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

Copyright © 2014, Mayo Clinic  
Copyright © 2014, Object Management Group, Inc.  
Copyright © 2014, Visumpoint, LLC

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 logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , IMM™ , 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>.)

Table of Contents

1. Scope 9

1.1. Archetype Modeling Language (AML) Background 9

1.2. AML Intended Users 10

1.3. AML Profiles 10

2. Conformance 11

2.1. Conformance Points 11

2.2. AML Reference Model Profile 11

2.3. AML Terminology Binding Profile 11

2.4. AML Constraint Model Profile 11

3. Normative References 11

4. Terms and Definitions 12

5. Symbols 16

5.1. Graphical Symbols 16

5.2. Abbreviations 16

6. Additional Information 17

6.1. Changes to Adopted OMG Specifications 17

6.2. Acknowledgements 17

7. AML Meta Model 17

7.1. <Package> Archetype Meta Model 18

7.1.1. <Package> Archetype Libraries 18

7.1.1.1. <DataType> ArchetypeId 18

7.1.1.2. <DataType> ArchetypeVersionId 19

7.1.1.3. <Class> Archetype 19

7.1.1.4. <Class> ArchetypeLibrary 21

7.1.1.5. <Class> ArchetypeVersion 22

7.1.1.6. <Class> AuthoredResource 23

7.1.1.7. <Class> Class 24

7.1.1.8. <Class> ComplexObjectConstraint 25

7.1.1.9. <Class> FlatArchetype 26

7.1.1.10. <Class> Package 27

7.1.1.11. <Class> RuleStatement 28

7.1.1.12. <Class> SourceArchetype 28

7.1.1.13. <Enumeration> ArchetypeType 29

7.1.2. <Package> Archetypes and the UML Reference Model 30

7.1.2.1. <Class> Classifier 30

7.1.2.2. <Class> NamedElement 31

7.1.2.3. <Class> Namespace 32

7.1.3. <Package> ADL Archetype Metadata 32

7.1.3.1. <Class> ADLResource 33

7.1.3.2. <Class> Description 34

7.1.3.3. <Class> ResourceDescription 34

7.1.3.4. <Class> ResourceDescriptionItem 35

7.1.3.5. <Class> ResourceTranslation 36

7.1.3.6. <Class> TagValue 37

7.1.3.7. <Class> TranslationDetails 37

7.1.3.8. <Enumeration> LifecycleState 38

7.2. <Package> Reference Model Meta Model 39

7.2.1. <Package> Primitive Data Types 39

7.2.1.1. <Class> DateInterval 40

7.2.1.2. <Class> DateTimeInterval 40

7.2.1.3. <Class> DurationInterval 40

7.2.1.4. <Class> IntegerInterval 40

7.2.1.5. <Class> Interval 40

7.2.1.6. <Class> RealInterval 41

7.2.1.7. <Class> TimeInterval 41

7.2.1.8. <DataType> AnyPrimitiveType 43

7.2.1.9. <DataType> BooleanPrimitive 43

7.2.1.10. <DataType> DateMatchPattern 43

7.2.1.11. <DataType> DatePrimitive 43

7.2.1.12. <DataType> DateTimeMatchPattern 44

7.2.1.13. <DataType> DateTimePrimitive 44

7.2.1.14. <DataType> DurationMatchPattern 44

7.2.1.15. <DataType> DurationPrimitive 45

7.2.1.16. <DataType> IntegerPrimitive 45

7.2.1.17. <DataType> RealPrimitive 45

7.2.1.18. <DataType> RegularExpression 45

7.2.1.19. <DataType> StringPrimitive 46

7.2.1.20. <DataType> TimeMatchPattern 46

7.2.1.21. <DataType> TimePrimitive 46

7.2.2. <Package> Reference Metamodel 47

7.2.2.1. <Class> CollectionProperty 47

7.2.2.2. <Class> DataType 48

7.2.2.3. <Class> Enumeration 48

7.2.2.4. <Class> Property 49

7.2.2.5. <Class> SingularProperty 50

7.2.3. <Package> Attribute Constraint References 50

7.2.3.1. <Class> AttributeCollectionConstraint 51

7.2.3.2. <Class> AttributeConstraint 52

7.2.3.3. <Class> SingularAttributeConstraint 54

7.2.3.4. <Enumeration> AttributeExistence 55

7.2.3.5. <Enumeration> CollectionType 55

7.2.4. <Package> Object Constraint References 56

7.2.4.1. <Class> EnumerationConstraint 57

7.2.4.2. <Class> NamedObjectConstraint 58

7.2.4.3. <Class> PrimitiveObjectConstraint 58

7.2.5. <Package> Template Metamodel 60

7.2.5.1. <Class> ClassifierTemplateParameter 60

7.2.5.2. <Class> RedefinableTemplateSignature 61

7.2.5.3. <Class> TemplateBinding 61

7.2.5.4. <Class> TemplateParameterSubstitution 61

7.2.6. <Package> Instance Metamodel 62

7.2.6.1. <Class> InstanceSpecification 63

7.2.6.2. <Class> PrimitiveType 64

7.2.6.3. <Class> RMClass 64

7.2.6.4. <Class> RMClassInstance 64

7.2.6.5. <Class> RMDataType 65

7.2.6.6. <Class> RMDataTypeInstance 65

7.2.6.7. <Class> RMPrimitiveType 65

7.2.6.8. <Class> RMPrimitiveTypeInstance 66

7.2.7. <Package> Package Metamodel 66

7.2.8. <Package> Enumeration Metamodel 67

7.2.8.1. <Class> EnumeratedValueDomain 68

7.2.8.2. <Class> Enumerated\_Value\_Domain 69

7.2.8.3. <Class> EnumerationLiteral 69

7.2.8.4. <Class> PermissibleValue 70

7.3. <Package> Constraint Meta Model 71

7.3.1. <Package> Atomic Data Type Constraints 71

7.3.1.1. <Class> BooleanConstraint 71

7.3.1.2. <Class> DateConstraint 72

7.3.1.3. <Class> DateTimeConstraint 72

7.3.1.4. <Class> DurationConstraint 73

7.3.1.5. <Class> IntegerConstraint 74

7.3.1.6. <Class> RealConstraint 74

7.3.1.7. <Class> StringConstraint 75

7.3.1.8. <Class> TimeConstraint 75

7.3.2. <Package> Terminology Constraints 76

7.3.2.1. <Class> ObjectConstraint 77

7.3.2.2. <Class> TerminologyCode 78

7.3.2.3. <Class> TerminologyCodeConstraint 79

7.3.3. <Package> Enumeration Constraints 80

7.3.3.1. <DataType> ValueSetAndDefinitionReference 80

7.3.3.2. <Class> ConceptReference 81

7.3.3.3. <Class> LocalEnumerationConstraint 81

7.3.4. <Package> Object Constraints 82

7.3.4.1. <Class> ArchetypeRootProxy 83

7.3.4.2. <Class> ArchetypeSlot 84

7.3.4.3. <Class> ObjectConstraintProxy 85

7.3.4.4. <Package> Named Object Constraints 85

7.3.5. <Package> Attribute Constraints 88

7.3.5.1. <Class> AttributeCollectionMember 88

7.3.5.2. <Class> MultiplicityInterval 89

7.4. <Package> Terminology Object Model 89

7.4.1. <Package> Common Terminology Services Components 90

7.4.1.1. <DataType> CodeSystemAndVersionReference 91

7.4.1.2. <DataType> CodeSystemReference 91

7.4.1.3. <DataType> CodeSystemVersionReference 91

7.4.1.4. <DataType> ExternalURI 92

7.4.1.5. <DataType> NameAndMeaningReference 92

7.4.1.6. <DataType> NamespaceIdentifier 93

7.4.1.7. <DataType> RenderingURI 93

7.4.1.8. <DataType> URI 93

7.4.1.9. <DataType> ValueSetDefinitionReference 94

7.4.1.10. <DataType> ValueSetReference 94

7.4.1.11. <Class> ScopedEntityName 94

7.4.1.12. <Class> URIAndEntityName 95

7.4.1.13. <Enumeration> ReferenceType 96

7.4.2. <Package> ISO Model Components 96

7.4.2.1. <Package> Designation, Definition and Identification 96

7.4.2.2. <Package> Describable Items in AML 104

7.4.3. <Package> AML Described Items 110

8. AML Profiles 111

8.1. <Package> Reference Model Profile 114

8.1.1. <Package> Assumed Data Types 114

8.1.1.1. <Class> Date 115

8.1.1.2. <Class> DateTime 115

8.1.1.3. <Class> Duration 115

8.1.1.4. <Class> TerminologyCode 116

8.1.1.5. <Class> Time 117

8.1.1.6. <Stereotype> AMLDataType 117

8.1.1.7. <Primitive Type> AMLBoolean 118

8.1.1.8. <Primitive Type> AMLInteger 118

8.1.1.9. <Primitive Type> AMLReal 118

8.1.1.10. <Primitive Type> AMLString 119

8.1.2. <Package> Reference Model Decorators 119

8.1.2.1. <Stereotype> DataBinding 119

8.1.2.2. <Stereotype> Infrastructure 120

8.1.2.3. <Stereotype> MappedDataType 121

8.1.2.4. <Stereotype> ReferenceModel 121

8.1.2.5. <Stereotype> Runtime 122

8.1.2.6. <Package> DataBinding Example 122

8.2. <Package> Terminology Profile 128

8.2.1. <Package> Identification and Designation 128

8.2.1.1. <Stereotype> ArchetypeType 129

8.2.1.2. <Stereotype> DescribedItem 129

8.2.1.3. <Stereotype> DesignatableItem 130

8.2.1.4. <Stereotype> IdentifiedItem 131

8.2.1.5. <Stereotype> KnownNamespace 132

8.2.1.6. <Stereotype> Language 132

8.2.1.7. <Stereotype> NamespaceInstance 133

8.2.1.8. <Stereotype> ScopedIdentifier 133

8.2.2. <Package> Resource References 134

8.2.2.1. <Package> Resource References 134

8.2.2.2. <Package> Resource Reference Relationships 138

8.2.3. <Package> Enumerated Value Domains 142

8.2.3.1. <Stereotype> EnumeratedValueDomain 143

8.2.3.2. <Stereotype> PermissibleValue 143

8.3. <Package> Constraint Profile 144

8.3.1. <Package> Archetypes 144

8.3.1.1. <Package> Archetype Libraries 144

8.3.1.2. <Package> Archetype Metadata 151

8.3.2. <Package> Data Type Constraints 154

8.3.2.1. <Package> Primitive Type Constraints 154

8.3.2.2. <Package> Date and Time Match Types 163

8.3.2.3. <Package> Intervals 163

8.3.3. <Package> Object and Property Constraints 167

8.3.3.1. <Package> Object Constraints 167

8.3.3.2. <Package> Attribute Constraints 172

8.3.3.3. <Package> Enumeration Constraints 174

8.3.3.4. <Package> Constraint Proxies 176

8.3.4. <Package> Terminology Constraints 177

**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.

# Scope

## Archetype Modeling Language (AML) Background

This specification defines the Archetype Modeling Language (AML). The AML defines a standard means for modeling Archetype Models (AMs) to support the representation of Clinical Information Modeling Initiative (CIMI) artifacts using modeling profiles as defined in the UML. Archetype Models are Platform Independent Models (PIMs) and are developed as a set of constraints on a specific Reference Model (RM).

