|  |
| --- |
|  |
| |  |  |  | | --- | --- | --- | | 75 | Date: | October 2014 | |
| Archetype Modeling Language (AML)  Version: 0.5  **OMG Document Number: health/2014-10-01**  **Standard document URL: http://www.omg.org/spec/AML/1.0**  Original File: N/A |

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE.   
  
IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.  
  
The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 140 Kendrick Street, Needham, MA 02494, U.S.A.

TRADEMARKS

MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group©, OMG© , Unified Modeling Language©, Model Driven Architecture Logo©, Model Driven Architecture Diagram©, CORBA logo©, XMI Logo©, CWM©, CWM Logo©, IIOP© , MOF© , OMG Interface Definition Language (IDL)© , and OMG SysML© are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.  
  
Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

**OMG's Issue Reporting Procedure**

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Documents, Report a Bug/Issue (http://www.omg.org/technology/agreement.)

**Acknowledgements**

The following individuals submitted parts of this specification and/or have assisted the AML team in the development of the specification:

Table of Contents

1 5

2 6

2.1 6

2.2 6

2.2.1 6

2.2.2 6

**Preface**

**OMG**

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.  
  
OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG™s specifications implement the Model Driven Architecture (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG™s specifications include: UML® (Unified Modeling Language); CORBA® (Common Object Request Broker Architecture); CWM (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.  
  
More information on the OMG is available at http://www.omg.org/.

**OMG Specifications**

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. A Specifications Catalog is available from the OMG website at:

*http://www.omg.org/technology/documents/spec\_catalog.htm*

Specifications within the Catalog are organized by the following categories:

**OMG Modeling Specifications**

• UML  
• MOF  
• XMI  
• CWM  
• Profile specifications

**OMG Middleware Specifications**

• CORBA/IIOP  
• IDL/Language Mappings  
• Specialized CORBA specifications  
• CORBA Component Model (CCM)

**Platform Specific Model and Interface Specifications**

• CORBAservices  
• CORBAfacilities  
• OMG Domain specifications  
• OMG Embedded Intelligence specifications  
• OMG Security specifications

OMG Headquarters   
 109 Highland Ave,   
 Needham, MA 02494 USA  
 USA   
   
 Tel: +1-781-444-0404   
 Fax: +1-781-444-0320   
 Email: pubs@omg.org  
   
Certain OMG specifications are also available as ISO standards. Please consult http://www.iso.org

**Typographical Conventions**

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

**Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.**

Courier - 10 pt. Bold: Programming language elements.

Helvetica/Arial - 10 pt : Exceptions

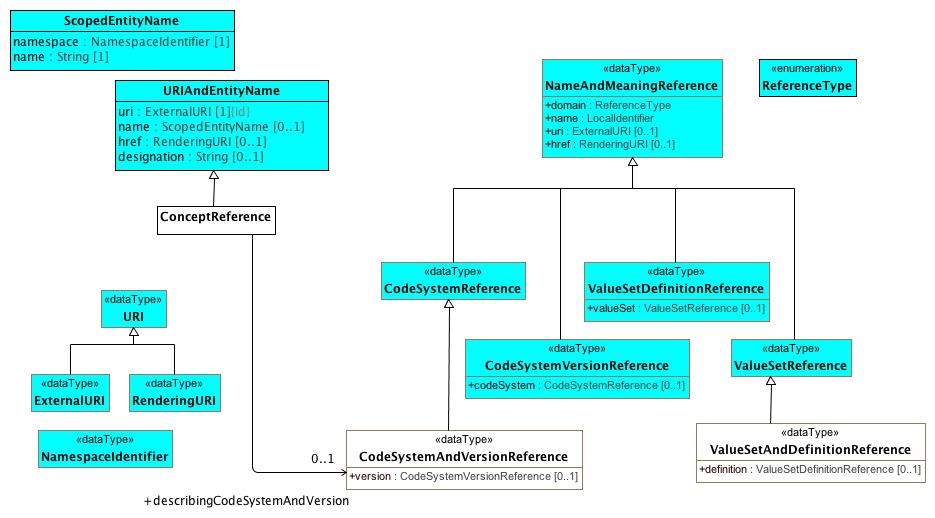
NOTE: Terms that appear in italics are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

# <Package> Terminology Object Model

This section describes how a combination of the applicable elements of the Common Terminology Services (CTS2) and the ISO 11179-3 Metadata registries (MDR) - Part 3: Registry metamodel and basic attributes 3rd Edition can be combined to implement the ADL/AOM "id", "at" and "ac" identifier schemes and the term\_definitions, term\_bindings and value set sections of the Archetype Modeling Language.

We begin by describing the subsetting the core set of elements that are used from the CTS2 Core Model and some minor extensions that are needed for the AML Object Model. We then repeat this process with the ISO 11179-3 elements, integrating them with the CTS2 core elements. We finally identify the AML model components that have identities and corresponding entries in the terminology section.

## <Package> Common Terminology Services Components



**CTSCore Components**

This diagram contains the subset of the [CTS2 Core Model Elements](http://www.omg.org/cgi-bin/doc?formal/2013-12-04) that are used by the AML Object Model. The URI types are derived from Section 2.1.3, ScopedEntityName and URIAndEntityName from Section 2.2.3 Entity References and the NameAndMeaning references from Section

The AML specification adds the following restrictions to these types:

1. The href attribute is not used in URIAndENtityName or NameAndMeaningReference
2. NameAndMeaningReference.domain is determined by the type of the specialization and is omitted.
3. URI is mandatory for NameAndMeaning reference.

The ConceptReference class is specific to AML. The CTS2 model used the term "EntityReference", but the AML and 11179 communities are both used to using the notion of "Concept", although sometimes it is used to reference an identifier, sometimes a description and sometimes an abstract category in the viewer's mind. In AML, we are only interested in the first case, where ConceptReference is an identifier that can be described (or "included" in 11179 parliance) in one or more CodeSystems (Concept\_System in 11179 terms).

The ADL and AOM require that a Terminology\_Code (concept reference) include the identifier of the code system or version of a code system version that was used to determine the intended meaning of the concept at the point in time it was used. This requires one additional data type, "CodeSystemVersionReference", which in the AML context always includes the identifier of a code system but may or may not include a version.

