B_3_4	**Datatype**	isodate1	A date in ISO 8601 basic format (YYYYMMDD)
B_3_5	**Datatype**	isodate2	A date in ISO 8601 extended format (YYYY-MM-DD)
B_3_6	**Item_Mapping**	isodate_mapping1	Mapping of an ISO 8601 basic format date from component datatypes
B_3_7	**Item_Mapping**	isodate_mapping2	Mapping of an ISO 8601 extended format date from component datatypes

NOTE 1 Since **Item_Mapping** is itself an **Item**, it can be administered, with a submission record to show its origin, and reference documents to provide additional detail.

NOTE 2 **Datatype** is defined in ISO/IEC 11179-31:2023, 7.4.2.14[17].

B.3.3 Object diagram for this example

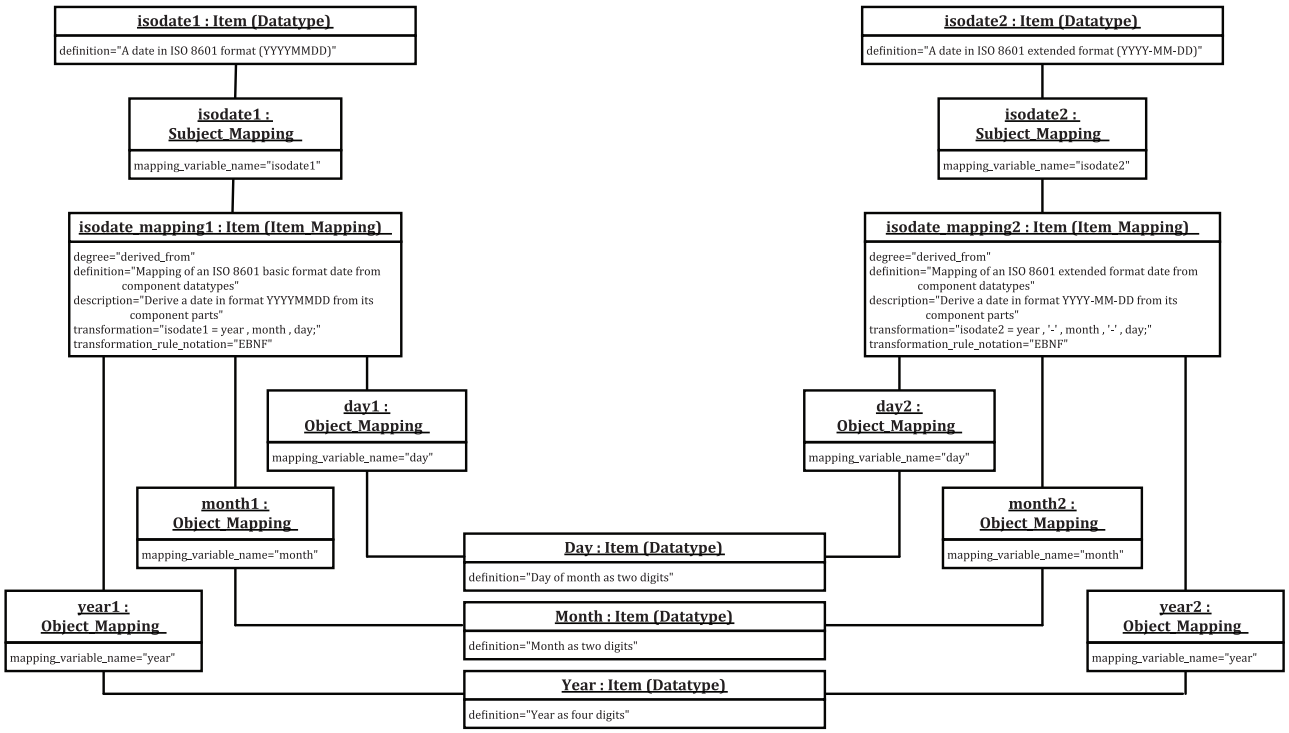

Figure B.2 — Object Diagram for the 'derived from' mapping example

Figure B.2 illustrates the item mapping, which is expanded on in B.3.4.

B.3.4 Details of Item_Mappings for this example

Table B.5 represents the instance of the **Item_Mapping** class for the isodate_mapping1.

Table B.5 — Item_Mapping class 'isodate_mapping1'

Attribute name	Attribute value
Item.identifier	B_3_6
degree	derived from
description	Derive a date in format YYYYMMDD from its component parts.

Table B.5 *(continued)*

Attribute name	Attribute value
transformation_rule	isodate1 = year , month , day ;
transformation_rule_notation	EBNF

Table B.6 represents the **Subject_Mapping** association class for the isodate_mapping1.

Table B.6 — Table of Subject_Mappings for isodate_mapping1

Item_Mapping.identifier	Item.identifier	mapping_variable_name
B_3_6	B_3_3	year
B_3_6	B_3_2	month
B_3_6	B_3_1	day

Table B.7 represents the **Object_Mapping** association class for the isodate_mapping1.

Table B.7 — Table of Object_Mappings for isodate_mapping1

Item_Mapping.identifier	Item.identifier	mapping_variable_name
B_3_6	B_3_4	isodate1

Table B.8 represents the instance of the **Item_Mapping** class for the isodate_mapping2.