The CIMI RM is the underlying RM on which CIMI’s clinical information models are defined. The reference model defines a rigorous and stable set of modeling patterns that include a set of structural patterns, complex data types, and demographic classes. All CIMI clinical models will be defined by constraining the CIMI reference model. Each instance of a CIMI Clinical Model will be a constrained instance of the CIMI reference model conforming to the constraints defined by the associated clinical model.

The motivation for including a reference model in the CIMI clinical modeling architecture is to provide a consistent computational framework upon which model authoring and translation tools can be based. The reference model is the ‘common language’ used to describe all clinical models. It provides a single information model that can be used to represent instances of all clinical models and upon which further constraints can be applied to represent the specific information requirements of all clinical model. This information model represents the core artifact implemented in software; it provides the physical structure of the clinical models and its example instances. Existing implementation experience has shown this increases the computational capabilities of the resulting modeling and translation tools.

Development of the AML specification was guided by:

1. The need for a means to accurately and usefully represent AMs in accordance with the openEHR Foundation’s Archetype Definition Language (ADL) and Archetype Object Model (AOM) version 2.0 specifications;
2. Compatibility with the Object Management Group (OMG) *Common Terminology Service 2 (CTS2)* specification; and
3. Where possible, being informed by and faithful to the *ISO/IEC 11179, Information Technology, -- Metadata registries*, specification.

In the AML RFP, the version of the openEHR Foundation’s ADL and AOM specifications cited for coverage by the OMG AML specification was version 1.5. In the process of producing the AML specification, however, a number of inconsistencies were discovered in the openEHR specifications, as well as opportunities for improvements. These were reported to the openEHR Foundation. In response, the openEHR Foundation revised the specifications. This resulted in a set of changes to the specifications that were not backward compatible with version 1.5. As a consequence, the revised specifications were released as version 2.0, subsuming the requirements found in version 1.5, now made consistent in version 2.0, and forming the updated requirements basis for AML coverage.

## AML Intended Users

The AML is primarily intended to support two clinical modeling communities of users:

* Those having subject matter expertise regarding clinical model domains and currently using ADL-based tools to develop such models, and
* Those familiar with modeling using the UML, though not necessarily familiar with clinical modeling domains or current methods employed to represent them.

Clause 7 of this specification, *AML Meta Model*, provides an informational meta model of the openEHR AOM as an aid to bridging between these communities.

While the AML specification targets CIMI clinical modeling practitioners, the modeling approach defined in the profiles is intended to be generalizable for use with other reference models and application in other domain areas.

## AML Profiles

The AML is specified by three UML profilescollectively meeting the requirements of archetype modeling. These are the:

* *Reference Model Profile (RMP)*: Enables the specification of reference models upon which archetypes can be based;
* *Constraint Model Profile (CMP)*: Supports the specification of constraints on a given reference model to enable the development of archetypes including Clinical Information Models (CIMs); and
* *Terminology Binding Profile (TBP)*: Supports the binding of information models to terminology. Terminology bindings include:
  1. *Value Bindings*: Support linking the data model to value domains that restrict the valid value of an attribute to a set of values corresponding to a set of meanings recorded in an external terminology;
  2. *Semantic Bindings:* Define the meaning of model elements using concepts in an external terminology; and
  3. *Constraint Bindings:* Specify constraints on the information model using concepts and relationships defined in an external terminology.

This set of UML profiles enables the specification of CIMI clinical model content (using the CIMI Reference Model) and the generation of CIMI clinical model artifacts, such as ones represented by the openEHR Foundation’s ADL. (The ADL is a serialization of the openEHR Foundation’s AOM.) While the transformation of AML models to an instance of the AOM was an optional requirement for the AML specification, the AML profile supports the representation of sufficient information in an AM to enable such a transformation.

# Conformance

## Conformance Points

This specification defines the following conformance points (also referred to as conformance targets):

* AML Reference Model Profile
* AML Terminology Binding Profile
* AML Constraint Model Profile

## AML Reference Model Profile

Sub clause 8.1 of this specification defines the AML Reference Model Profile.

## AML Terminology Binding Profile

Sub clause 8.2 of this specification defines the AML Terminology Binding Profile. The Terminology Binding Profile imports the Reference Model Profile.

## AML Constraint Model Profile

Sub clause 8.3 of this specification defines the AML Constraint Model Profile. The Constraint Model Profile imports both the Reference Model Profile and Terminology Binding Profile.

# Normative References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

[ADL] openEHR *Archetype Definition Language: ADL2*, <http://www.openehr.org/releases/trunk/architecture/am/adl2.pdf>

[AOM] *openEHR Archetype Object Model* (AOM), <http://www.openehr.org/releases/trunk/architecture/am/aom2.pdf>

[AOMT] openEHR *openEHR Templates* (supersedes *openEHR Archetype Templates*), <http://www.openehr.org/releases/trunk/architecture/am/tom.pdf>

[ARCH] *openEHR Archetypes: Constraint-based Domain Models for Future-proof Information Systems*, <http://www.openehr.org/publications/archetypes/archetypes_beale_oopsla_2002.pdf>

[CIMI] CIMI Reference Model Requirements, <http://informatics.mayo.edu/CIMI/index.php/CIMI_Reference_Model_Requirements>

[CTS2] OMG *Common Terminology Service 2 (CTS2)*, <http://www.omg.org/spec/CTS2/1.1/>

[HLV7v3] *HL7 Version 3 Standard: Core Principles and Properties of Version 3 Models*, <http://www.hl7.org/implement/standards/product_brief.cfm?product_id=58>

[MDMI] OMG *Model Driven Message Interoperability (MDMI), Version 1.0*, <http://www.omg.org/spec/MDMI/1.0/>

[MDR] *ISO/IEC 11179, Information Technology, -- Metadata registries*, <http://metadata-standards.org/11179/>

[NIEM] OMG *UML Profile for NIEM Version 1.0*, <http://www.omg.org/spec/NIEM-UML/1.0/>

[OCL] OMG *Object Constraint Language (OCL), Version 2.4*, <http://www.omg.org/spec/OCL/2.4/>

[ODM] OMG *Ontology Definition Metamodel (ODM) Version 1.1*, <http://www.omg.org/spec/ODM/1.1/>

[QVT] OMG *Meta Object Facility (MOF) 2.0 Query/View/Transformation, V1.2 (Beta)*, <http://www.omg.org/spec/QVT/1.2/Beta/>

[UML] OMG *Unified Modeling Language (UML) Version 2.5 – Beta 2*, <http://www.omg.org/spec/UML/2.5/Beta2/>

# Terms and Definitions

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

Archetype

An archetype is a re-usable formal definition of domain level information defined in terms of constraints on an information model. The key feature of the archetype approach to computing is a complete separation of information models (such as object models of software or models of database schemas) from domain models.

Archetype Definition Language (ADL)

ADL is a formal language for expressing archetypes. It provides a formal, textual syntax for describing constraints on any domain entity whose data is described by an information model (also known as the 'underlying reference model'). The ADL syntax is semantically equivalent to the AOM and represents one possible serialization of the AOM. The current version of ADL is known as 'ADL 2'.

Archetype Instance

An archetype instance is a single instantiation of data conforming to a specific archetype. In the context of CIMI this data will typically be clinical.

Archetype Model (AM)

An AM is a re-usable, formal model of an archetype expressed as a computable set of constraint statements on an underlying reference model (URM). Concepts that can be modeled using archetypes include weight measurement, blood pressure, microbiology results, discharge referral, prescription, or diagnosis. CIMI archetypes will be represented as an instance of the ‘Archetype Object Model’.

Archetype Object Model (AOM)

The AOM is the definitive expression of archetype semantics and is independent of any particular syntax. It is defined as an object model using a UML class diagram. It is a generic model, meaning it can be used to express archetypes for any reference model in a standard way. Version 1.4 of the AOM was standardized in ISO-13606:2. The current version is known as 'AOM 2'.

Archetype Query Language (AQL)

The AQL is a declarative query language developed specifically for expressing queries used for searching and retrieving the clinical data found in archetype-based EHRs. AQL expresses queries at the archetype level, i.e. semantic level, and not at the data instance level. This is key to achieving shared queries across system or enterprise boundaries.

Clinical Data Repository (CDR)

A CDR is a data store holding and managing clinical data collected from service encounters at the point-of-service locations such as hospitals, clinics, etc.

Clinical Document Architecture (CDA)

A CDA is an HL7 XML-based markup standard intended to specify the encoding, structure, and semantics of clinical documents for exchange.

Clinical Information Model (CIM)

A CIM is a representation of the structured clinical information (including relationships, constraints and terminology) describing a specific clinical concept - e.g. a blood pressure observation, a Discharge Summary, or a Medication Order.

Clinical Information Modeling Initiative (CIMI)

CIMI is an initiative established to “improve the interoperability of healthcare information systems through shared implementable clinical information models.”

Clinical Information Modeling Initiative (CIMI) Reference Model (RM)

The CIMI RM is the underlying Reference Model on which CIMI's clinical models (i.e. archetypes) are defined. This reference model defines a rigorous and stable set of modeling patterns, including a set of complex data types, information patterns (e.g. data, qualifier, state), and structural patterns (e.g. composition, entry, tree). All CIMI clinical models (i.e. archetypes) will be defined by constraining the CIMI RM. The RM is intended to be instantiated with patient data which conforms to the constraints defined by the associated clinical model.