### <DataType> CodeSystemAndVersionReference

**Description**

A reference to a code system and optional version.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff), [Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc), [Data Element and Data Element Concept](#_ea1c0f7548bd076f87be813f2a93f734)

**Direct Known Superclasses (Generalization)**

[CodeSystemReference](#_ef6a61554a10734f76a4adfa457998fd)

**Attributes**

• public version : [CodeSystemVersionReference](#_15d03d8cc99c5a7bf9bc7d48df26c292) [0..1]

### <DataType> CodeSystemReference

**Description**

A URI *reference* to a code system (aka. "concept system, "terminology" or "ontology")

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[NameAndMeaningReference](#_7677b94abdbbdc2c76d49314388386a7)

**Direct Known Subclasses (Specialization)**

[CodeSystemAndVersionReference](#_2a8e174187d7187f0716f48b9f8c271e)

### <DataType> CodeSystemVersionReference

**Description**

A reference to a specific version of a code system.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[NameAndMeaningReference](#_7677b94abdbbdc2c76d49314388386a7)

**Attributes**

• public codeSystem : [CodeSystemReference](#_ef6a61554a10734f76a4adfa457998fd) [0..1]

A reference to the code system that the code system version reference is a version of.

### <DataType> ExternalURI

**Description**

A URI that references an "real world" (vs. digital) entity. Examples include individual people, locations, organizations as well as abstract concepts or classes. "Well behaved" External URI's do not directly reference a digital resource, although they may result in a redirection to a *RenderingURI* that resolves to a description of the actual target.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[URI](#_887928f30f99c8a1ca89ed7a082356aa)

### <DataType> NameAndMeaningReference

**Description**

"NameAndMeaningReference consists of a local identifier that references a unique meaning within the context of a given domain in a CTS2 service instance and a globally unique URI that identifies the intended meaning of the identifier. " (p11 CTS2 1.1 Core Model Elements)

We have tweaked the NameAndMeaningReference model slightly to meet the AML

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Subclasses (Specialization)**

[CodeSystemReference](#_ef6a61554a10734f76a4adfa457998fd), [CodeSystemVersionReference](#_15d03d8cc99c5a7bf9bc7d48df26c292), [ValueSetDefinitionReference](#_bf76d842365737931f787efdb4a36f12), [ValueSetReference](#_5e70a8ee4206e2396a68ca679af92933)

**Attributes**

• public domain : [ReferenceType](#_428e9af279e2df7756c01dffd2ccc1d4)

The "domain" or scope of the reference. This attribute is not used in the AML specification.

• public name : [LocalIdentifier](#_a7bba5b5da9db2f4faa58b64fb2e1eca)

The locally unique identifier string within the context of the domain. As an example, the local name of the SNOMED CT International Edition coding system might be "SNOMEDCT" or "SCT"

• public uri : [ExternalURI](#_de932b9629138c166e8cfb00efa65177) [0..1]

A URI that identifies the specific resource within a global context. As an example, the URI of the SNOMED CT International Code System would be "http://snomed.info/sct/900000000000207008".

• public href : [RenderingURI](#_821273fdc1c3295a17225200782229ea) [0..1]

A URI that, when resolved, results in a description of the referenced resource. The href attribute is not used in the AML Specification.

### <DataType> NamespaceIdentifier

**Description**

An identifier that uniquely references the scoping namespace of an Entity (class, role or individual) within a the context of a service. NamespaceIdentifier syntax must match the [PNAME\_NS](http://www.w3.org/TR/rdf-sparql-query/#rPNAME_NS) production as defined in the [SPARQL Query Specification](http://www.w3.org/TR/rdf-sparql-query/)

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

### <DataType> RenderingURI

**Description**

A URI that represents a digital resource, such as a page in a REST service, an online document or other digital artifact.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[URI](#_887928f30f99c8a1ca89ed7a082356aa)

### <DataType> URI

**Description**

A Universal Resource Identifier (URI) as defined in [IETF 3986](http://www.ietf.org/rfc/rfc3986.txt) . Implementations are encouraged to consider implementing this data type using the IRI ([RFC3987](http://www.ietf.org/rfc/rfc3987.txt)) specification.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Subclasses (Specialization)**

[ExternalURI](#_de932b9629138c166e8cfb00efa65177), [RenderingURI](#_821273fdc1c3295a17225200782229ea)

### <DataType> ValueSetAndDefinitionReference

**Description**

A reference to a value set and optional definition.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff), [Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Direct Known Superclasses (Generalization)**

[ValueSetReference](#_5e70a8ee4206e2396a68ca679af92933)

**Attributes**

• public definition : [ValueSetDefinitionReference](#_bf76d842365737931f787efdb4a36f12) [0..1]

### <DataType> ValueSetDefinitionReference

**Description**

A reference to a set of rules for constructing a value set along with a reference to the corresponding value set if known.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[NameAndMeaningReference](#_7677b94abdbbdc2c76d49314388386a7)

**Attributes**

• public valueSet : [ValueSetReference](#_5e70a8ee4206e2396a68ca679af92933) [0..1]

### <DataType> ValueSetReference

**Description**

A reference to a name set of entity references.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Superclasses (Generalization)**

[NameAndMeaningReference](#_7677b94abdbbdc2c76d49314388386a7)

**Direct Known Subclasses (Specialization)**

[ValueSetAndDefinitionReference](#_25ca98283400275c47128a3c11af1824)

### <Class> ConceptReference

**Description**

A URI that uniquely identifies a "concept" (aka. class, entity, individual or, in some contexts "term"), accompanied by additional information that conveys the intended meaning, code and source of the information used to determine the intent of the URI.

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff), [Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc), [Data Element and Data Element Concept](#_ea1c0f7548bd076f87be813f2a93f734)

**Direct Known Superclasses (Generalization)**

[URIAndEntityName](#_4d1f571ab5e9384786ffe39444e822b4)

**Associations**

• public describingCodeSystemAndVersion : [CodeSystemAndVersionReference](#_2a8e174187d7187f0716f48b9f8c271e)[0..1]

A reference to a code system and optional version.

### <Class> ScopedEntityName

**Description**

The combination of a local identifier for a namespace and a name that is unique within the context of the scoping namespace. Note that ScopedEntityNames cannot be exchanged between implementations as different implementations may chose different scoping namespaces. As an example, one implementation may choose to represent the SNOMED CT namespace as "SCTID" while another may choose "SCT".

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Attributes**

• public namespace : [NamespaceIdentifier](#_94cb7eefb9b55dbc722d53bf1ec0f163) [1]

An identifier that references a unique namespace URI within the context of a service. The type of this identifier is NamespaceIdentifier rather than string to allow conversion to and from XML, RDF, etc.

• public name : String [1]

A unique identifier in the context of the scoping namespace.

### <Class> URIAndEntityName

**Description**

A TerminologyCodeReference </i>(alias: URIAndEntityName) consists of a local identifier that references a unique meaning within the context of a given domain in a terminology service instance and a globally unique <i>URI </i>that identifies the intended meaning of the identifier.</p>

**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

**Direct Known Subclasses (Specialization)**

[ConceptReference](#_4082bf99060eea349dab8f548d297f3d)

**Attributes**

• public uri : [ExternalURI](#_de932b9629138c166e8cfb00efa65177) [1]

The URI that represents this particular referenced entity. Note: We have added the UML {id} attribute to this attribute for clarity.