Table B.8 — Item_Mapping class 'isodate_mapping2'

Attribute name	Attribute value
Item.identifier	B_3_7
degree	derived from
description	Derive a date in format YYYY-MM-DD from its component parts.
transformation_rule	isodate2 = year , '-', month , '-' , day ;
transformation_rule_notation	EBNF

Table B.9 represents the **Subject_Mapping** association class for the isodate_mapping2.

Table B.9 — Table of Subject_Mappings for isodate_mapping2

Item_Mapping.identifier	Item.identifier	mapping_variable_name
B_3_7	B_3_3	year
B_3_7	B_3_2	month
B_3_7	B_3_1	day

Table B.10 represents the **Object_Mapping** association class for the isodate_mapping2.

Table B.10 — Table of Object_Mappings for isodate_mapping2

Item_Mapping.identifier	Item.identifier	mapping_variable_name
B_3_7	B_3_5	isodate2

B.4 Example of 'Semantically_equivalent' mapping degree

B.4.1 Overview of example

This example shows the use of Item Mapping to record that two models using different notations are semantically equivalent. The example illustrates the mapping between an information model expressed as a UML Class Diagram used by System A (see Figure B.3), and an equivalent information model expressed using the IDEF1X notation and used by System B (see Figure B.4).

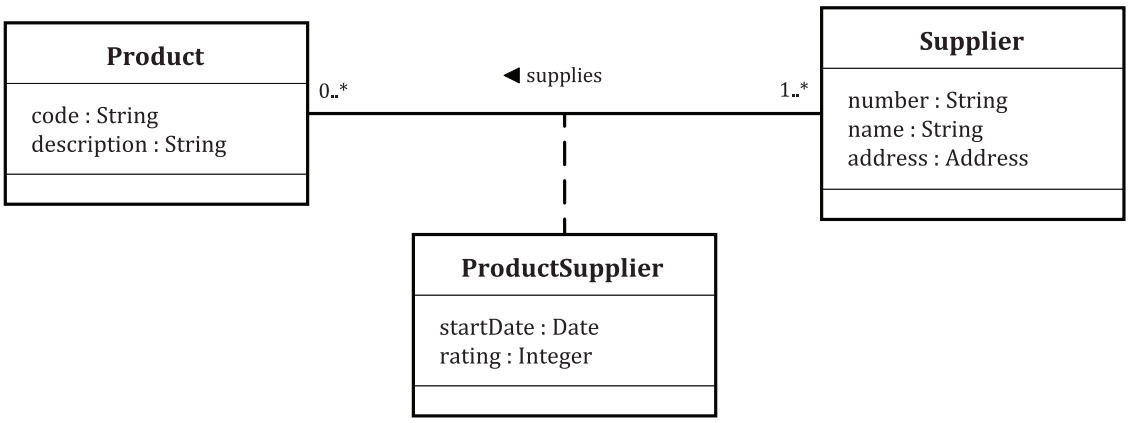

Figure B.3 — Example UML Class Diagram for the Product Supplier concept (as used by System A)

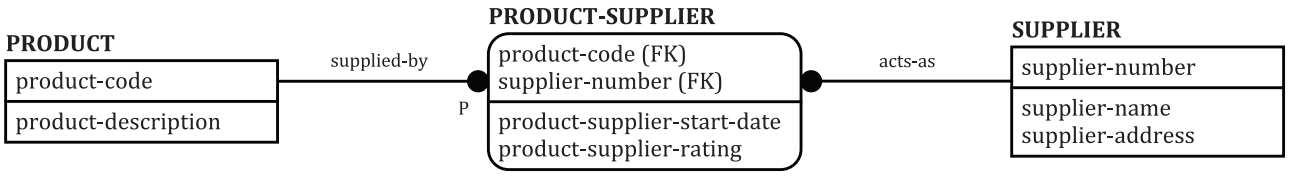

Figure B.4 — Example IDEF1X Model for the Product Supplier concept (as used by System B)

B.4.2 Items used in this example

The view of the metamodel in Table B.11 has been flattened for simplicity. It includes attributes from Identification, Designation, Definition and **Item** sub-classing.

Table B.11 — Table of Items used in this example

identifier	type	name	definition
B_4_1	**Model**	System A Product-Supplier	UML class diagram for Product Supplier concept
B_4_2	**Model_Element**	System A Product class	UML class representing the concept 'Product'
B_4_3	**Model_Element**	System A Supplier class	UML class representing the concept 'Supplier'
B_4_4	**Model_Element**	System A Product-Supplier association class	UML association class representing the concept 'Product-Supplier'
B_4_5	**Model**	System B Product-Supplier	IDEF1X Model for the Product Supplier concept
B_4_6	**Model_Element**	System B Product entity	IDEF1X independent entity representing the concept 'Product'

Table B.11 *(continued)*

identifier	type	name	definition
B_4_7	**Model_Element**	System B Supplier entity	IDEF1X independent entity representing the concept 'Supplier'
B_4_8	**Model_Element**	System B Product-Supplier entity	IDEF1X dependent entity representing the concept 'Product-Supplier'
B_4_9	**Item_Mapping**	model mapping	A mapping of the model of system A to the model of system B.
B_4_10	**Item_Mapping**	product mapping	A mapping of the Product class in System A to the Product entity in System B
B_4_11	**Item_Mapping**	supplier mapping	A mapping of the Supplier class in System A to the Supplier entity in System B
B_4_12	**Item_Mapping**	product-supplier mapping	A mapping of the Product-Supplier association class in System A to the Product-Supplier dependent entity in System B

NOTE 1 Since **Item_Mapping** is itself an **Item**, it can be administered, with a submission record to show its origin, and reference documents to provide additional detail.