Clinical Model Governance

Clinical Model Governance is a set of policies and processes through which the high clinical quality of all clinical artifacts (including clinical models and-or archetypes) is maintained during creation, storage, verification, maintenance, and distribution, by, for, and on behalf of CIMI.

Clinical Model Repository

The Clinical Model Repository is a data store holding clinical information models and associated artifacts in an agreed sharable format.

Clinical Model Verification

Clinical Model Verification is the act of reviewing, inspecting, or testing in order to establish a clinical model specification meets appropriate clinical safety and quality standards.

Clinical Modeling Language

A Clinical Modeling Language is a modeling language defining clinical information models.

Clinical Requirement

Clinical Requirements are requirements articulating clinical needs including clinical practices, standards, guidelines, principles, and other clinical concepts.

Code System

A Code System is a managed collection of uniquely identifiable concepts with associated representations. A code system may also form an ontological system for representing a set of concepts, e.g. SNOMED-CT, LOINC, ICD-10, etc.

Common Terminology Services 2 (CTS2)

CTS2 is an OMG specification providing a standard interface to disparate terminology sources. The Information Model specifies the structural definition, attributes, and associations of resources common to structured terminologies such as Code Systems, Binding Domains, and Value Sets. The Computational Model specifies the service descriptions and interfaces needed to access and maintain structured terminologies.

Concept

In information modeling, a concept represents an “idea” as a word or phrase in order to support human understanding, but may also be represented with a concept identifier in order to bind it to a controlled terminology or ontology.

Concept Domain

A Concept Domain is a named category of like concepts bound to one or more coded elements in an information model. Concept Domains exist to constrain the intent of the coded element and are independent of any specific vocabulary, code system, or Realm. A Concept Domain provides a high level grouping for all things possible in a given domain from which value sets will be constructed.

Concept Domain Binding

A Concept Domain Binding is the association of a value set with a concept domain in a given context.

Conceptual Information Model

A Conceptual Information Model is a representation of real-world objects and their relationships and constraints as understood by domain experts. A conceptual model should include no implementation-specific details.

Conformance

Conformance is the requirement that those who participate in CIMI by contributing data components or creating and sharing ADL artifacts are following the agreed-upon procedures for doing so and that all documentation meets minimum criteria and the CIMI Naming and Design Rules where applicable.

Constraint Model

A Constraint Model is a formal specification used for describing constraints on an Underlying Reference Model. The Constraint Model is used to express clinical information models (i.e. archetypes), not to be confused with the clinical information models that are instances of the constraint model.

Detailed Clinical Model

A Detailed Clinical Model is a relatively small standalone information model designed to express a precise clinical concept in a standardized and reusable manner.

Fully Defined Concept

A Fully Defined Concept is a concept uniquely defined by a set of defining relationships.

Information Model

An Information Model is a structured representation of the information requirements of a domain including the classes of information required and their attributes, relationships, and constraints.

Node

A Node is a named part of an information model.

Ontology

An Ontology is a formal representation of knowledge as a set of concept identifiers, terms describing the concepts so identified, and the relationships among them.

Reference Model

A Reference Model is an information model defining a set of modeling patterns upon which clinical models are defined.

Reference Terminology

A Reference Terminology is a terminology designed to provide common semantics for diverse implementations.

Semantic Binding

Semantic Binding is the association of a node in an information model with a concept from a controlled terminology representing its meaning.

Terminology

A Terminology is a vocabulary of technical terms used in a particular field, subject, science, or art.

Terminology Binding

Terminology Binding is the assertion of a relationship between an information model and a terminology.

Value Binding

Value Binding is the association of a given node in a clinical model with the set of valid concepts that may populate it.

Value Set

A Value Set is a set of concept identifiers deemed valid for use in a specific context, especially to define the domain of a data element.

# Symbols

## Graphical Symbols

No AML-specific graphical symbols are defined in this specification.

## Abbreviations

ADL Archetype Definition Language

AM Archetype Model

AML Archetype Modeling Language

AOM Archetype Object Model

AQL Archetype Query Language

CDA Clinical Document Architecture

CDL Clinical Document Language

CDR Clinical Data Repository

CIM Clinical Information Model

CIMI Clinical Information Modeling Initiative

CMP Constraint Model Profile

CRM Clinical Reference Model

CTS2 Common Terminology Services 2

EHR Electronic Health Record

HL7 Health Level Seven

ICD-10 International Statistical Classification of Diseases and Related Health Problems, 10th Edition

LOINC Logical Observation Identifiers Names and Codes

MDA Model Driven Architecture

OCL Object Constraint Language

OMG Object Management Group

OpenEHR Open Electronic Health Record

PIM Platform Independent Model

PSM Platform Specific Model

RM Reference Model

RMP Reference Model Profile

SNOMED CT Systematized Nomenclature of Medicine – Clinical Terms

TBP Terminology Binding Profile

UML Unified Modeling Language

URI Uniform Resource Identifier

URM Underlying Reference Model

# Additional Information

## Changes to Adopted OMG Specifications

No changes to adopted OMG specifications are required to adopt this specification.

## Acknowledgements

The following companies submitted this specification:

1. Mayo Clinic
2. Visumpoint, LLC

The following companies supported this specification:

1. Escape Velocity, LLC

The following individuals aided the development of this specification:

|  |  |
| --- | --- |
| 1. Thomas Beale | Ocean Informatics |
| 1. Dave Carlson | Intermountain Healthcare (Consultant) |
| 1. Angelique Cortez | Accenture |
| 1. Bob Daniel | Escape Velocity LLC |
| 1. Tom Digre | Visumpoint LLC |
| 1. Stanley M. Huff, MD | Intermountain Healthcare |
| 1. Patrick Langford | Intermountain Healthcare (Consultant) |
| 1. Robert Lario | Visumpoint LLC |
| 1. Jay Lyle | Ockham Information Services LLC |
| 1. Virginia Riehl | Independent Consultant |
| 1. Deepak Kumar Sharma, M.S. | Mayo Clinic |
| 1. Harold Solbrig | Mayo Clinic |
| 1. Mason Tran | Visumpoint LLC |
| 1. Michael van der Zel | University Medical Center Groningen  Results4Care |

# AML Meta Model

The AML Object Model package describes how the Archetype Definition Language (ADL) and Archetype Object Model (AOM) requirements relate to the corresponding entities the UML 2.5 Specification, ISO 11179-3 and the OMG Common Terminology Services (CTS2) 1.1 specification. It models the required features as UML classes and describes their relationship to the classes and properties in the UML specification itself.

## <Package> Archetype Meta Model

This section shows the relationship between archetype libraries, archetypes and archetype versions. It also describes the ADL/AOM specific metadata that accompanies a given version of an archetype.

### <Package> Archetype Libraries

This section describes Archetype Libraries, Archetypes and Archetype versions and their relationships.



**Archetype Libraries**

This diagram shows the relationships between the archetype library, its member archetypes and their versions and to the corresponding UML classes. It also shows the links between archetypes and their associated metadata.

#### <DataType> ArchetypeId

**Description**

An **Archetype** identifier. **ArchetypeId** must uniquely identify an **Archetype** within the context of the containing **ArchetypeLibrary**. **ArchetypId** may be extended to support specific workflows and community needs, but all implementations must support a *string* representation.

**Diagrams**

[Archetype Libraries](#_7412d314307bc61dfa0287deae4ee4ce)

**Attributes**

**• public string : String [1]**

The string representation of the archetype identifier.

#### <DataType> ArchetypeVersionId

**Description**

An **ArchetypeVersion** identifier. **ArchetypeVersionId** must uniquely identify an **ArchetypeVersion** within the context of the containing **Archetype**. **ArchetypVersionId** may be extended to support specific workflows and community needs, but all implementations must support a *string* representation.

**Diagrams**

[Archetype Libraries](#_7412d314307bc61dfa0287deae4ee4ce)

**Attributes**

**• public string : String [1]**

The string representation of the archetype version identifier.

#### <Class> Archetype

**Description**

A collection of compatible, versioned constraints on a target UML **Class** or other **Archetype**.

**Diagrams**

[Archetype Libraries](#_7412d314307bc61dfa0287deae4ee4ce)

**Attributes**

**• public archetypeId :** [**ArchetypeId**](#_abe68de6d7b599f5e4ea361caee12c81) **[1]**

An identifier that uniquely names the **Archetype** within the context of the **ArchetypeLibrary**. This identifier should be treated as immutable as it can be used to reference a particular archetype from outside of the model. In ADL, an example *archetypeId* might be "CIMI-Core-ITEM\_GROUP.cimi\_composition.v1".

**• public archetypeName : String [1]**

The human readable name of the **Archetype**. Example: "clinical data group", "laboratory test", "serum sodium", "cimi composition"

**• public rmPackagePath : String [1]**

The qualifiedName of the package containing the root class constrained by this archetype.

**• public rmClassName : String [1]**

Name of the root **Class** constrained of this archetype.

**• public originalLanguage :** [**URIAndEntityName**](#_4d1f571ab5e9384786ffe39444e822b4) **[0..1]**

The original spoken or written language in which the archetype was authored. Example: EN, DE, ES...

**• public archetypeType :** [**ArchetypeType**](#_115a19030ad491571447653210109e56) **[0..1]**

The implementation specific type or classification of the archetype. In the ADL context, this would include archetype, template\_overlay, etc.

**Associations**

**• public constrains :** [**Class**](#_a75c06fc93e516ccf92a1e38e18c46f3)**[1]**

The Reference Model class whose possible instances are constrained by the referencing Archetype. The constrained class is considered part of the Archetype's identity.

The UML **Class** constrained by the **Archetype**.

**• public version :** [**ArchetypeVersion**](#_1de96fa71501cf96b27b14f3f9f1bb99)**[1..\*]**

A version of an archetype. All versions of a given archetype must constrain the same class and the changes between versions must be backwards compatible.

The list of versions of a particular archetype, ordered from earliest to latest.

**• public specializes :** [**ArchetypeVersion**](#_1de96fa71501cf96b27b14f3f9f1bb99)**[1..\*]**

The archetype that is constrained by the referencing archetype. *specializes* is part of an archetype's identity -- if it changes, it is no longer the same archetype.

The list of versions of a particular archetype, ordered from earliest to latest.

**• public currentVersion :** [**ArchetypeVersion**](#_1de96fa71501cf96b27b14f3f9f1bb99)**[1]**

The archetype version that is considered to be the "latest" or "current" version. The current version is the referent of specializations and archetype references that omit the complete version identifier.