• public name : [ScopedEntityName](#_bf3eeb4d95f5d93bbd59440cca5ed9d6) [0..1]

A namespace/name combination that uniquely represents the entity. This can be the primary entityID, as determined by the service or any valid alternateId. Service implementers are encouraged to develop mechanisms that will allow clients to choose an appropriate namespace for rendering URIAndEntityName instances. As an example, it should be possible to view SNOMED-CT entity references by either the SctId, the “fully specified name” or, where appropriate, the CTV3ID or SNOMED-3 identifier. Similar mechanisms would apply to ontologies that have both id and label fields.

• public href : [RenderingURI](#_821273fdc1c3295a17225200782229ea) [0..1]

A URI that, when resolved, will return a CTS2 compliant description of the resource referenced by the uri attribute.

• public designation : String [0..1]

A string that represents the target of the reference in a given language and context. The designation attribute is strictly informative and does not participate in or contribute to the identity of the target resource.

**Constraints**

* **identifierOrURI**

A TerminologyCodeReference must either have a *uri*, an *identifier*, or both.

uri->notEmpty() or identifier->notEmpty()

### <Enumeration> ReferenceType

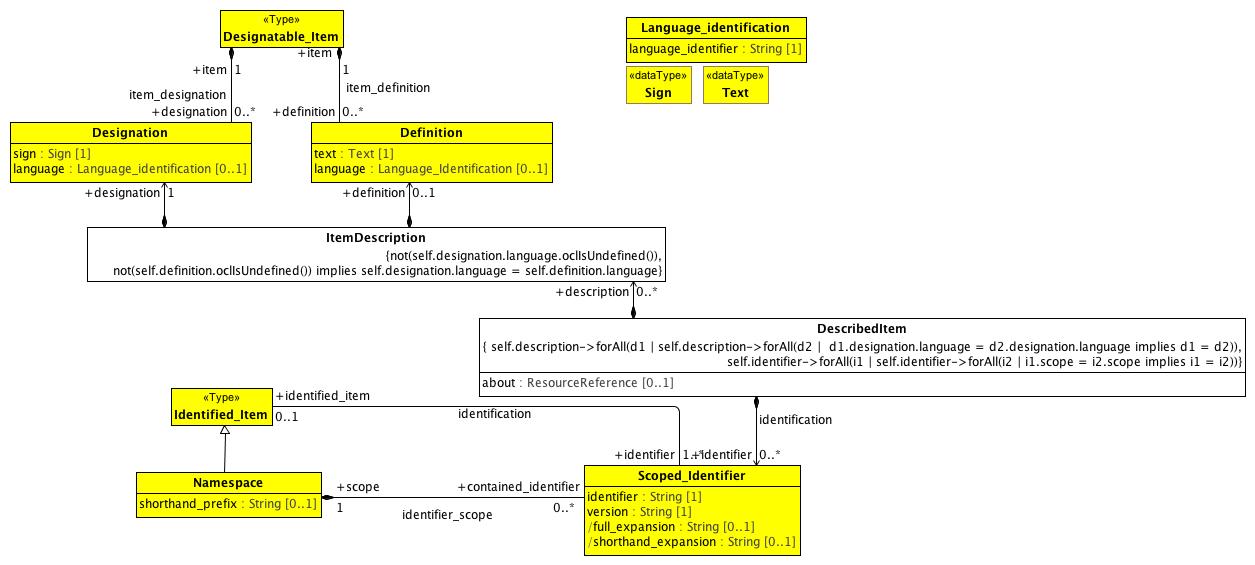
**Diagrams**

[CTSCore Components](#_dcc5ad33945195c14fa1ef7b2e5b33ff)

## <Package> ISO Model Components

This section describes the subset of the *ISO 11179-3 Information technology - Metadata registries (MDR) - Part 3: Registry metamodel and basic attributes 3rd Edition* that are used in the construction of the AML Object Model. Model elements borrowed from ISO 11179 are identified with a yellow background to distinguish them from AML, CTS2 and UML Reference Model components. The first section describes the relationship between the designation, definition and identification sections and AML while the second addresses the meaning/representation links and their AML equivalent.

### <Package> Designation, Definition and Identification



**Designation and Definition metamodel region**

This diagram contains the subset of the ISO 11179-3:2012(E) model elements in sections 7.2.2 Identification metamodel region and 7.3.2 Designation and Definition metamodel region that are used in the AML Object Model, accompanied by supporting elements drawn from several other sections.

The AML object model is able to use the Designation, Definition and Scoped\_Identifier classes directly with the restrictions mentioned below. It is not able to use Designatable\_Item, which supports an unlimited number of definitions and/or designations while the AML specification includes the following restrictions:

1. There must be at most one ItemDescription per language
2. Every description must consist of exactly one Designation and zero or one Definitions
3. The language attribute is required
4. The language attributes of the Definition and Designation pairs must match.

It is also not able to take advantage of the Identified\_Item class because ISO 11179 requires that all Identified\_Items have at least one Scoped\_Identifier. While this is strictly true in the case of UML, the UML NamedEntity name serves this role and does not need to be exposed to the users unless they intend to export or import from an external system such as ADL. Because of this, we need to create a new relationship between DescribedItem and Scoped\_Identifier with a cardinality of 0..\*.

### <DataType> Sign

**Description**

A sign may be a character string, graphic image, sound clip or other symbol that can be used to denote or designate a concept. The Sign datatype may be implemented using the Reflit(of\_type) data structure (see ISO/IEC 19773:2011 10.4.2), where the list of supported types is implementation defined. At a minimum, datatype String must be supported.

**Note:** The AML specification uses the minimum form of sign -- String.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

### <DataType> Text

**Description**

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 [ISO/IEC 2382-23:1994]

Note: AML treats the Text data type as a simple String.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

### <Class> Definition

**Description**

The Definition class records the binding of a definition text and its language to a Designatable\_Item.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Attributes**

• public text : [Text](#_b7253cf7959d2481cd50a89513cd395b) [1]

text of the Definition

• public language : [Language\_Identification](#_fbb634b0a8bc2adc02ea2c96829b9d49) [0..1]

language (3.2.68) used to write the definition text

**Associations**

• public item : [Designatable\_Item](#_b6cf75a1c58d0a2c045bd8f3929ee2bd)[1]

the class of objects which can have designations and definitions.