NOTE 2 **Model** is defined in ISO/IEC 11179-35:2023, 7.2.2.2[21].

NOTE 3 **Model_Element** is defined in ISO/IEC 11179-35:2023, 7.2.2.3[21].

NOTE 4 A full representation of each model would also include **Model_Element**s representing each attribute within each class or entity and the associations/relationships among classes/entities. These have been omitted to simplify the example.

B.4.3 Object diagram for this example

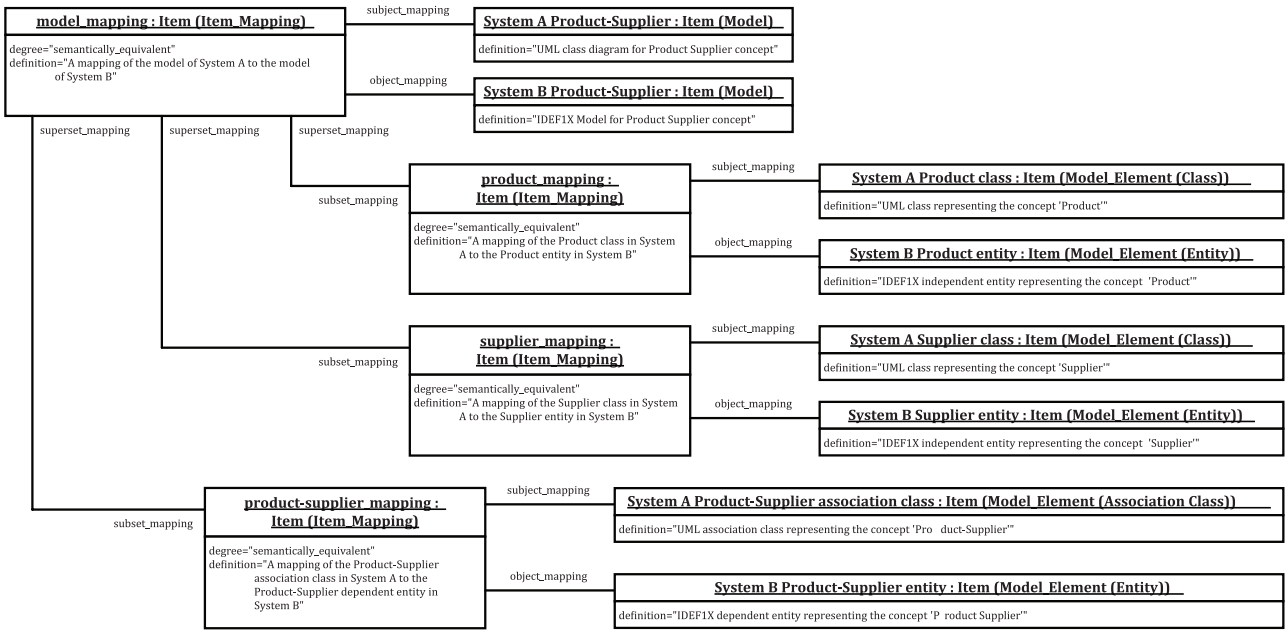

Figure B.5 — Object Diagram for the 'semantically equivalent' mapping example

B.4.4 Details of Item_Mapping for this example

Models can be mapped at different levels of detail, depending on the requirements of the organization doing the mapping. At the highest level, the models can be mapped as a whole. Saying that the whole model is semantically equivalent, can suffice for some purposes. Model elements can also be mapped. In this example, we map only at the entity/class level, but attributes can also be mapped for completeness.

Table B.12 represents the instances of the **Item_Mapping** class for the model mapping and model element mappings.

NOTE Only those attributes with values are shown, and the associations are represented as references to **Item.identifier**s. The mapping hierarchy is shown in Table B.13.

Table B.12 — Table of Item_Mappings for this example

Item_Mapping Item.identifier	degree	Subject_Mapping Item.identifier	Object_Mapping Item.identifier
B_4_9	semantically equivalent	B_4_1	B_4_5
B_4_10	semantically equivalent	B_4_2	B_4_6
B_4_11	semantically equivalent	B_4_3	B_4_7
B_4_12	semantically equivalent	B_4_4	B_4_8

Table B.13 — Table of Mapping_Hierarchy associations

superset_mapping identifier	subset_mapping identifier
B_4_9	B_4_10
B_4_9	B_4_11
B_4_9	B_4_12

B.5 Example of 'Semantically similar' mapping of Enumerated Conceptual Domains

B.5.1 Overview of example

Consider these two enumerated conceptual domains:

1) The list of special meals that can be ordered by British Airways passengers: "Vegetarian Hindu", "Lacto ovo vegetarian", "Vegan vegetarian", "Hindu". "Muslim" and "Kosher".