**Constraints**

* **uri**

[OCL]

rmURI = rmUMLModel.URI

* **package**

The rmPackagePath attribute is qualified name of the

[OCL]

self.rmPackagePath = self.constrains.namespace.qualifiedName

* **class**

The Archetype rmClassName is the name of the class that the archetype constrains

[OCL]

self.rmClassName = self.constrains.name

* **classpackage**

[OCL]

rmPackage.member->exists(c|c=constrains)

* **differentArchetype**

An archetype cannot specialize itself or any of its descendants (note - only self is included in formal OCL)

[OCL]

not (self = self.specializes)

#### <Class> ArchetypeLibrary

**Description**

A collection of archetypes that constrain one or more classes within a single reference model.

**Diagrams**

[Archetype Libraries](#_7412d314307bc61dfa0287deae4ee4ce)

**Attributes**

**• public rmPublisher : String [0..1]**

The name of the reference model publisher, which represents the rm\_publisher attribute of the AOM 2 ARCHETYPE\_HRID class as well as the archetype identifier.

**• public rmNamespace : String [0..1]**

A human-readable organization identifier corresponding to the current custodian of the artifact. In ADL this is represented as the reverse domain name of the organization. This represents the namespace attribute of the AOM 2 ARCHETYPE\_HRID class as well as the archetype identifier.

**• public rmRelease : String [0..1]**

The release id of the reference model. In ADL this would be in semver.org format, as ARCHETYPE rm\_release attribute.

**Associations**

**• public archetype :** [**Archetype**](#_f45a7b68ecac449e953ff8a65d6eff75)**[0..\*]**

**• public referenceModel :** [**Package**](#_a0a843d7d41881592e31e887cebd6da4)**[1]**

The reference model to which the library applies.

The reference model, whose member classes are constrained by the archetypes, within the library.

**Constraints**

* **constrainsInModel**

All classes constrained by archetypes in the library are members of the reference model.

[OCL]

self.archetype->forAll(a | self.referenceModel.member->exists(m = a.constrains))

#### <Class> ArchetypeVersion

**Description**

A set of constraints that can be applied as a predicate against instances of **Class** referenced by the *constrains* attribute of the containing **Archetype**.

**Diagrams**

[Archetype Libraries](#_7412d314307bc61dfa0287deae4ee4ce)

**Direct Known Subclasses (Specialization)**

[FlatArchetype](#_166addccec3fe63279b3f9eeb9930ab8), [SourceArchetype](#_dcaf9716e9bc2255b93c20393e8712f2)

**Attributes**

**• public archetypeVersionId [1]**

The identifier of the specific **ArchetypeVersion**. *archetypeVersionId* must, at a minimum, be unique within the context of the referencing archetype although some tools, such as ADL, may use globally unique identifiers. In ADL, an example of archetypeId might be "CIMI-Core-ITEM\_GROUP.cimi\_composition.v1.0.0".

**• public amlVersion : String [1]**

The version of the AML specification used to define this version of the archetype.

**• public rmRelease : String [0..1]**

The specific version of the reference model that was current when the ArchetypeVersion instance was last updated. This should be set to the *rmRelease* identifier of the archetype library that contains the archetype in which this version resides at the time that the version is created.

**Associations**

**• public definition :** [**ComplexObjectConstraint**](#_abfab8c8e983a73b4981f6fcfdd16134)**[1]**

The ComplexObjectConstraint that constraints the class referenced in the containing archetype *constrains* property.

The root constraint to be applied to the *specialized* **Archetype** if the **Archetype** is a specialization or directly to the *constrains* **Class** referenced by the containing **Archetype** if not.

**• public rules :** [**RuleStatement**](#_f8740e8d27529166da46265bd8521c94)**[0..\*]**

A collection of validation rules that assert invariants about various archetype components.

A set of rules, expressed in formal logic, that constrain various elements of the referencing archetype version with respect to other elements

**• public archetypeMetadata :** [**AuthoredResource**](#_47dea9d0676ad6870be946fa52e870ad)**[0..1]**

The source, state, language, etc. of the archetype version. The **AuthoredResource** class may be extended to include workflow and other information.

**• public archetype :** [**Archetype**](#_f45a7b68ecac449e953ff8a65d6eff75)**[1]**

A version of an archetype. All versions of a given archetype must constrain the same class and the changes between versions must be backwards compatible.

#### <Class> AuthoredResource

**Description**

**AuthoredResource** carries a minimal set of information about the source and origin of an **Archetype**. Its intent is to be a "connection point" to attach additional workflow and other provenance information to the target **Archetype***.*

**Diagrams**

[Archetype Libraries](#_7412d314307bc61dfa0287deae4ee4ce)

**Direct Known Subclasses (Specialization)**

[ADLResource](#_cd31e9a8f0e3c565868ba26256810179)

**Attributes**

**• public isControlled : Boolean [1]**

A flag indicating whether the archetype is change-controlled or not can be included after the version. Archetypes that include the “controlled” flag should have the revision history section included, while those with the “uncontrolled” flag, or no flag at all, may omit the revision history. This enables archetypes to be privately edited in an early development phase without generating large revision histories of little or no value.

**• public isGenerated : Boolean [1]**

A flag indicating whether the archetype was generated or authored. This marker is used to support the migration to differential archetype representation introduced in ADL 1.5, to enable proper representation of specialised archetypes.

**• public resourceSource :** [**URI**](#_887928f30f99c8a1ca89ed7a082356aa) **[0..1]**

A URI that references the source document (if any) from which the original resource was derived.

**• public resourceDocumentLanguage : String [0..1]**

The language (e.g. AOM, CEM, ...) of the source of the constraints, if any.

**• public resourceDocumentSyntax : String [0..1]**

The syntax of the resource document (ADL, XML, XMI, ...).

**• public resourceLanguageVersion : String [0..1]**

The version of the resourceDocumentLanguage (e.g. ADL 1.5, ADL 2.0, XMI 2.1 etc).

**• public resourceSourceUID : String [0..1]**

An external identifier that uniquely identifies this *Archetype*. The format and structure of this identifier are determined by the rules of the *resourceDocumentLanguage* and/or *resourceDocumentSyntax.* This identifier cannot be used as an identifier within AML itself as it may not always be present. It must be preserved, however, for export to external resources.

**Associations**

**• public describedArchetype :** [**ArchetypeVersion**](#_1de96fa71501cf96b27b14f3f9f1bb99)**[1]**

The archetype version to which the **AuthoredResource** metadata applies.

#### <Class> Class

**Description**

The purpose of a Class is to specify a classification of objects and to specify the Features that characterize the structure and behavior of those objects.

**Diagrams**

[Archetype Libraries](#_7412d314307bc61dfa0287deae4ee4ce), [Object Constraint References](#_960b4dbbd4fb12cd1ac74b16e654b984), [Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054), [Template Metamodel](#_d132633918c4cbb0c3c69e16de3df9c2)

**Direct Known Superclasses (Generalization)**

[Classifier](#_31f3bed9860f1a34043799bd12ffe873)

**Associations**

**• public ownedAttribute :** [**Property**](#_652433ba6af347b5ab3d4b0b4b2931c9)**[0..\*]**

An attribute that is defined in a Class instance.

*ownedAttribute* subsets *ownedMember*, meaning that the set of RMMProperty names must be unique within the context of the owning RMClass.

**• public ownedTemplateSignature :** [**RedefinableTemplateSignature**](#_6afdd25f5589999ef5ae78a4eab8563d)**[0..1]**

**• public templateBinding :** [**TemplateBinding**](#_039ec0a61521832e985575d3d9688234)**[0..\*]**

**• public attribute :** [**Property**](#_652433ba6af347b5ab3d4b0b4b2931c9)**[0..\*]**

The union of the ownedAttributes of the owning Class and the closure if its super classes. In the case where two or more attribute Properties have the same name, only the most proximal name is visible, and AML cannot address or constrain attributes with the same name in parent or ancestor classes.

**• public superClass :** [**Property**](#_652433ba6af347b5ab3d4b0b4b2931c9)**[0..\*]**

The superclasses of a Class, derived from its Generalizations

#### <Class> ComplexObjectConstraint

**Description**

A collection of constraints on an instance of an instance of a UML **Class.** A **ComplexObjectConstraint** may constrain the existence, cardinality or possible values on one or more of the constrained class attributes.

**Diagrams**

[Archetype Libraries](#_7412d314307bc61dfa0287deae4ee4ce), [Object Constraint References](#_960b4dbbd4fb12cd1ac74b16e654b984)

**Direct Known Superclasses (Generalization)**

[NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9)

**Attributes**

**• public assumedValue :** [**RMClassInstance**](#_935cd7de4b22d47dd2c6aef93bed5c7a) **[0..1]**

Value to be assumed in instances in which no value is provided. **Note**: assumedValue for a constrained class is not implemented in the AML profile.

**Associations**

**• public attributeTuple :** [**AttributeTupleConstraint**](#_f6da15c71717330ae1b56f8b41e3dd51)**[0..\*]**

A set of correlated constraints or "tuples", each of which simulataneously constraints two or more *attributes* in the referenced UML **Class**.

**• private targetObject :** [**ObjectConstraintProxy**](#_6da4a9bc7db41a2b89064f79f0c4ed36)**[0..\*]**

**• public constrains :** [**Class**](#_a75c06fc93e516ccf92a1e38e18c46f3)**[1]**

A constraint on the cardinality or attributes of a UML Class.

**Constraints**

* **instanceOfConstraint**

[English]

If assumedValue exists, assumedValue.classifier must be equal to or a specialization of self.parent

#### <Class> FlatArchetype

**Description**

A **FlatArchetype** is generated from one or more **SourceArchetypes** via an external flattening process. The flattening operation:

* Replaces ComplexObjectConstraintProxies with ComplexObjectConstraints that contain copies of the subtrees to which they point.
* Applies SourceArchetype overlays to the parent structure resulting in a full archetype structure.

From the UML perspective, archetypes are already in the fully "flattened" form, where the inherited *member* properties and constraints represent the closure of the properties and constraints expressed at the leaf or source level.