### <Class> DescribedItem

**Description**

A DescribedItem is an ISO 11179 Designatable\_Item and Identifiable\_item with the following exceptionns

1. Identifiers are optional.
2. ItemDescriptions must come in designation and optional definition pairs
3. The language of a designation and definition pair is required and must match.
4. There can only be one ItemDescription for a given language.
5. There can only be one Scoped\_Identifier for any given namespace.

**Diagrams**

[AML DescribedItems](#_e01fdbb84f109fa2531577050a9e4ad8), [Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Direct Known Subclasses (Specialization)**

[EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855), [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)

**Attributes**

• public about : [ResourceReference](#_3011e434b91fcdb8310b1acf9765e89c) [0..1]

The about attribute supports the data\_element\_meaning, value\_domain\_meaning and permissible\_value\_meaning associations in the ISO 11179 model. See

**Associations**

• public identifier : [Scoped\_Identifier](#_e9986a35a9b1fdfe3b6504bff891a4a7)[0..\*]

Scoped\_Identifier is a class each instance of which models a scoped identifier (3.2.122), an identifier (7.2.2.2.2.1) with a particular scope provided by a Namespace (7.2.2.3).

• public description : [ItemDescription](#_70b51d918df4b49dcc323ca2d82f1258)[0..\*]

**Constraints**

* **sameLanguage**

not(self.definition.oclIsUndefined()) implies self.definition.language = self.designation.language

* **zeroOrOneDefinition**
* **mandatoryLanguage**

not(self.designation.language.oclIsUndefined())

* **oneIdPerNamespace**

self.identifier->forAll(i1 | self.identifier->forAll(i2 | i1.scope = i2.scope implies i1 = i2))

* **oneDescriptionPerLanguage**

self.description->forAll(d1 | self.description->forAll(d2 | d1.designation.language = d2.designation.language implies d1 = d2))

### <Class> Designatable\_Item

**Description**

the class of objects which can have designations and definitions.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Associations**

• public designation : [Designation](#_f46de41197f2753a2a6dc47a2e5c89fd)[0..\*]

Records the binding of a pair comprised of a sign and its language to a Designatable\_Item. Each Designation ... may be paired with a Definition.

• public definition : [Definition](#_de3b93f77a3823bde9b40110899bbf83)[0..\*]

The Definition class records the binding of a definition text and its language to a Designatable\_Item.

### <Class> Designation

**Description**

Records the binding of a pair comprised of a sign and its language to a Designatable\_Item. Each Designation ... may be paired with a Definition.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Attributes**

• public sign : [Sign](#_3e3bff45f2bcef12ecdff2975e0f26aa) [1]

used to designate a Designatable\_Item, e.g., a name of an object or concept. The sign may be a word or phrase in a natural language (as specified by the language), or it may be an icon or other symbol.

• public language : [Language\_identification](#_290b504024c5117158636715bff68ef8) [0..1]

used to record the language or dialect in which the sign (usually a name) is used, when the sign has an associated language. Usually the language will refer to a natural human language.

**Associations**

• public item : [Designatable\_Item](#_b6cf75a1c58d0a2c045bd8f3929ee2bd)[1]

the class of objects which can have designations and definitions.

### <Class> Identified\_Item

**Description**

Identified\_Item is a class each instance of which models an identified item, a metadata item that is identified in a metadata registry.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Direct Known Subclasses (Specialization)**

[Namespace](#_540375a26e1a8a2f020e1b8d908aad0c)

**Associations**

• public identifier : [Scoped\_Identifier](#_e9986a35a9b1fdfe3b6504bff891a4a7)[1..\*]

Scoped\_Identifier is a class each instance of which models a scoped identifier (3.2.122), an identifier (7.2.2.2.2.1) with a particular scope provided by a Namespace (7.2.2.3).

### <Class> ItemDescription

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Associations**

• public definition : [Definition](#_de3b93f77a3823bde9b40110899bbf83)[0..1]

The Definition class records the binding of a definition text and its language to a Designatable\_Item.

• public designation : [Designation](#_f46de41197f2753a2a6dc47a2e5c89fd)[1]

Records the binding of a pair comprised of a sign and its language to a Designatable\_Item. Each Designation ... may be paired with a Definition.

**Constraints**

* **mandatoryLanguage**

All designations must have a language

not(self.designation.language.oclIsUndefined())

* **matchingDescriptionLanguage**

not(self.definition.oclIsUndefined()) implies self.designation.language = self.definition.language

### <Class> Language\_identification

**Description**

Language\_Identification serves as an identifier for a language. Language\_Identification always defines a language as spoken (or written, signed or otherwise signaled) by human beings for communication of information to other human beings. Computer languages such as programming languages are explicitly excluded.

**Note 1:** ISO 11179-3 defines several additional attributes for the Language\_Identification class. As the AML specification simply references languages, details such as script, geopolitical, variant identifiers, etc. are not in scope.

**Note 2**: The ISO 11179-3 references several language identificatin documents, including the W3C IETF, IANA and RFC5646. The AML specification is agnostic when it comes to language tags and will treat language as another terminology / value set.

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Attributes**

• public language\_identifier : String [1]

identifies the primary language.

### <Class> Namespace

**Description**

Namespace is a class each instance of which represents a namespace. Namespace is a scoping construct used to group sets of Designations and/or Scoped\_Identifiers used in a metadata registry. Distinct Namespaces permit independent development of metadata collections and/or ontologies. They permit enforcement of uniqueness constraints on identifiers or designation\_signs within a specific Namespace without central coordination.

**Note:** The namespace Class has several additional properties and relationships in the 11179 specification. For the purposes of the AML specification, we need only to know the namespace, its official identifier and its shorthand prefix. In particular, AML will assume that (a) shorthand\_prefix is always present (multiplicity: [1..1]).

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Direct Known Superclasses (Generalization)**

[Identified\_Item](#_1248f279d4892b4c57cebdb831316891)

**Attributes**

• public shorthand\_prefix : String [0..1]

prefix conventionally used as shorthand for a namespace, for greater

readability, in text for human consumption.

**Associations**

• public contained\_identifier : [Scoped\_Identifier](#_e9986a35a9b1fdfe3b6504bff891a4a7)[0..\*]

Scoped\_Identifier is a class each instance of which models a scoped identifier (3.2.122), an identifier (7.2.2.2.2.1) with a particular scope provided by a Namespace (7.2.2.3).

### <Class> Scoped\_Identifier

**Description**

Scoped\_Identifier is a class each instance of which models a scoped identifier (3.2.122), an identifier (7.2.2.2.2.1) with a particular scope provided by a Namespace (7.2.2.3).

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Attributes**

• public identifier : String [1]

An identifier may be used as an unambiguous identifier for an Identified\_Item within a particular Namespace.

• public version : String [1]

version allows more than one version of an Identified\_Item to be identified within a particular Namespace.