2) The similar list of religious meals that can be ordered by Singapore Airlines passengers: "Kosher Meal", "Muslim Meal", "Hindu Meal", "Raw Vegetarian Meal", "Vegetarian Oriental Meal", "Vegetarian Indian Meal", "Vegetarian Jain Meal", "Western Vegetarian Meal" and "Vegetarian Vegan Meal".

Both of these enumerated conceptual domains are registered in an MDR as instances of the **Enumerated_Conceptual_Domain** class, with the meals forming the value meanings registered as instances of the **Value_Meaning** class. The instances of the **Value_Meaning** class are then linked to the respective instances of the **Enumerated_Conceptual_Domain** class. Both the instances of the **Enumerated_Conceptual_Domain** class and all of the instances of the **Value_Meaning** class are also instances of the **Item** class.

B.5.2 Items used in this example

The view of the metamodel in Table B.14 has been flattened for simplicity. It includes attributes from Identification, Designation, Definition and **Item** sub-classing.

Table B.14 — Table of Items used in this example

identifier	type	name	definition
B_5_CD1	**Enumerated_ Conceptual_Domain**	BA menu	The list of special meals that can be ordered by British Airways passengers
B_5_CD1_VM1	**Value_Meaning**	Vegetarian Hindu	Does not contain fish, shellfish, meat, poultry or eggs. It is a meatless meal and spicy in content.
B_5_CD1_VM2	**Value_Meaning**	Lacto ovo vegetarian	Does not contain meat, fish or seafood. May contain dairy products such as milk, butter, cheese and eggs, etc.
B_5_CD1_VM3	**Value_Meaning**	Vegan	Does not contain meat, fish, fowl, eggs, honey, dairy products or derivatives.
B_5_CD1_VM4	**Value_Meaning**	Hindu	Does not contain beef, beef derivatives, veal or pork. It is not a meatless meal.
B_5_CD1_VM5	**Value_Meaning**	Muslim	Does not contain pork, by-products of pork or foodstuffs containing alcohol. All meats come from ritually slaughtered animals.
B_5_CD1_VM6	**Value_Meaning**	Kosher	These meals are prepared to comply with Jewish dietary laws.
B_5_CD2	**Enumerated_ Conceptual_Domain**	Singapore Airlines menu	The list of special meals that can be ordered by Singapore Airlines passengers
B_5_CD2_VM1	**Value_Meaning**	Kosher Meal	Pre-packed and sealed; contains meat.
B_5_CD2_VM2	**Value_Meaning**	Muslim Meal	No alcohol/pork/ham/bacon.
B_5_CD2_VM3	**Value_Meaning**	Hindu Meal	No beef, veal, pork, smoked or raw fish but can contain other types of meat.
B_5_CD2_VM4	**Value_Meaning**	Raw Vegetarian Meal	Only raw fruits and vegetables.
B_5_CD2_VM5	**Value_Meaning**	Vegetarian Oriental Meal	No meat or seafood of any sort; can contain dairy products; cooked Chinese-style.
B_5_CD2_VM6	**Value_Meaning**	Vegetarian Indian Meal (non-strict)	No meat of any sort; can contain dairy products; cooked Indian style.
B_5_CD2_VM7	**Value_Meaning**	Vegetarian Jain Meal (strict; suitable for Jain)	No meat of any sort; no onion, garlic, ginger and all root vegetables; cooked Indian style.
B_5_CD2_VM8	**Value_Meaning**	Western Vegetarian Meal (non-strict; ovo-lacto)	No meat of any sort; can contain dairy products; cooked western style.
B_5_CD2_VM9	**Value_Meaning**	Vegan Meal (strict)	No meat of any sort; no dairy products; cooked western style.
B_5_IM1	**Item_Mapping**	Menu mapping	Mapping of BA menu CD to Singapore airlines menu CD.

Table B.14 *(continued)*

identifier	type	name	definition
B_5_IM2	**Item_Mapping**	Indian vegetarian mapping	Mapping of BA Vegetarian Hindu meal to Singapore Airlines Indian vegetarian meals
B_5_IM3	**Item_Mapping**	Western vegetarian mapping	Mapping of BA Lacto ovo vegetarian meal to Singapore Airlines western vegetarian meals
B_5_IM4	**Item_Mapping**	Vegan meal mapping	Mapping of BA vegan meal to Singapore Airlines Vegan Meal (strict) and Raw Vegetarian Meal
B_5_IM5	**Item_Mapping**	Hindu meal mapping	Mapping of BA Hindu meal to Singapore Airlines Hindu meal
B_5_IM6	**Item_Mapping**	Muslim meal mapping	Mapping of BA Muslim meal to Singapore Airlines Muslim meal
B_5_IM7	**Item_Mapping**	Kosher meal mapping	Mapping of BA Kosher meal to Singapore Airlines Kosher meal