**Diagrams**

[Archetype Libraries](#_7412d314307bc61dfa0287deae4ee4ce)

**Direct Known Superclasses (Generalization)**

[ArchetypeVersion](#_1de96fa71501cf96b27b14f3f9f1bb99)

**Associations**

**• public flatteningOf :** [**SourceArchetype**](#_dcaf9716e9bc2255b93c20393e8712f2)**[1]**

The root source archetype from which a flat archetype is generated. A FlatArchetype is directly derived from (is a flatteningOf) exactly one SourceArchetype. Note, however, that one or more additional archetypes may be included in the flattening process, as the parent, parent of the parent, etc. of the flatteningOf SourceArchetype will also be replicated during the flattening process as well as the flattening of any component archetypes that are directly or indirectly referenced by the source.

The **SourceArchetype** from which the **FlatArchetype** was derived.

**Constraints**

* **noSpecialization**

A flat archetype cannot specialize another archetype

[OCL]

not exists(archetype.specializes)

#### <Class> Package

**Description**

A **Package** is a **Namespace** for its members, which comprise those elements associated via packagedElement (which are said to be owned or contained), and those imported.

**Diagrams**

[Archetype Libraries](#_7412d314307bc61dfa0287deae4ee4ce)

**Direct Known Superclasses (Generalization)**

[Namespace](#_f762e4ef59f1948849a49d421126c16b)

**Attributes**

**• public URI : String [0..1]**

Provides an identifier for the package that can be used for many purposes. A URI is the universally unique identification of the package following the IETF URI specification, RFC 2396 http://www.ietf.org/rfc/rfc2396.txt and it must comply with those syntax rules.

#### <Class> RuleStatement

**Description**

An assertion or collection of assertions that allow the expression of additional invariants across archetypes and reference model artifacts that cannot be directly expressed directly as AML Constraints.

**Diagrams**

[Archetype Libraries](#_7412d314307bc61dfa0287deae4ee4ce)

**Direct Known Superclasses (Generalization)**

[RuleElement](#_aca03c36f52ec3dbae79ae94604d6df9)

**Direct Known Subclasses (Specialization)**

[Assertion](#_f9e7c553caf3e674732fe386e3d45466), [VariableDeclaration](#_8a634b04f92ff4c449cdcaaae16ba015)

#### <Class> SourceArchetype

**Description**

The source form of an archetype, potentially including references to other archetypes whose contents are not explicitly reproduced in the source form.

**Diagrams**

[Archetype Libraries](#_7412d314307bc61dfa0287deae4ee4ce)

**Direct Known Superclasses (Generalization)**

[ArchetypeVersion](#_1de96fa71501cf96b27b14f3f9f1bb99)

**Associations**

**• public specializes :** [**SourceArchetype**](#_dcaf9716e9bc2255b93c20393e8712f2)**[1]**

Archetypes can be specialised in a similar way to classes in object-oriented programming languages. Common to both situations is the use of a differential style of declaration, i.e. the contents of a specialised entity are expressed as differences with respect to the parent - previously defined elements from the parent that are not changed are not repeated in the descendant. The basic test that must be satisfied by a specialised archetype is as follows:

All possible data instance arrangements that conform to the specialised archetype must also conform to all of its parents, recursively to the ultimate parent. This condition ensures that data created by a specialised archetype that is not itself shared by two systems can be processed by the use of a more general parent that is shared. The semantics that allow this are similar to the ‘covariant redefinition’1 notion used in some object oriented programming languages, and can be summarised as follows:

A non-specialised (i.e. top-level) archetype defines an instance space that is a subset of the space defined by the class in the reference information model on which the archetype is based

A specialised archetype can specialise only one parent archetype, i.e. single inheritance. A specialised archetype defines an instance space defining the following elements:

unchanged object and attribute constraints inherited from the parent archetype.

one or more:

redefined object constraints, that are proper subsets of the corresponding parent object constraints;

redefined attribute constraints, that are proper subsets of the corresponding parent attribute constraints;

extensions, i.e. object constraints added to a container attribute with respect to the corresponding attribute in the parent archetype, but only as allowed by the underlying reference model.

All elements defined in a parent archetype are either inherited unchanged or redefined in a specialised child.

Specialised archetypes are expressed differentially with respect to the parent, i.e. they do not mention purely inherited elements, only redefinitions and extensions.

Extensions always define an additional subset of the instance space defined by the reference model element being extended (i.e. to which the ‘new’ objects belong). The extension capability allows archetypes to remain extensible without having to know in advance how or if they will be extended.

The **SourceArchetype** from which the **FlatArchetype** was derived.

**Constraints**

* **sameArchetypes**

[OCL]

exists(self.specializes) implies self.archetype.specializes = self.specializes.archetype.specializes

#### <Enumeration> ArchetypeType

**Description**

An implementation specific classification of **Archetype** types. In the ADL context, the ADL 2.0.5 specification describes the following types:

* archetype
* template
* template\_overlay
* operational\_template

This would be implemented with a specialization of the **ArchetypeType** enumeration with the four literals above. **ArchetypeType** does not affect the semantics of an AML implementation.

**Diagrams**

[Archetype Libraries](#_7412d314307bc61dfa0287deae4ee4ce)

### <Package> Archetypes and the UML Reference Model

This section shows the details of the relationships between archetype libraries, archetypes, archetype versions and the corresponding UML 2.5 classes. Note that the UML classes are shaded grey.



**ArchetypeRM**

An Archetype references (or constrains) a single UML Class.

The constrained Class must be a member of the UML Package that is constrained by the Archetype Library.

#### <Class> Classifier

**Description**

A **Classifier** has a set of **Features**, some of which are **Properties** called the attributes of the **Classifier**.

**Diagrams**

[ArchetypeRM](#_8371278cdefade515c9d54e183ad347b), [Object Constraint References](#_960b4dbbd4fb12cd1ac74b16e654b984), [Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054), [Template Metamodel](#_d132633918c4cbb0c3c69e16de3df9c2)

**Direct Known Superclasses (Generalization)**

[Namespace](#_f762e4ef59f1948849a49d421126c16b)

**Direct Known Subclasses (Specialization)**

[Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [DataType](#_d5914eb0da42172989bbe57f23fc4310)

**Attributes**

**• public isFinalSpecialization : Boolean [1]**

If true, the **Classifier** instance cannot be specialized or constrained.

**Associations**

**• public templateParameter :** [**ClassifierTemplateParameter**](#_3d9b09fe9052c8305d90ab92bc37d26b)**[0..1]**

#### <Class> NamedElement

**Description**

NamedElement is the superclass of all named elements in the Reference Model, and represents the subset of UML::NamedElements that are referenced by the AML profile.

While a Reference Model may contain UML::NamedElements without names, an AML Archetype can only constrain those with have names and are of type PrimitiveDataType, Class or Property or a descendant thereof.

**Diagrams**

[ArchetypeRM](#_8371278cdefade515c9d54e183ad347b), [Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054)

**Direct Known Subclasses (Specialization)**

[EnumerationLiteral](#_41faf6a7041d7068bbbf4f9ff3924d22), [Namespace](#_f762e4ef59f1948849a49d421126c16b), [Property](#_652433ba6af347b5ab3d4b0b4b2931c9)

**Attributes**

**• public name : String [1]**

The name of the Reference Model element. Name must be unique within the context of the owning *namespace.*

**• public qualifiedName : String [1]**

A name that allows the NamedElement to be identified within a hierarchy of nested Namespaces. It is constructed from the names of the containing Namespaces starting at the root of the hierarchy and ending with the name of the NamedElement itself.

**Associations**

**• public namespace :** [**Namespace**](#_f762e4ef59f1948849a49d421126c16b)**[0..1]**

The namespace that owns the named element.

#### <Class> Namespace

**Description**

An element in a model that owns and/or imports a set of NamedElements that can be identified by name.

**Diagrams**

[ArchetypeRM](#_8371278cdefade515c9d54e183ad347b), [Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054)

**Direct Known Superclasses (Generalization)**

[NamedElement](#_527fd9eb1e787c36a3748854a9431816)

**Direct Known Subclasses (Specialization)**

[Classifier](#_31f3bed9860f1a34043799bd12ffe873), [Package](#_a0a843d7d41881592e31e887cebd6da4)

**Associations**

**• public member :** [**NamedElement**](#_527fd9eb1e787c36a3748854a9431816)**[0..\*]**

Specifies the Namespace that owns the NamedElement.

A collection of **NamedElements** identifiable within the **Namespace**, either by being owned or by being introduced by importing or inheritance.

**• public ownedMember :** [**NamedElement**](#_527fd9eb1e787c36a3748854a9431816)**[0..\*]**

A collection of NamedElements owned by the Namespace.

**Known other classes**

[Archetype](#_f45a7b68ecac449e953ff8a65d6eff75), [ArchetypeLibrary](#_214dd64d9ba4a03a70f9f9f80f8c47d2), [Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [Package](#_a0a843d7d41881592e31e887cebd6da4)

### <Package> ADL Archetype Metadata

This section describes the AOM 2.0 specific metadata that accompanies the AuthoredResource archetype description. For the purposes of the AML specification, this section is intended to:

1. Provide a concrete example of the sort of provenance and workflow related metadata that will accompany archetypes
2. Provide a model that can be used to extend the metadata in the AML profile to accomodate ADL/AOM specific metadata requirements.



**Metadata Object Model**

This diagram show the ADL 2.0 metadata applied as an specialize to the AuthoredResource class. Note that different implementations of AML may have different AuthoredResource specializations.

#### <Class> ADLResource

**Description**

An **AuthoredResource** in the ADL/AOM 2.0 context.

**Diagrams**

[Metadata Object Model](#_75f885fd66dc2690c9965931a1d13720)

**Direct Known Superclasses (Generalization)**

[AuthoredResource](#_47dea9d0676ad6870be946fa52e870ad)

**Associations**

**• public translation :** [**ResourceTranslation**](#_b5ca6c13619e9a3d461d676d4a7daa58)**[0..\*]**

Information about translations of the resource. In the ADL context, there must be a **TranslationDetails** record for every language used in an **ItemDescription**.

**• public description :** [**Description**](#_88968fc275e704f97d56f651857679d9)**[0..1]**

The description of the resource.

#### <Class> Description

**Diagrams**

[Metadata Object Model](#_75f885fd66dc2690c9965931a1d13720)

**Attributes**

**• public lifecycleState :** [**LifecycleState**](#_38271094f762e64fa7b50e4ddfd130be) **[0..1]**

The state of an ADLResource in terms of a defined lifecycle.

**• public resourcePackageUri : String [0..1]**

The resource package URI as defined in the ADL specification.