**Note:** the AML specification does not include the notion of identified item "versioning", so the version attribute is not used in the AML specification. For the purposes of compatibility with ISO 11179, one can assume that the version attribute has a single, global default value.

• public full\_expansion : String [0..1]

full\_expansion is formed by prefixing the unique identifier of Namespace to the identifier of this Scoped\_Identifier. For the purposes of the AML specification

• public shorthand\_expansion : String [0..1]

short\_expansion (sic) is formed by prefixing the short\_prefix (7.2.2.3.2.5) of Namespace to the identifier of this Scoped\_Identifier. short\_expansion will exist if and only if the corresponding short\_prefix exists.

**Associations**

• public scope : [Namespace](#_540375a26e1a8a2f020e1b8d908aad0c)[1]

Namespace is a class each instance of which represents a namespace. Namespace is a scoping construct used to group sets of Designations and/or Scoped\_Identifiers used in a metadata registry. Distinct Namespaces permit independent development of metadata collections and/or ontologies. They permit enforcement of uniqueness constraints on identifiers or designation\_signs within a specific Namespace without central coordination.

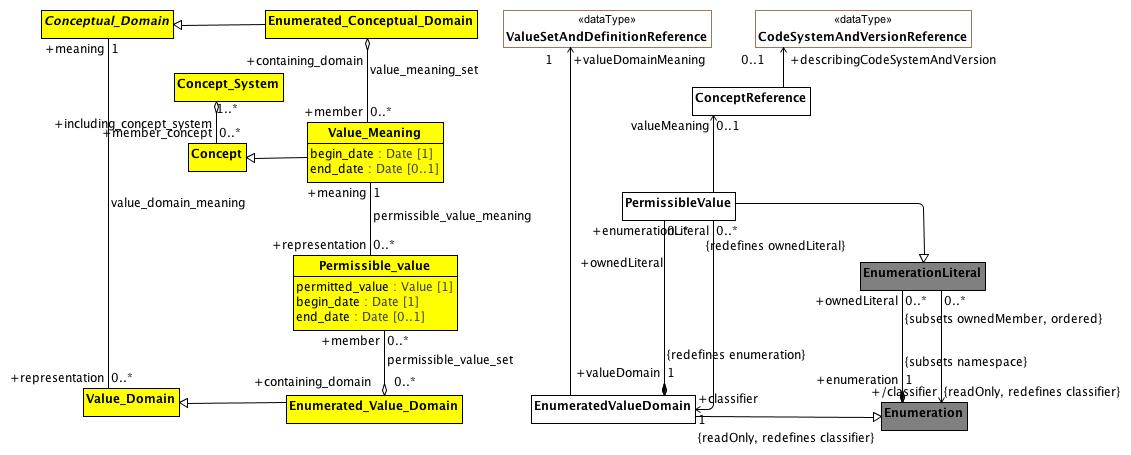
**Note:** The namespace Class has several additional properties and relationships in the 11179 specification. For the purposes of the AML specification, we need only to know the namespace, its official identifier and its shorthand prefix. In particular, AML will assume that (a) shorthand\_prefix is always present (multiplicity: [1..1]).

• public identified\_item : [Identified\_Item](#_1248f279d4892b4c57cebdb831316891)[0..1]

Identified\_Item is a class each instance of which models an identified item, a metadata item that is identified in a metadata registry.

### <Package> Describable Items in AML

#### Conceptual and Value Domains



**Conceptual and value domain metamodel region**

A Conceptual\_Domain sometimes contains a finite allowed inventory of notions that can be enumerated. Such a Conceptual\_Domain is referred to as an Enumerated\_Conceptual\_Domain.

**Known other Data Types**

[CodeSystemAndVersionReference](#_2a8e174187d7187f0716f48b9f8c271e), [ValueSetAndDefinitionReference](#_25ca98283400275c47128a3c11af1824)

### <Class> Concept

**Description**

a unit of knowledge created by a unique combination of characteristics. A concept is independent of representation.

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc), [Data Element and Data Element Concept](#_ea1c0f7548bd076f87be813f2a93f734)

**Direct Known Subclasses (Specialization)**

[Data\_Element\_Concept](#_957a61f61bd5a7299aaa33e4cc6ae00a), [Value\_Meaning](#_2a66de942c64db0df9c0839ab3abc288)

**Associations**

• public including\_concept\_system : [Concept\_System](#_3c62b4de493817e9b573867c82d00f2a)[1..\*]

a set of concepts structured according to the relations among them.

### <Class> Conceptual\_Domain

**Description**

a set of value meanings which may either be enumerated or expressed via a description.

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Direct Known Subclasses (Specialization)**

[Enumerated\_Conceptual\_Domain](#_12c689c6a7d403bb204b68d02b775064)

**Associations**

• public representation : [Value\_Domain](#_9b81960c26ee442be19393f5a5fc3ef2)[0..\*]

### <Class> Concept\_System

**Description**

a set of concepts structured according to the relations among them.

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc), [Data Element and Data Element Concept](#_ea1c0f7548bd076f87be813f2a93f734)

**Associations**

• public member\_concept : [Concept](#_2f48f391605b5ae297a56304194c2a30)[0..\*]

a unit of knowledge created by a unique combination of characteristics. A concept is independent of representation.

### <Class> EnumeratedValueDomain

**Description**

An EnumeratedValueDomain represents a discrete set of possible values for a particular field or data element. Each permissible value represents an intended meaning that, while sometimes determinable from the string itself or its accompanying documentation, can only be fully fixed by connecting it to an official "value meaning" reference in an external terminological resource.

**Diagrams**

[AML DescribedItems](#_e01fdbb84f109fa2531577050a9e4ad8), [Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc), [EnumerationConstraints](#_9a0e8e1c8e8cdbf68da35592986a39eb)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_4cfed31aa26e11dff4000228bcd222ae), [Enumerated\_Value\_Domain](#_f7ddc2c93ef0e7318e8f00f08a71d10c), [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e)

**Direct Known Subclasses (Specialization)**

[ValueSet](#_73d0fab5bddf198ab14a77c3fed1636a)

**Attributes**

• public p1

**Associations**

• public ownedLiteral : [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)[0..\*]

A permissible value within the context of a value domain. While permissible values may be represented as integers, strings or simply as named data type instances (as is the case in UML), all permissible values need to have a mechanism for providing a String representation of the represented value. The String returned by the value function must be unique within the context of the containing domain.

• public valueDomainMeaning : [ValueSetAndDefinitionReference](#_25ca98283400275c47128a3c11af1824)[1]

A reference to a value set and optional definition.