B.5.3 Object diagram for this example

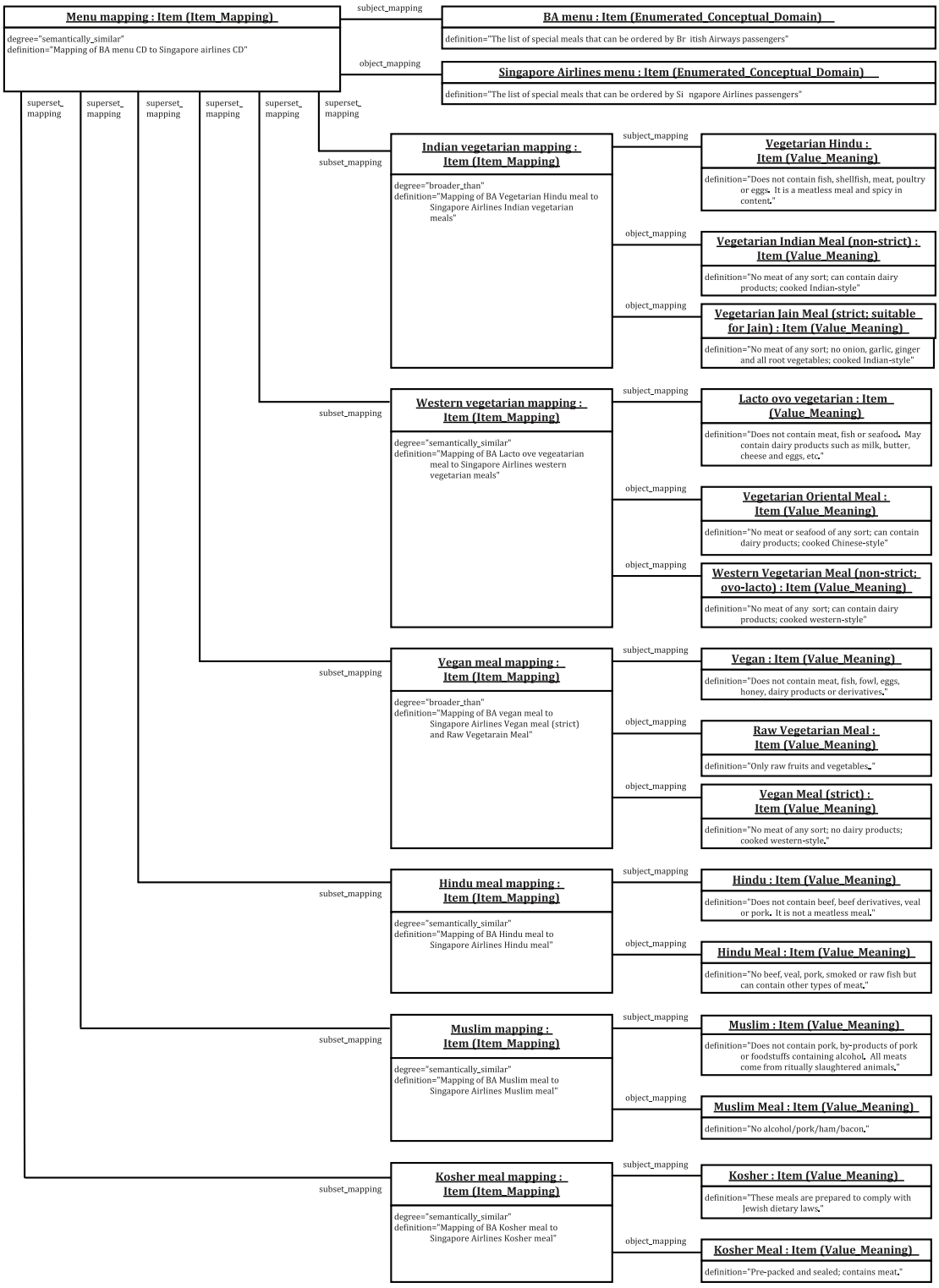

Figure B.6 — Object Diagram for the ‘semantically similar’ mapping example

B.5.4 Details of Item_Mapping for this example

Table B.15 represents the instances of the **Item_Mapping** class for the model mapping and model element mappings.

NOTE Only those attributes with values are shown, and the associations are represented as references to Item.identifiers. The mapping hierarchy is shown in Table B.16.

Table B.15 — Table of Item_Mappings for this example

Item_Mapping Item.identifier	degree	Subject_Mapping Item.identifier	Object_Mapping Item.identifier
B_5_IM1	semantically_similar	B_5_CD1	B_5_CD2
B_5_IM2	broader_than	B_5_CD1_VM1	B_5_CD2_VM6, B_5_CD2_VM7
B_5_IM3	semantically_similar	B_5_CD1_VM2	B_5_CD2_VM5, B_5_CD2_VM8
B_5_IM4	broader_than	B_5_CD1_VM3	B_5_CD2_VM4, B_5_CD2_VM9
B_5_IM5	semantically_similar	B_5_CD1_VM4	B_5_CD2_VM3
B_5_IM6	semantically_similar	B_5_CD1_VM5	B_5_CD2_VM2
B_5_IM7	semantically_similar	B_5_CD1_VM6	B_5_CD2_VM1

Table B.16 — Table of Mapping_Hierarchy association

superset_mapping identifier	subset_mapping identifier
B_5_IM1	B_5_IM2
B_5_IM1	B_5_IM3
B_5_IM1	B_5_IM4
B_5_IM1	B_5_IM5
B_5_IM1	B_5_IM6
B_5_IM1	B_5_IM7

Annex C
(informative)

Example of Registering a simple Conceptual Domain

C.1 Overview of the example

Figure C.1 shows an object model representing the registration of a simple conceptual domain. Each of the objects are described below. In the figure, the shaded objects come from ISO/IEC 11179-31. All other objects come from this document.