**Associations**

**• public description :** [**ResourceDescription**](#_a1d220d9249c2add7f803e925a99d503)**[1]**

The description of this particular resource.

#### <Class> ResourceDescription

**Description**

A detailed description of the source, provenance, copyright, etc of the ADL Resource.

**Diagrams**

[Metadata Object Model](#_75f885fd66dc2690c9965931a1d13720)

**Attributes**

**• public originalAuthor :** [**TagValue**](#_7ce4def81f76bf28ab4cf4bf2c4fd567) **[1..\*]**

The name, email address, contact information and other information about the original author of the resource.

**• public originalNamespace : String [0..1]**

The namespace of the owning organization that originally created the resource.

**• public originalPublisher : String [0..1]**

The name of the original publisher of the resource

**• public otherContributors : String [0..\*]**

The names of other contributors to the resource.

**• public copyright : String [0..1]**

Copyright information about the resource.

**• public license : String [0..1]**

Licensing information about the resource.

**• public custodianNamespace : String [0..1]**

The identifier of the organization currently responsible for the resource.

**• public custodianOrganisation : String [0..1]**

The name of the organization currently responsible for the resource.

**• public otherDetails :** [**TagValue**](#_7ce4def81f76bf28ab4cf4bf2c4fd567) **[0..\*]**

Other miscellaneous details about the resource in the form of tag/value pairs.

**Associations**

**• public details :** [**ResourceDescriptionItem**](#_3ac0cbf518892e4287800d79b7b28bca)**[0..\*]**

Language specific details about the resource.

**• public descriptionOf :** [**Description**](#_88968fc275e704f97d56f651857679d9)**[1..\*]**

The collection of resources to which the description applies. The information in a **ResourceDescription** may apply to one or more **ADLResources**.

#### <Class> ResourceDescriptionItem

**Diagrams**

[Metadata Object Model](#_75f885fd66dc2690c9965931a1d13720)

**Attributes**

**• public language :** [**URIAndEntityName**](#_4d1f571ab5e9384786ffe39444e822b4) **[1]**

The language of the particular description.

**• public purpose : String [1]**

The purpose of the particular archetype.

**• public keywords : String [0..\*]**

A collection of keywords about the resource.

**• public use : String [0..1]**

A description of the intended use of the resource.

**• public misuse : String [0..1]**

A description of places where the resource should not be used.

**• public originalResourceUri :** [**TagValue**](#_7ce4def81f76bf28ab4cf4bf2c4fd567) **[0..\*]**

The original URI of the resource.

**• public otherDetails :** [**TagValue**](#_7ce4def81f76bf28ab4cf4bf2c4fd567) **[0..\*]**

Other miscellaneous details about the resource as tag/value pairs.

**Constraints**

* **language**

[OCL]

self.language.classifier.stereotypedBy('Language')

#### <Class> ResourceTranslation

**Description**

A collection of translations for a resource.

**Diagrams**

[Metadata Object Model](#_75f885fd66dc2690c9965931a1d13720)

**Attributes**

**• public versionLastTranslated :** [**ArchetypeVersionId**](#_561a78931042144a362602066e152651) **[0..1]**

The **ArchetypeVersion** identifier of the version that was last translated.

**Associations**

**• public details :** [**TranslationDetails**](#_dec552b5ef914f87d2d3ea873667fac2)**[1]**

#### <Class> TagValue

**Description**

A tag/value pair that provides additional information whose semantics is outside the scope of the specification.

**Diagrams**

[Metadata Object Model](#_75f885fd66dc2690c9965931a1d13720)

**Attributes**

**• public tag : String [1]**

The tag identifier.

**• public value : String [0..1]**

The associated value.

#### <Class> TranslationDetails

**Diagrams**

[Metadata Object Model](#_75f885fd66dc2690c9965931a1d13720)

**Attributes**

**• public language :** [**URIAndEntityName**](#_4d1f571ab5e9384786ffe39444e822b4) **[1]**

The resource reference of the translation language.

**• public author :** [**TagValue**](#_7ce4def81f76bf28ab4cf4bf2c4fd567) **[1..\*]**

Information about the translator which may include name, affiliation, contact information and other details.

**• public accreditation : String [0..1]**

Information about the translation accreditation agency, etc. for the author.

**• public otherDetails :** [**TagValue**](#_7ce4def81f76bf28ab4cf4bf2c4fd567) **[0..\*]**

Other details about the particular translation.

**Associations**

**• public resource :** [**ResourceTranslation**](#_b5ca6c13619e9a3d461d676d4a7daa58)**[1..\*]**

**Constraints**

* **languageEnumeration**

[OCL]

self.language.classifier.stereotypedBy('Language')

**Known other classes**

[AuthoredResource](#_47dea9d0676ad6870be946fa52e870ad)

#### <Enumeration> LifecycleState

**Description**

The valid life cycle states of an ADL resource, as defined in the openEHR Knowledge Artifact Identification specification.

**Diagrams**

[Metadata Object Model](#_75f885fd66dc2690c9965931a1d13720)

**Enumeration Literals**

* **deprecated**

* **development**

* **published**

* **rejected**

* **release\_candidate**

* **unmanaged**

## <Package> Reference Model Meta Model

This section identifies the subset of the UML 2.5 Specification that is used in the definition of the AML Object Model. Classes with direct UML analogs will be shown in grey. We have removed generalizations, associations and properties that are addressed in the AML specifciation and, in several cases, have flattened inherited attributes and associations into a single class.

### <Package> Primitive Data Types

In the AML/ADL context, the term "primitive data type" is used to indicate "leaf nodes" -- data elements that are treated as being atomic and are only constrained in terms of their possible value ranges. AML data types, like UML data types, are "model Types whose instances are distinguished only by their value" but, unlike the UML definition of "Primitive Type", AML primitive types can embody the notion of substructure.

When a AML profile is applied to a UML Reference Model, it may be necessary to map one or more of the AML types to corresponding types in the target model.



**Intervals**

This diagram shows data types for intervals that can be used to construct AML primitive type constraints. The duration intervals are realized by substitution of their respective types through the parameterized Interval class.

#### <Class> DateInterval

**Description**

An interval over an AML DatePrimitive type.

**Diagrams**

[Intervals](#_632ab8c7f599e54834cd79cfa711f90b)

#### <Class> DateTimeInterval

**Description**

A interval over an AML DateTimePrimitive type.

**Diagrams**

[Intervals](#_632ab8c7f599e54834cd79cfa711f90b)

#### <Class> DurationInterval

**Description**

A interval over an AML DurationPrimitive type.

**Diagrams**

[Intervals](#_632ab8c7f599e54834cd79cfa711f90b)

#### <Class> IntegerInterval

**Description**

A interval over an AML IntegerPrimitive type.

**Diagrams**

[Intervals](#_632ab8c7f599e54834cd79cfa711f90b)

#### <Class> Interval

**Diagrams**

[Intervals](#_632ab8c7f599e54834cd79cfa711f90b)

**Attributes**

**• public lower :** [**T**](#_e36b5e306e1664a3e0bc600785109705) **[0..1]**

A lower limit or start of the interval.

**• public upper :** [**T**](#_e36b5e306e1664a3e0bc600785109705) **[0..1]**

An upper limit or end of the interval.

**• public lowerIncluded : Boolean**

A flag to indicate whether lower limit or start of the interval is included as part of the interval.

**• public upperIncluded : Boolean**

A flag to indicate whether upper limit or end of the interval is included as part of the interval.

#### <Class> RealInterval

**Description**

A interval over an AML RealPrimitive type.

**Diagrams**

[Intervals](#_632ab8c7f599e54834cd79cfa711f90b)

#### <Class> TimeInterval

**Description**

A interval over an AML TimePrimitive type.

**Diagrams**

[Intervals](#_632ab8c7f599e54834cd79cfa711f90b)



**PrimitiveDataTypes**

This diagram shows the set of primitive data types that can be constrained using AML primitive type constraints along with the various match patterns that are used in to construct constraints.

Date and Time constraints follow the string syntax of the *ISO 8601 Representation of dates and times*. Constraint patterns are formed by using the ‘?’ (optional) or ‘x’ (prohibited) characters.

The syntax of legal patterns is shown by the following regular expressions:

**date\_pattern**: yyyy-(mm|??|XX)-(dd|??|XX)

**time\_pattern**: hh:(mm|??|XX):(ss|??|XX)

**time\_in\_date\_pattern**: T(hh|??|XX):(mm|??|XX):(ss|??|XX)

**date\_time\_pattern**: date\_pattern time\_in\_date\_pattern

Date and time intervals also require operators to define boundaries. The greater than, less than, and equal s signs are used to indicate values bounded by a value. Two stops (periods) are used to define an interval between two values.

**Single value pattern**: |> 09:30:00| -- any time after 9:30 am

**Interval pattern**: |2004-05-20..2004-06-02| -- a date range

Durations use the ISO 8601 prefix ‘P’ (period) followed by a set of unit/magnitude pairs indicating the number of years, months, weeks, days, and (with a ‘T’ time delimiter), minutes, etc. intended.

**Date duration template**: Pwd -- a duration containing weeks and/or days only, e.g. P4w

**Time duration template**: PThm -- a duration containing hours and/or minutes only, e.g. PT2h30m

**DateTime duration pattern**: P1dT8h -- 1 day 8 hrs

**Time range pattern**: |PT0m..PT1m30s| -- Reasonable time offset of first apgar sample

#### <DataType> AnyPrimitiveType

**Description**

The abstract super type of AML primitive data types. This type typically maps to a type like "Any" or "Object" in an object system; it is defined here to provide the value and reference equality semantics

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Subclasses (Specialization)**

[BooleanPrimitive](#_19737cdaaaee2179c50553b52361808b), [DatePrimitive](#_db5d020506d0af7a330f0b4fe1cb870a), [DateTimePrimitive](#_8347879bb381db17040637cc3ba0a25c), [DurationPrimitive](#_64b0026498682fb721ccdb38c186bba2), [IntegerPrimitive](#_89e2b2b9de405e6d05c4c5259fc8ffd6), [RealPrimitive](#_c596f10fb93cb4f697f2f1b0b64b43ce), [StringPrimitive](#_6a90be7cfa784ea4b4e8cad8f4a47e82), [TimePrimitive](#_05e0f2a221f0733d693058e9253b0017)

#### <DataType> BooleanPrimitive

**Description**

A type with two values supporting binary logic.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

#### <DataType> DateMatchPattern

**Description**

A string pattern defining valid Date values

Following ISO 8601 for literal representations, the date match pattern uses the letters ‘y’, ‘m’, and ‘d’ to indicate required but unspecified characters, ‘?’ to indicate optional characters, and ‘x’ to indicate prohibited characters.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

#### <DataType> DatePrimitive

**Description**

A primitive type specifying a calendar date.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

#### <DataType> DateTimeMatchPattern

**Description**

A string pattern defining valid DateTime values.