### <Class> Enumerated\_Conceptual\_Domain

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Direct Known Superclasses (Generalization)**

[Conceptual\_Domain](#_7e63d205ca8838d42d16c4aae3fbbf3c)

**Associations**

• public member : [Value\_Meaning](#_2a66de942c64db0df9c0839ab3abc288)[0..\*]

Value\_Meaning is a class each instance of which models a value meaning, which provides *semantic content* of a possible value.

Each member of an Enumerated\_Conceptual\_Domain has a Value\_Meaning that provides its distinction from other members. In the example of ISO 3166, the notion of each country as specified would be the Value\_Meanings. The *representation of Value\_Meanings in a registry shall be independent of (and shall not constrain) their representation in any corresponding Value\_Domain.* A particular Value\_Meaning may have more than one means of representation by Permissible\_Values — each from a distinct Enumerated\_Value\_Domain.

The italics in the above definition are ours. The key aspects of Value\_Meaning are:

1. It represents a concept that is a member ("contained") of one or more Concept\_Systems
2. It provides semantic content of a permissible value.
3. The representation of the Value\_Meaning is independent of how the information is represented in an Enumerated\_Value\_Domain.

### <Class> Enumerated\_Value\_Domain

**Description**

a class each instance of which models an enumerated value domain, a value domain that is specified by a list of all its permissible values.

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Direct Known Superclasses (Generalization)**

[Value\_Domain](#_9b81960c26ee442be19393f5a5fc3ef2)

**Direct Known Subclasses (Specialization)**

[EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51)

**Associations**

• public member : [Permissible\_value](#_2fd4e293390c50189eca5b7ab34d63cd)[0..\*]

a class each instance of which models a permissible value.

**Note:** The AML model views Enumerated\_Value\_Domains as a extension to the UML Enumeration DataType and Permissible\_values as Enumeration\_Literals. The notion of begin\_date becomes an aspect of the model (version) itself rather than the date when the specific member "became valid". As a result, the model will assume that begin\_date is assigned a date that is sufficiently far in the past that we do not have to concern ourselves with it. In addition, permissible values will be removed from the accompanying domain, so end\_date will not be populated.

### <Class> Enumeration

**Description**

A subset of the UML::Enumeration data type. While UML::Enumeration data types can have both ownedAttributes and ownedOperations, these aspects are ignored from the AML perspective. The only aspects of an RMMEnumeration that are visible in the AML model is the package name.

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc), [EnumerationConstraints](#_9a0e8e1c8e8cdbf68da35592986a39eb)

**Direct Known Superclasses (Generalization)**

[RMMDataType](#_d5914eb0da42172989bbe57f23fc4310)

**Direct Known Subclasses (Specialization)**

[EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51)

**Associations**

• public ownedLiteral : [EnumerationLiteral](#_41faf6a7041d7068bbbf4f9ff3924d22)[0..\*]

A RMMEnumerationLiteral is a specialization UML::EnumerationLiteral. The only characteristic that is significant from the AML model perspective is the *RMMEnumerationLiteral* name, which is unique within the context of the RMMEnumeration namespace. An *RMMEnumerationLiteral* returns its *name* as the *PermissibleValue* value().

**Constraints**

* **enumerateValueDomain**

### <Class> EnumerationLiteral

**Description**

A RMMEnumerationLiteral is a specialization UML::EnumerationLiteral. The only characteristic that is significant from the AML model perspective is the *RMMEnumerationLiteral* name, which is unique within the context of the RMMEnumeration namespace. An *RMMEnumerationLiteral* returns its *name* as the *PermissibleValue* value().

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc), [EnumerationConstraints](#_9a0e8e1c8e8cdbf68da35592986a39eb)

**Direct Known Superclasses (Generalization)**

[RMMNamedElement](#_527fd9eb1e787c36a3748854a9431816)

**Direct Known Subclasses (Specialization)**

[PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)

**Associations**

• public enumeration : [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e)[1]

A subset of the UML::Enumeration data type. While UML::Enumeration data types can have both ownedAttributes and ownedOperations, these aspects are ignored from the AML perspective. The only aspects of an RMMEnumeration that are visible in the AML model is the package name.

• public classifier : [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e)

A subset of the UML::Enumeration data type. While UML::Enumeration data types can have both ownedAttributes and ownedOperations, these aspects are ignored from the AML perspective. The only aspects of an RMMEnumeration that are visible in the AML model is the package name.

### <Class> PermissibleValue

**Description**

A permissible value within the context of a value domain. While permissible values may be represented as integers, strings or simply as named data type instances (as is the case in UML), all permissible values need to have a mechanism for providing a String representation of the represented value. The String returned by the value function must be unique within the context of the containing domain.

**Diagrams**

[AML DescribedItems](#_e01fdbb84f109fa2531577050a9e4ad8), [Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc), [EnumerationConstraints](#_9a0e8e1c8e8cdbf68da35592986a39eb)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_4cfed31aa26e11dff4000228bcd222ae), [EnumerationLiteral](#_41faf6a7041d7068bbbf4f9ff3924d22)

**Associations**

• public valueDomain : [EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51)[1]

An EnumeratedValueDomain represents a discrete set of possible values for a particular field or data element. Each permissible value represents an intended meaning that, while sometimes determinable from the string itself or its accompanying documentation, can only be fully fixed by connecting it to an official "value meaning" reference in an external terminological resource.

• public classifier : [EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51)[1]

An EnumeratedValueDomain represents a discrete set of possible values for a particular field or data element. Each permissible value represents an intended meaning that, while sometimes determinable from the string itself or its accompanying documentation, can only be fully fixed by connecting it to an official "value meaning" reference in an external terminological resource.

• valueMeaning : [ConceptReference](#_4082bf99060eea349dab8f548d297f3d)[0..1]

A URI that uniquely identifies a "concept" (aka. class, entity, individual or, in some contexts "term"), accompanied by additional information that conveys the intended meaning, code and source of the information used to determine the intent of the URI.

### <Class> Permissible\_value

**Description**

a class each instance of which models a permissible value.

**Note:** The AML model views Enumerated\_Value\_Domains as a extension to the UML Enumeration DataType and Permissible\_values as Enumeration\_Literals. The notion of begin\_date becomes an aspect of the model (version) itself rather than the date when the specific member "became valid". As a result, the model will assume that begin\_date is assigned a date that is sufficiently far in the past that we do not have to concern ourselves with it. In addition, permissible values will be removed from the accompanying domain, so end\_date will not be populated.

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Attributes**

• public permitted\_value : [Value](#_52e70dc394d1abd3621f32ef313b1424) [1]

the actual value of the Permissible\_Value

• public begin\_date : [Date](#_b5130c0b7557264e040caf21b3b5ef6d) [1]

date at which the Permissible\_Value became valid

• public end\_date : [Date](#_b5130c0b7557264e040caf21b3b5ef6d) [0..1]

date at which the Permissible\_Value ceased to be valid

**Associations**

• public meaning : [Value\_Meaning](#_2a66de942c64db0df9c0839ab3abc288)[1]

Value\_Meaning is a class each instance of which models a value meaning, which provides *semantic content* of a possible value.