C.1.1 Context1

Context1 is an instance of **Context** which is a subclass of **Item** and provides the default context for the registry in this registration example.

C.1.2 Designation1

Designation1 is an instance of **Designation** which provides the sign (or name) “Default Context for My Registry” to Context1 through the link **item_designation**. Context1 in turn provides the context for Designation1 through the link **designation_context**.

C.1.3 Context2

Context2 is an instance of **Context** which is a subclass of **Item** and provides the context for the items being registered in this registration example.

C.1.4 Designation2

Designation2 is an instance of **Designation** which provides the sign (or name) “Human Physiology” to Context2 through the link **item_designation**. Context1 provides the context for Designation2 through the link **designation_context**.

C.1.5 Designation_Context1

Designation_Context1 is an instance of **Designation_Context** through which Context2 provides context to Designation3, with **acceptability** = “preferred”.

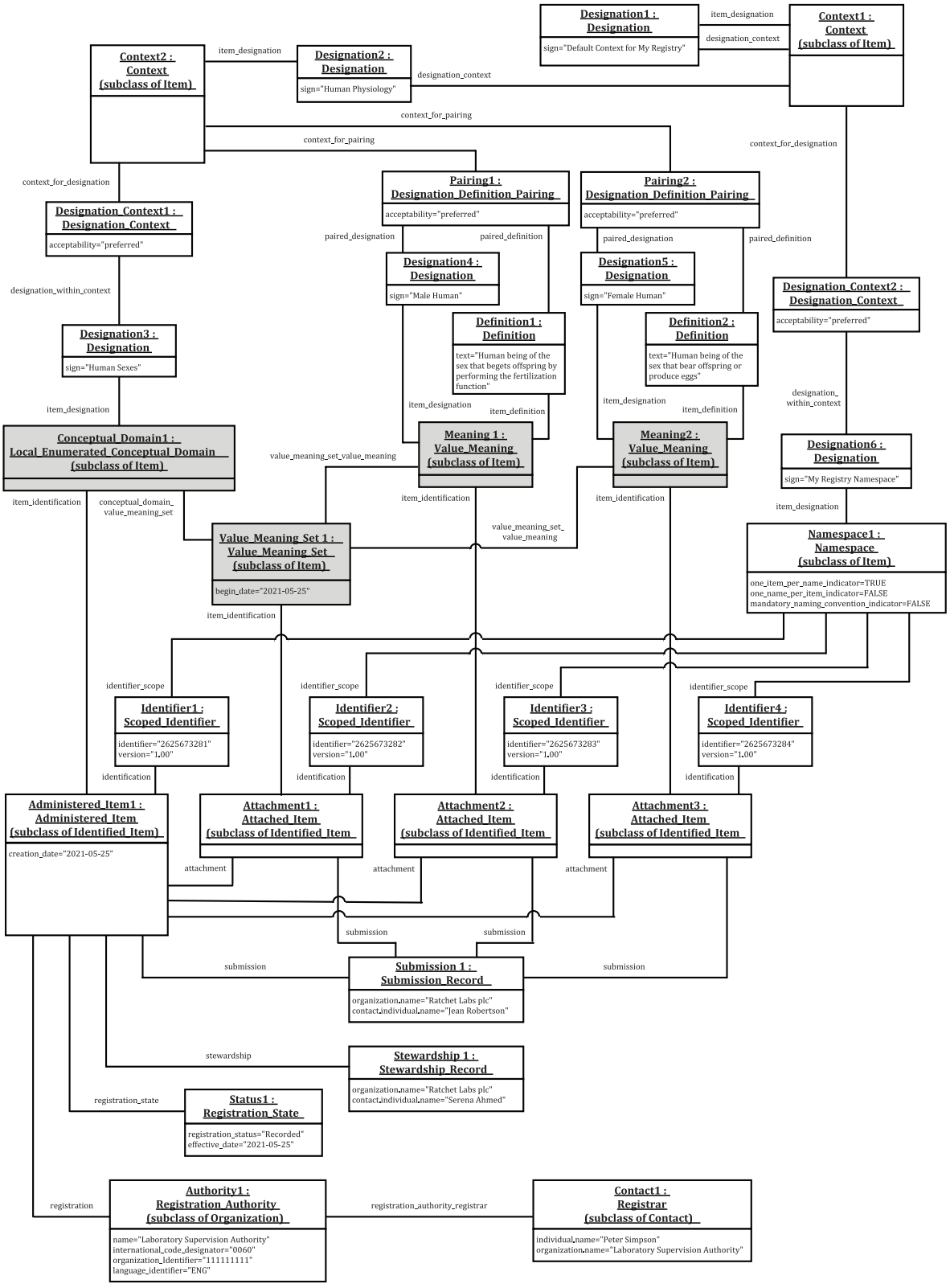

Figure C.1 — Example object model of Conceptual Domain registration

C.1.6 Designation3

Designation3 is an instance of **Designation** which provides the sign (or name) “Human Sexes” to Conceptual_Domain1.