Following ISO 8601 for literal representations, the time match pattern uses the letters ‘y’, ‘m’, ‘d’, ‘h’, ‘m’, ‘s’ to indicate required but unspecified characters, ‘?’ to indicate optional characters, and ‘x’ to indicate prohibited characters.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

#### <DataType> DateTimePrimitive

**Description**

A point in time with a given precision.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

#### <DataType> DurationMatchPattern

**Description**

A set of Duration value ranges, any value of which is considered valid (e.g., ‘|>= P5d|’, ’P5d..P8d’)

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

#### <DataType> DurationPrimitive

**Description**

A primitive type specifying a quantity of time.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

#### <DataType> IntegerPrimitive

**Description**

A primitive type specifying an integer.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

#### <DataType> RealPrimitive

**Description**

A real number.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

#### <DataType> RegularExpression

**Description**

A Perl regular expression defining a string pattern.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

#### <DataType> StringPrimitive

**Description**

A character string.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

#### <DataType> TimeMatchPattern

**Description**

A string pattern defining valid Time values.

Following ISO 8601 for literal representations, the time match pattern uses the letters ‘h’, ‘m’, and ‘s’ to indicate required but unspecified characters, ‘?’ to indicate optional characters, and ‘x’ to indicate prohibited characters.

**Diagrams**

[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

#### <DataType> TimePrimitive

**Description**

A point in time.

**Diagrams**

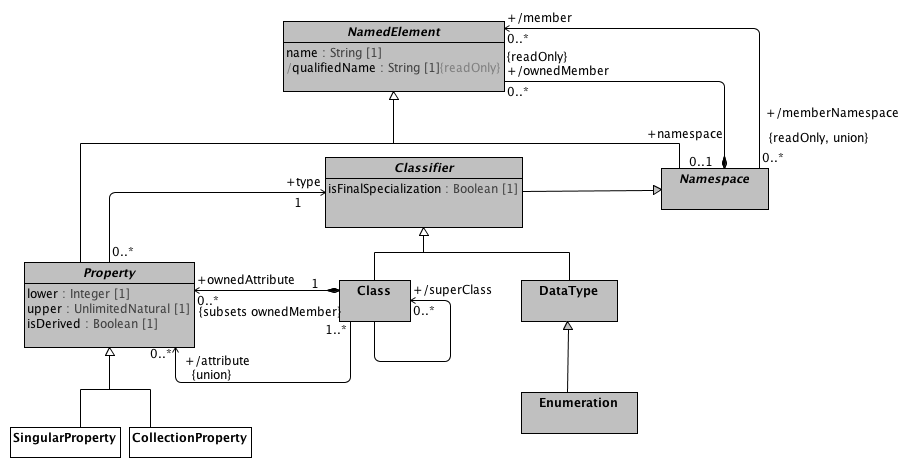
[PrimitiveDataTypes](#_1b4f676a5a33d3d5b1673552f8bd8ac1)

**Direct Known Superclasses (Generalization)**

[AnyPrimitiveType](#_72b7d3db018aa6b2ac9fd5dcdf816a12)

### <Package> Reference Metamodel

This section describes the aspects of the basic UML building blocks (Class, Property, DataType, Enumeration) that are used by the AML Object Model. Note that the representation, while faithful to UML 2.5, has been flattened and simplified.



**Reference Metamodel**

This diagram shows the aspects of the UML Classifier, Property and DataType elements that are used in the AML Object Model. Some of the aspects of the model such as the MultiplicityElement have been flattened for simplification. Note that the AML Object Model specializes UML Property by differentiating properties with an upper multiplicity value of 1 and an upper multiplicity value greater than one.

#### <Class> CollectionProperty

**Description**

CollectionProperty represents the subset of UML Property instances that can occur more than one time. A CollectionProperty instance is viewed by AML as a collection of objects of a given type that possess two separate characteristics:

* The collection as a whole may be required, optional or prohibited.
* The cardinality of the collection may be constrained.

This combination allows a number of useful constructs, including:

* Requiring that a list be present but that it have no members, which can be used to assert a relationship between an object and an empty set of objects
* Making an attribute optional, but, if present, requiring that it have a minimum number of members

**Diagrams**

[Attribute Constraint References](#_39027eaec61a2eaccc1fccb451cdda98), [Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054)

**Direct Known Superclasses (Generalization)**

[Property](#_652433ba6af347b5ab3d4b0b4b2931c9)

#### <Class> DataType

**Description**

A type whose instances are distinguished only by their value.

**Diagrams**

[Enumeration Metamodel](#_c323459faa5aa97d8abc3c64dcd86661), [Object Constraint References](#_960b4dbbd4fb12cd1ac74b16e654b984), [Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054)

**Direct Known Superclasses (Generalization)**

[Classifier](#_31f3bed9860f1a34043799bd12ffe873)

**Direct Known Subclasses (Specialization)**

[Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e), [PrimitiveType](#_2fd719f1d4a4fafd390224b10cd96510)

#### <Class> Enumeration

**Description**

A kind of data type that specifies a set of EnumerationLiterals. Each value of an Enumeration corresponds to one of its user-defined EnumerationLiterals.

**Diagrams**

[Enumeration Metamodel](#_c323459faa5aa97d8abc3c64dcd86661), [Object Constraint References](#_960b4dbbd4fb12cd1ac74b16e654b984), [Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054)

**Direct Known Superclasses (Generalization)**

[DataType](#_d5914eb0da42172989bbe57f23fc4310)

**Direct Known Subclasses (Specialization)**

[EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51)

**Associations**

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

**Constraints**

* **enumerateValueDomain**

#### <Class> Property

**Description**

A simplified representation of a UML **Property**. The aspects of a property that can be referenced by an archetype includes:

* *name* -- The property name. Only named properties may be constrained within a reference model. A non-hidden Property is always referred to in its unqualified form. If it is necessary to reference hidden elements within an archetype, the qualified name (N::x) form should be used. Qualification should be the minimum that sufficient to render the name unique.
* *lower* and *upper* -- The lower and upper bounds. The UML MultiplicityElement shows lower and upper as derived properties. The AML specification assumes that, if present, these properties have been computed and it is up to the implementer to correctly interpret **MultiplicityElement** *lowerValue* and upperValue properties to determine these results.
* *isDerived* -- Derived properties cannot be constrained using AML -- the rationale being that (a) derived properties may or may not be present in object instances and (b) the primary constraints need to be applied to the parameters of the derivation rather than the result.

UML Associations are not used in AML -- the model is traversed via the *ownedAttribute* property . **AssociationClasses** are not differentiated from any other Reference Model **Class** in the model, and may be referenced and traversed via. whatever *ownedAttribute*/*type* links that are available.

While default values can be specified in the Reference Model, they are ignored in AML. Note, however, that AML can specify assumed values.

All other UML Property links, including *aggregation*, *isComposite*, *isID*, *association*, *qualifier*, *opposite*, *redefines*, *subsettedProperty*, and *interface* are ignored in the AML object model.

**Diagrams**

[Attribute Constraint References](#_39027eaec61a2eaccc1fccb451cdda98), [Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054), [Template Metamodel](#_d132633918c4cbb0c3c69e16de3df9c2)

**Direct Known Superclasses (Generalization)**

[NamedElement](#_527fd9eb1e787c36a3748854a9431816)

**Direct Known Subclasses (Specialization)**

[CollectionProperty](#_702e30cd0381b6e9726bcc6fe779a70f), [SingularProperty](#_5917d2795b1a9ae4f33929e6edb8af81)

**Attributes**

**• private lower : Integer [1]**

The lower bound of the multiplicity interval.

**• private upper : UnlimitedNatural [1]**

The upper bound of the multiplicity interval.

**• public isDerived : Boolean [1]**

Specifies whether the Property is derived, i.e., whether its value or values can be computed from other information.

**Associations**

**• public type :** [**Classifier**](#_31f3bed9860f1a34043799bd12ffe873)**[1]**

The type of the Property.

**Constraints**

* **nonDerived**

[OCL]

isDerived=false

#### <Class> SingularProperty

**Description**

SingularProperty represents the subset of reference model Property instances having an upper bound of 1. AML treats Singular properties as single values (vs. collections) of attributes that can be required, optional or prohibited.

**Diagrams**

[Attribute Constraint References](#_39027eaec61a2eaccc1fccb451cdda98), [Reference Metamodel](#_4067d3d86b09d7e8c4b542bda7773054)

**Direct Known Superclasses (Generalization)**

[Property](#_652433ba6af347b5ab3d4b0b4b2931c9)

**Constraints**

* **singular**

[OCL]

upper = 1

* **collection**

[OCL]

upper > 1 or upper.isUnlimited()

**Known other classes**

[Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [Classifier](#_31f3bed9860f1a34043799bd12ffe873), [NamedElement](#_527fd9eb1e787c36a3748854a9431816), [Namespace](#_f762e4ef59f1948849a49d421126c16b)

### <Package> Attribute Constraint References

This section describes the relationships between AML attribute constraints and the corresponding specializations of the UML Property



**Attribute Constraint References**

#### <Class> AttributeCollectionConstraint

**Description**

A constraint on an attribute with an upper multiplicity of greater than one. An AttributeCollectionConstraint may assert an ordering on the member AttributeCollectionMembers and may assert the minimum and maximum number of instances that may occur for the referenced attribute.

**Diagrams**

[Attribute Constraint References](#_39027eaec61a2eaccc1fccb451cdda98)

**Direct Known Superclasses (Generalization)**

[AttributeConstraint](#_11f887fb6f19248bf7193bca31772c05)

**Attributes**

**• public collectionType :** [**CollectionType**](#_ac8c7771bf7f68d6747022e6924749ca) **[0..1]**

An indicator that asserts whether the referenced property members must be unique and/or ordered. This attribute applies at the model level and describes the ordering and uniqueness of the targets of the collection property itself, as opposed to the UML isOrdered and isUnique, which applies at the instance level.