Each member of an Enumerated\_Conceptual\_Domain has a Value\_Meaning that provides its distinction from other members. In the example of ISO 3166, the notion of each country as specified would be the Value\_Meanings. The *representation of Value\_Meanings in a registry shall be independent of (and shall not constrain) their representation in any corresponding Value\_Domain.* A particular Value\_Meaning may have more than one means of representation by Permissible\_Values — each from a distinct Enumerated\_Value\_Domain.

The italics in the above definition are ours. The key aspects of Value\_Meaning are:

1. It represents a concept that is a member ("contained") of one or more Concept\_Systems
2. It provides semantic content of a permissible value.
3. The representation of the Value\_Meaning is independent of how the information is represented in an Enumerated\_Value\_Domain.

• public containing\_domain : [Enumerated\_Value\_Domain](#_f7ddc2c93ef0e7318e8f00f08a71d10c)[0..\*]

a class each instance of which models an enumerated value domain, a value domain that is specified by a list of all its permissible values.

### <Class> Value\_Domain

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Direct Known Subclasses (Specialization)**

[Enumerated\_Value\_Domain](#_f7ddc2c93ef0e7318e8f00f08a71d10c)

**Associations**

• public meaning : [Conceptual\_Domain](#_7e63d205ca8838d42d16c4aae3fbbf3c)[1]

a set of value meanings which may either be enumerated or expressed via a description.

• public usage : [Data\_Element](#_742378c87010c3a79ca426ed64113f23)[0..\*]

Data\_Element is a class each instance of which models a data element, a unit of data that is *considered in context to be indivisible*. A data element is a basic unit of data of interest to an organization, for which the definition, identification, representation, and permissible values are specified by means of a set of attributes. Examples of data element include: a column in a table of a relational database, a field in a record or form, an XML element, the attribute of a Java class, or a variable in a program. The description of data elements is a major purpose of ISO/IEC 11179 Metadata Registries.

(Italics added)

### <Class> Value\_Meaning

**Description**

Value\_Meaning is a class each instance of which models a value meaning, which provides *semantic content* of a possible value.

Each member of an Enumerated\_Conceptual\_Domain has a Value\_Meaning that provides its distinction from other members. In the example of ISO 3166, the notion of each country as specified would be the Value\_Meanings. The *representation of Value\_Meanings in a registry shall be independent of (and shall not constrain) their representation in any corresponding Value\_Domain.* A particular Value\_Meaning may have more than one means of representation by Permissible\_Values — each from a distinct Enumerated\_Value\_Domain.

The italics in the above definition are ours. The key aspects of Value\_Meaning are:

1. It represents a concept that is a member ("contained") of one or more Concept\_Systems
2. It provides semantic content of a permissible value.
3. The representation of the Value\_Meaning is independent of how the information is represented in an Enumerated\_Value\_Domain.

**Diagrams**

[Conceptual and value domain metamodel region](#_c940c3b06c77c12f4e6ddf81cb3280bc)

**Direct Known Superclasses (Generalization)**

[Concept](#_2f48f391605b5ae297a56304194c2a30)

**Attributes**

• public begin\_date : [Date](#_b5130c0b7557264e040caf21b3b5ef6d) [1]

date at which this Value\_Meaning became, or will become, a valid Value\_Meaning.

**Note:** from the AML perspective, the Value\_Meaning membership in an Enumerated\_Conceptual\_Domain is determined by a ValueSetDefinition, which is a set of rules for determining the members of a ValueSet (Enumerated\_Conceptual\_Domain) at a given point in time. As such, the begin\_date attribute will be ignored, although it could be derived from one of the date/time attributes of the CTS2 ValueSetDefinition.

• public end\_date : [Date](#_b5130c0b7557264e040caf21b3b5ef6d) [0..1]

date on which the Value\_Meaning ceased, or will cease, to be valid.

**Note:** as with begin\_date, end\_date is not used in the AML specification.

**Associations**

• public containing\_domain : [Enumerated\_Conceptual\_Domain](#_12c689c6a7d403bb204b68d02b775064)

• public representation : [Permissible\_value](#_2fd4e293390c50189eca5b7ab34d63cd)[0..\*]

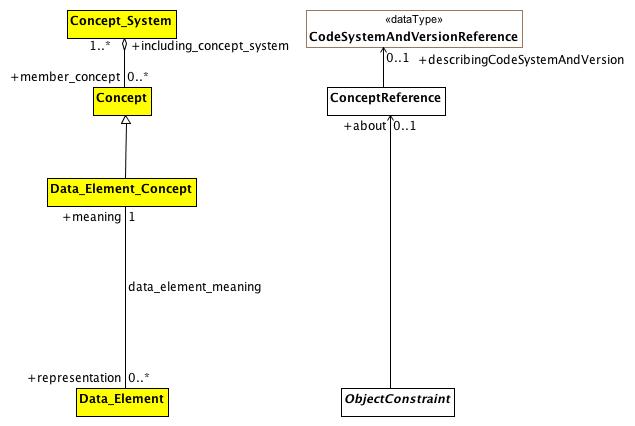
a class each instance of which models a permissible value.

**Note:** The AML model views Enumerated\_Value\_Domains as a extension to the UML Enumeration DataType and Permissible\_values as Enumeration\_Literals. The notion of begin\_date becomes an aspect of the model (version) itself rather than the date when the specific member "became valid". As a result, the model will assume that begin\_date is assigned a date that is sufficiently far in the past that we do not have to concern ourselves with it. In addition, permissible values will be removed from the accompanying domain, so end\_date will not be populated.

**Known other classes**

[ConceptReference](#_4082bf99060eea349dab8f548d297f3d)

#### Data Element and Data Element Concept



**Data Element and Data Element Concept**

This diagram shows the AML equivalent for the ISO 11179 Data\_Element and Data\_Element\_Concept element. The first thing to note is that, according to ISO 11179, a Data\_Element is "a unit of data that is *considered in context to be indivisible*". While this covers certain ObjectConstraints, the AML requirements need to assign meaning to both indivisible *and* divisible information artifacts, meaning that the data\_element\_meaning relationship is a subset of ObjectConstraint.about.

The second thing to note is that, from the perspective of ISO 11179, every Data\_Element must be associated with exactly one meaning. From the AML perspective, while every ObjectConstraint possibly *should* be associated with a corresponding (Data Element) concept that provides meaning, it will frequently be the case that such an association will not be available.