C.1.7 Conceptual_Domain1

Conceptual_Domain1 is an instance of **Local_Enumerated_Conceptual_Domain** which is a subclass of **Item**. Conceptual_Domain1 is the main item to be registered in this example, along with its constituent **Value_Meaning**s.

C.1.8 Administered_Item1

Administered_Item1 is an instance of **Administered_Item** which is a subclass of **Registered_Item** and **Identified_Item**. Administered_Item1 provides identification through the link **item_identification** to Conceptual_Domain1 and the **identification** link to Identifier1.

C.1.9 Identifier1

Identifier1 is an instance of **Scoped_Identifier** which provides the **identifier** "2625673281" and **version** "1.00" to Administered_Item1 and through that to Conceptual_Domain1.

C.1.10 Authority1

Authority1 is an instance **Registration_Authority** which is a subclass of **Organization**. Authority1 is responsible for the registration of *Administered_Item1* through the link **registration**.

C.1.11 Contact1

Contact1 is an instance of **Registrar** (a subclass of **Contact**) which is a representative for the **Registration_Authority** Authority1, as recorded by the link **registration_authority_registrar**.

C.1.12 Status1

Status1 is an instance of **Registration_State** which provides the **registration_status** "Recorded" and **effective_date** "2021-05-25" for Administered_Item1 through the link **registration_state**.

C.1.13 Stewardship1

Stewardship1 is an instance of **Stewardship_Record** which records the **organization** and **contact** which provide stewardship for Administered_Item1 through the link **stewardship**.

C.1.14 Submission1

Submission1 is an instance of **Submission_Record** which records the **organization** and **contact** which submitted Administered_Item1 as recorded through the link **submission**.

C.1.15 Attachment1

Attachment1 is an instance of **Attached_Item** which is a subclass of **Registered_Item** and **Identified_Item**. Attachment1 provides identification through the link **item_identification** to Value_Meaning_Set1 and the **identification** link to Identifier2.

C.1.16 Identifier2

Identifier2 is an instance of **Scoped_Identifier** which provides the **identifier** "2625673282" and **version** "1.00" to Attachment1 and through that to Value_Meaning_Set1.

C.1.17 Value_Meaning_Set1

Value_Meaning_Set1 is an instance of **Value_Meaning_Set** (a subclass of **Item**) which provides a grouping for **Value_Meanings** Meaning1 and Meaning2. Value_Meaning_Set1 is associated with Conceptual_Domain1 via the link **conceptual_domain_value_meaning_set**.

C.1.18 Attachment2

Attachment2 is an instance of **Attached_Item** which is a subclass of **Registered_Item** and **Identified_Item**. Attachment2 provides identification through the link **item_identification** to the **Value_Meaning** Meaning1and the **identification** link to Identifier3.

C.1.19 Identifier3

Identifier3 is an instance of **Scoped_Identifier** which provides the **identifier** "2625673283" and **version** "1.00" to Attachment2 and through that to **Value_Meaning** Meaning1.

C.1.20 Meaning1

Meaning1 is an instance of **Value_Meaning** (a subclass of **Item**) which is contained within Value_Meaning_Set1 through the link **value_meaning_set_value_meaning**.

C.1.21 Designation4

Designation4 is an instance of **Designation** which provides the sign (or name) "Human Male" to the **Value_Meaning** Meaning1 through the link **item_designation**.

C.1.22 Definition1

Definition1 is an instance of **Definition** which provides definition to the **Value_Meaning** Meaning1 through the link **item_definition**.

C.1.23 Pairing1

Pairing1 is an instance of **Designation_Definition_Pairing** which pairs Definition1 with Designation4 and Context2.

C.1.24 Attachment3

Attachment3 is an instance of **Attached_Item** which is a subclass of **Registered_Item** and **Identified_Item**. Attachment3 provides identification through the link **item_identification** to the **Value_Meaning** Meaning2and the **identification** link to Identifier4.

C.1.25 Identifier4

Identifier4 is an instance of **Scoped_Identifier** which provides the **identifier** "2625673284" and **version** "1.00" to Attachment3 and through that to **Value_Meaning** Meaning2.

C.1.26 Meaning2

Meaning2 is an instance of **Value_Meaning** (a subclass of **Item**) which is contained within Value_Meaning_Set1 through the link **value_meaning_set_value_meaning**.

C.1.27 Designation5

Designation5 is an instance of **Designation** which provides the sign (or name) "Human Female" to the **Value_Meaning** Meaning2 through the link **item_designation**.

C.1.28 Definition2

Definition2 is an instance of **Definition** which provides definition to the **Value_Meaning** Meaning2 through the link **item_definition**.

C.1.29 Pairing2

Pairing2 is an instance of **Designation_Definition_Pairing** which pairs Definition2 with Designation5 and Context2.

C.1.30 Namespace1

Namespace1 is an instance of **Namespace** which provides the scope for Identifier1, Identifier2, Identifier3 and Identifier4 via the links **identifier_scope**.

C.1.31 Designation6

Designation6 is an instance of **Designation** which provides the sign (or name) “My Registry Namespace” to Namespace1 through the link **item_designation**.

C.1.32 Designation_Context2

Designation_Context2 is an instance of **Designation_Context** through which Context1 provides context to Designation6, with **acceptability** = “preferred”.

Bibliography

[1] ISO 639-2, *Codes for the representation of names of languages — Part 2: Alpha-3 code*

[2] ISO 704, *Terminology work — Principles and methods*

[3] ISO/IEC 646, *Information technology — ISO 7-bit coded character set for information interchange*

[4] ISO 1087:2019, *Terminology work and terminology science — Vocabulary*

[5] ISO/IEC 2382:2015, *Information technology — Vocabulary*

[6] ISO 3166-1, *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes*

[7] ISO 5127:2017, *Information and documentation — Foundation and vocabulary*

[8] ISO/IEC 6523-1:1998, *Information technology — Structure for the identification of organizations and organization parts — Part 1: Identification of organization identification schemes*

[9] ISO 8601, *Data elements and interchange formats — Information interchange — Representation of dates and times*

[10] ISO/IEC 10646, *Information technology — Universal coded character set (UCS)*

[11] ISO/IEC 11179-1:2023, *Information technology — Metadata registries (MDR) — Part 1: Framework*

[12] ISO/IEC 11179-3:2003, *Information technology — Metadata registries (MDR) — Part 3: Registry metamodel and basic attributes (Edition 2)*

[13] ISO/IEC 11179-3:2013, *Information technology — Metadata registries (MDR) — Part 3: Registry metamodel and basic attributes*

[14] ISO/IEC 11179-4, *Information technology — Metadata registries (MDR) — Part 4: Formulation of data definitions*

[15] ISO/IEC 11179-5, *Information technology — Metadata registries (MDR) — Part 5: Naming principles*

[16] ISO/IEC 11179-30, *Information technology — Metadata registries (MDR) — Part 30: Basic attributes of metadata*

[17] ISO/IEC 11179-31:2023, *Information technology — Metadata registries (MDR) — Part 31: Metamodel for data specification registration*

[18] ISO/IEC 11179-32:2023, *Information technology — Metadata registries (MDR) — Part 32: Metamodel for concept system registration*

[19] ISO/IEC 11179-33, *Information technology — Metadata registries (MDR) — Part 33: Metamodel for data set registration (formerly ISO/IEC 11179-7)*

[20] ISO/IEC 11179-34,[1)]*Information technology — Metadata registries (MDR) — Part 34: Metamodel for computable object registration (under development)*

[21] ISO/IEC 11179-35, *Information technology — Metadata registries (MDR) — Part 35: Metamodel for model registration*

[22] ISO/IEC 14977, *Information technology — Syntactic metalanguage — Extended BNF*

1) Under preparation. Stage at the date of publication: ISO/AWI 11179-34:2023.

[23] ISO 15924, *Information and documentation — Codes for the representation of names of scripts*

[24] ISO/IEC 15944-1:2011, *Information technology — Business operational view — Part 1: Operational aspects of open-edi for implementation*

[25] ISO/IEC 19501:2005, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*

[26] ISO/IEC 19505-1:2012, *Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 1: Infrastructure* (See [2)]) (accessed 2022-08-31)

[27] ISO/IEC 19505-2:2012*Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure* (See [3)]) (accessed 2022-08-31)

[28] ISO/IEC/TR 19583-1, *Information technology — Concepts and usage of metadata — Part 1: Metadata concepts*

[29] ISO/IEC 19763 (all parts), *Information technology — Metamodel framework for interoperability (MFI)*

[30] ISO/IEC 19773:2011, *Information technology — Metadata Registries (MDR) modules*

[31] ISO/IEC 24707, *Common Logic Includes a specification of XCL*

[32] DOI® Handbook, < https://www.doi.org/hb.html > (accessed 2022-08-31).

[33] Phillips, A., Ed., and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, < https://www.rfc-editor.org/info/rfc5646 > (accessed 2022-08-31).

[34] Davis M., Phillips A., Umaoka Y., BCP 47 Extension U", RFC 6067, DOI 10.17487/RFC6067, December 2010, < https://www.rfc-editor.org/info/rfc6067 > (accessed 2022-08-31).

[35] ITU-T Recommendation E.164 (2005-02) *The international public telecommunications numbering plan*[4)] (accessed 2022-08-31)

[36] OASIS ebXML Registry Information Model (RIM) version 3.0 [5)] (accessed 2022-08-31) Defines the types of metadata and content that can be stored in an ebXML Registry

[37] RDF – Resource Description Framework [6)] (accessed 2022-08-31)

[38] Universal Postal Union (UPU) S42-1:2003 International postal address components and templates UPU is the Universal Postal Union at "[7)]" (accessed 2022-08-31). UPU S42-1 is based on EN 14142-1, Postal services – Address data bases – Part 1 – Components of Postal Addresses.

[39] METeOR (Australian Institute of Health and Welfare (AIHW) Metadata Online Registry) https://meteor.aihw.gov.au/ (accessed 2022-08-31)

2) http://www.omg.org/spec/UML/ISO/19505-1/PDF

3) http://www.omg.org/spec/UML/ISO/19505-2/PDF

4) https://www.itu.int/rec/T-REC-E.164/en

5) https://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.pdf

6) http://www.w3.org/RDF/

7) http://www.upu.int

ICS 35.040.50

Price based on 97 pages