**Known other Data Types**

[CodeSystemAndVersionReference](#_2a8e174187d7187f0716f48b9f8c271e)

### <Class> Data\_Element

**Description**

Data\_Element is a class each instance of which models a data element, a unit of data that is *considered in context to be indivisible*. A data element is a basic unit of data of interest to an organization, for which the definition, identification, representation, and permissible values are specified by means of a set of attributes. Examples of data element include: a column in a table of a relational database, a field in a record or form, an XML element, the attribute of a Java class, or a variable in a program. The description of data elements is a major purpose of ISO/IEC 11179 Metadata Registries.

(Italics added)

**Diagrams**

[Data Element and Data Element Concept](#_ea1c0f7548bd076f87be813f2a93f734)

**Associations**

• public meaning : [Data\_Element\_Concept](#_957a61f61bd5a7299aaa33e4cc6ae00a)[1]

a concept that is an association of a property with an object class. A data element concept is a concept that can be represented in the form of a data element described independently of any particular representation.

• public domain : [Value\_Domain](#_9b81960c26ee442be19393f5a5fc3ef2)[1]

### <Class> Data\_Element\_Concept

**Description**

a concept that is an association of a property with an object class. A data element concept is a concept that can be represented in the form of a data element described independently of any particular representation.

**Diagrams**

[Data Element and Data Element Concept](#_ea1c0f7548bd076f87be813f2a93f734)

**Direct Known Superclasses (Generalization)**

[Concept](#_2f48f391605b5ae297a56304194c2a30)

**Associations**

• public representation : [Data\_Element](#_742378c87010c3a79ca426ed64113f23)[0..\*]

Data\_Element is a class each instance of which models a data element, a unit of data that is *considered in context to be indivisible*. A data element is a basic unit of data of interest to an organization, for which the definition, identification, representation, and permissible values are specified by means of a set of attributes. Examples of data element include: a column in a table of a relational database, a field in a record or form, an XML element, the attribute of a Java class, or a variable in a program. The description of data elements is a major purpose of ISO/IEC 11179 Metadata Registries.

(Italics added)

### <Class> ObjectConstraint

**Description**

ObjectConstraint represents the properties and associations that are common to all types of object constraints:

Every object constraint may directly specialize at most one parent ObjectConstraint

Every AttributeConstraint is owned by exactly one ObjectConstraint

Every ObjectConstraint is referenced by exactly one AttributeConstraint, with the exception of the root Archetype definition ComplexObjectConstraint that is not owned by any referencing attribute.

The subtypes of ObjectConstraint include:

ObjectConstraintProxy - a reference to an existing NamedObjectConstraint. ObjectConstraintProxys only exist in SourceArchetypes and are replaced by a copy of their targetObject during the flattening process.

NamedObjectConstraint - the set of ObjectConstraints that reference a Reference Model Class and have node identifiers

EnumerationConstraint - constraints on the Reference Model Enumeration class

ArchetypeSlot - identifies a (constrained) slot to be filled by a separate archetype

ArchetypeRootProxy - references an archetype that constraints the type and/or attributes of a Reference Model Class and optionally fills an ArchetypeSlot defined in a parent Archetype

ComplexObjectConstraint - a constraint on the type and/or attributes of a Reference Model Class

TerminologyConstraint - constraints on the TerminologyCodeReference type

PrimitiveObjectConstraint - constraints on the set of primitive data types supplied in the reference model

**Diagrams**

[AML DescribedItems](#_e01fdbb84f109fa2531577050a9e4ad8), [Data Element and Data Element Concept](#_ea1c0f7548bd076f87be813f2a93f734), [EnumerationConstraints](#_9a0e8e1c8e8cdbf68da35592986a39eb)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_4cfed31aa26e11dff4000228bcd222ae)

**Direct Known Subclasses (Specialization)**

[EnumerationConstraint](#_42c2e4f902eddd2a1629a431a96cd94f), [NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9), [ObjectConstraintProxy](#_6da4a9bc7db41a2b89064f79f0c4ed36), [PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da), [TerminologyConstraint](#_b2d4edbc24f651e5a3d756933fff1326)

**Associations**

• public parent : [Data\_Element](#_742378c87010c3a79ca426ed64113f23)[0..\*]

Data\_Element is a class each instance of which models a data element, a unit of data that is *considered in context to be indivisible*. A data element is a basic unit of data of interest to an organization, for which the definition, identification, representation, and permissible values are specified by means of a set of attributes. Examples of data element include: a column in a table of a relational database, a field in a record or form, an XML element, the attribute of a Java class, or a variable in a program. The description of data elements is a major purpose of ISO/IEC 11179 Metadata Registries.

(Italics added)

• public about : [ConceptReference](#_4082bf99060eea349dab8f548d297f3d)[0..1]

A URI that uniquely identifies a "concept" (aka. class, entity, individual or, in some contexts "term"), accompanied by additional information that conveys the intended meaning, code and source of the information used to determine the intent of the URI.

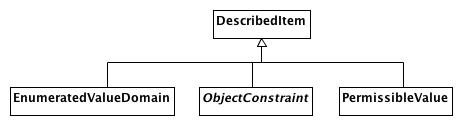
**Known other classes**

[Concept](#_2f48f391605b5ae297a56304194c2a30), [ConceptReference](#_4082bf99060eea349dab8f548d297f3d), [Concept\_System](#_3c62b4de493817e9b573867c82d00f2a)

## <Package> AML Described Items

This section describes how the term\_definitions subsection of the ARCHETYPE\_TERMINOLOGY class is represented in the AML Object Model. The term\_definitions attribute is defined as "Hash <Hash <ARCHETYPE\_TERM, String>, String> [0..1]", where the outer hash key is a language code, e.g. "en", "de" while the inner hash codes are term codes, e.g. "id17", "at4".

ARCHETYPE\_TERM, in turn, is defined as a combination of a *code* with accompanying *text*, *description* and *other\_items*, an arbitrary collection of tag/value pairs.



**AML DescribedItems**

This document shows the AML artifacts that can have entries in the ADL terminology\_definitions section. The ADL equivalent of an ObjectConstraint is identified by an "id" code, an EnumeratedValueDomain as an "ac" code and a PermissibleValue as an "at" code.

Note that the ADL language imposes an additional constraint that there must be an entry for the archetype originalLanguage as well as an entry for each translation language and that there cannot be an entry for any additional languages. The AML specification leaves enforcement of this type of constraint to language specific validation and export tools.

Also note that the ADL specification requires that every entry have an Scoped\_Identifier. Again, this requirement is not necessary if one is modeling strictly with UML and will need to be enforced in a language specific validation/export tool.

**Known other classes**

[DescribedItem](#_4cfed31aa26e11dff4000228bcd222ae), [EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855), [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)