**• public cardinality :** [**MultiplicityInterval**](#_c810ec7fa381fa249b7a7d9fecae85b6) **[0..1]**

The minimum and maximum number of collection property instances that are allowed. If absent, the cardinality of the referenced *attribute* applies.

**Note:** UML specifics a default minimum and maximum cardinality if the lower or upper values are omitted. These rules cannot be applied in the AML case. In AML, the absence of a stated cardinality means that the minimum and maximum values default to those of the constrained attribute.

**Associations**

**• public member :** [**AttributeCollectionMember**](#_177e37623ae3f5642980fd445bf78af1)**[1..\*]**

A (possibly named) subset of the set of possible attributes referenced by the **AttributeCollectionConstraint**. The set is ordered which allows attributes to be declared to appear before (ADL isBefore) or after (ADL isAfter) one another.

**• public constrains :** [**CollectionProperty**](#_702e30cd0381b6e9726bcc6fe779a70f)**[1]**

A constraint on the existence, cardinality or type of a collection property.

**Constraints**

* **collectionType**

[OCL]

parentProperty.isOrdered implies collectionType = CollectionType::LIST and parentProperty.isUnique implies collectionType = CollectionType.SET

* **cardinality**

[OCL]

cardinality.minimum >= parent.lower and (cardinality.maximum = unlimitedValue() or parent.upper = unlimitedValue() or cardinality.maximum <= parent.upper)

#### <Class> AttributeConstraint

**Description**

A constraint on an *attribute* of a **Class** or **DataType**

**Diagrams**

[Attribute Constraint References](#_39027eaec61a2eaccc1fccb451cdda98)

**Direct Known Subclasses (Specialization)**

[AttributeCollectionConstraint](#_5eefba8eca7402f09bd5619804038771), [SingularAttributeConstraint](#_48ee2586ffa14e5bb1cf8ad893969da7)

**Attributes**

**• public rmAttributeName : String [1]**

Name of attribute within the reference model that is constrained by this node.

**• public existence :** [**AttributeExistence**](#_4f99fbfcf9617d7ad55eca111d84fb67) **[0..1]**

A constraint on the existence of an attribute. An attribute with a lower multiplicity value of 0 and a positive upper value, n, can be prohibited ([0..0]), required ([1..n]) or left optional ([0..n]). The optional case exists solely as an error check, as it is not valid to restrict an existing required or prohibited attribute as optional.

**• public matchNegated : Boolean [0..1] = false**

A flag that indicates whether the match operator is to be inverted so that the constraint specifies anything except what is represented. If absent, matchNegated is

**Note:** *matchNegated* is not implemented in the current AML profile.

**Associations**

**• public attribute :** [**ComplexObjectConstraint**](#_abfab8c8e983a73b4981f6fcfdd16134)

**• public constrains :** [**Property**](#_652433ba6af347b5ab3d4b0b4b2931c9)**[1]**

The property that is constrained by the attribute constraint.

**• public parent :** [**Property**](#_652433ba6af347b5ab3d4b0b4b2931c9)**[1]**

An attribute constraint can specialize at most one AttributeConstraint in a parent defined in a parent or ancestor Archetype

The property that is constrained by the attribute constraint.

**• public object :** [**ObjectConstraint**](#_aa52f11e5760ad2f47030803962bb855)**[0..1]**

A constraint on the *type* of the UML **Property** instance constrained by an **AttributeConstraint**. *object* is optional because it is possible for **AttributeConstraints** to restrict the existence, cardinality, etc. of the **Property** itself without restricting *type* itself.

**Constraints**

* **name**

[OCL]

rmAttributeName = parentProperty.name

#### <Class> SingularAttributeConstraint

**Description**

An **AttributeConstraint** that constrains the possible values for a singular property -- a property with an upper multiplicity of 1.

**Diagrams**

[Attribute Constraint References](#_39027eaec61a2eaccc1fccb451cdda98)

**Direct Known Superclasses (Generalization)**

[AttributeConstraint](#_11f887fb6f19248bf7193bca31772c05)

**Associations**

**• public alternative :** [**ObjectConstraint**](#_aa52f11e5760ad2f47030803962bb855)**[0..\*]**

An alternative represents a 'choice'. If there are one or associated *alternative ObjectConstraints* an instance of the target of a valid *SingularAttributeConstraint* must match at least one of them. If no alternatives are specified, the instance of the *SingleAttributeConstraint* target is constrained by the inherited type of the attribute -- either from the reference model itself or from a parent archetype.

A set of one or more alternative constraints, at least one of which must be satisfied if the **SingularAttributeConstraint** is to be satisfied.

**• public excludes :** [**ObjectConstraint**](#_aa52f11e5760ad2f47030803962bb855)**[0..\*]**

The set of *types* that are *not* allowed as instances of the specified attribute. This allows the selection of various values or types that cannot appear. Types that match both *alternative* and *exclude* are treated as not allowed -- as not matching the supplied constraint.

**Note:** *exclude* is not implemented in this version of the AML Profile.

**• public constrains :** [**SingularProperty**](#_5917d2795b1a9ae4f33929e6edb8af81)**[1]**

A constraint on the type or existence of a singular property.

**Constraints**

* **existence**

[OCL]

parentProperty.lower = 1 implies existence = AttributeExistence::REQUIRED

**Known other classes**

[CollectionProperty](#_702e30cd0381b6e9726bcc6fe779a70f), [Property](#_652433ba6af347b5ab3d4b0b4b2931c9), [SingularProperty](#_5917d2795b1a9ae4f33929e6edb8af81)

#### <Enumeration> AttributeExistence

**Description**

A constraint on the cardinality of an *attribute*.

**Diagrams**

[Attribute Constraint References](#_39027eaec61a2eaccc1fccb451cdda98)

**Enumeration Literals**

* **OPTIONAL**

The lower multiplicity of the constrained attribute must be 0.

* **PROHIBITED**

The upper multiplicity of the referenced must be 0. There may be no instances of this attribute.

* **REQUIRED**

There must be at least one instance of the referenced attribute.

#### <Enumeration> CollectionType

**Description**

The uniqueness and ordering constraints on a constrained CollectionProperty.

**Diagrams**

[Attribute Constraint References](#_39027eaec61a2eaccc1fccb451cdda98)

**Enumeration Literals**

* **BAG**

A collection may contain any property in any order whether unique or not.

* **LIST**

The set of properties must be ordered.

* **SET**

Every property in the collection must be unique.

### <Package> Object Constraint References

This section describes the relationship between AML **NamedObjectConstraint, PrimitiveObjectConstraint** and the corresponding UML **Classifier** specializations. It should be noted that an AML constraint applies to *an instance of an instance of* a **Classifier**. As an example, a reference model might define an instance of a UML **Class** named "Person" with an attribute named "gender" with a type of an instance of a UML **Enumeration** called "Person Gender". A ComplexObjectConstraint named "Female Person" could be defined that restricts the instances of the Person class to those in a restriction Person Gender enumeration called "Female Gender"



**Object Constraint References**

#### <Class> EnumerationConstraint

**Description**

A constraint on a the possible values in a UML Enumeration.

**Diagrams**

[Object Constraint References](#_960b4dbbd4fb12cd1ac74b16e654b984)

**Direct Known Superclasses (Generalization)**

[NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

**Direct Known Subclasses (Specialization)**

[LocalEnumerationConstraint](#_fac1fd23e79b0fc3d709d92006b38e40)

**Attributes**

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

The set of enumeration literals that are valid instances of the referenced **Enumeration**.

The members of *possibleValues* must be instances of the *constrained* **Enumeration**.

If *possibleValues* is non-empty, only the **EnumerationLiterals** in the set are considered valid. If it is empty, all non-disallowed values are considered valid.

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

The set of enumeration literals that are valid instances of the referenced **Enumeration**. The members of *disallowValues* must be instances of the *constrained* **Enumeration**.

If both *possibleValues* and *disallowValues* are specified, the disallowed values take precedence -- if the same literal appears in *possibleValues* and *disallowValues*, it is not valid for the target type.

**• public assumedValue :** [**EnumerationLiteral**](#_41faf6a7041d7068bbbf4f9ff3924d22) **[0..1]**

The value to be assumed in the case that no value is present in an instance.

**Associations**

**• public constrains :** [**Enumeration**](#_190e24bd48f094ad9ad981ac0b4eb47e)**[1]**

The relationship to the constrained enumeration.

**Constraints**

* **pvValues**

The list of possible values must be a subset of the set of possible enumeration literals.

[OCL]

self.possibleValues->asSet()->forAll(d | self.parentClass.ownedLiteral->exists(p | p=d))

* **dvValues**

The list of disallowed values must be in the set of enumeration literals.

[OCL]

self.disallowValues->asSet()->forAll(d | self.parentClass.ownedLiteral->exists(p | p=d))

* **pORd**

An constraint may either specify possible values or disallow values but not both.

[OCL]

possibleValues->size() = 0 or disallowValues->size() = 0

#### <Class> NamedObjectConstraint

**Description**

A named constraint on the set of possible instances of a an instance of UML **Classifier**.

**Diagrams**

[Object Constraint References](#_960b4dbbd4fb12cd1ac74b16e654b984)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

**Direct Known Subclasses (Specialization)**

[ArchetypeRootProxy](#_12e58855caae51d65fb43e2837534f63), [ArchetypeSlot](#_e518b2b75b6f66417345772b8440e6f2), [ComplexObjectConstraint](#_abfab8c8e983a73b4981f6fcfdd16134), [EnumerationConstraint](#_42c2e4f902eddd2a1629a431a96cd94f)

**Attributes**

**• public redefinability :** [**Redefinability**](#_45bc3b03e253b26272fb450b2c34f5f2) **[0..1] = REDEFINABLE**

An indicator that determines whether this constraint can be further specialized.

Equivalent to **Classifie**r.*isFinalSpecialization*.

**Associations**

**• public constrains :** [**Classifier**](#_31f3bed9860f1a34043799bd12ffe873)**[1]**

A relationship to the classifier to be constrained.

#### <Class> PrimitiveObjectConstraint

**Description**

A constraint that restricts the possible values of an ADL data type. Note that the term "Primitive" is not intended to imply that the target type is *primitive* in the UML sense, but simply that it represents a leaf node that lacks any additional structure besides its identity.

Also note that, in the AML context **PrimitiveObjectConstraint** applies to a UML **DataType**, but the target object in the reference model may be represented as a UML **PrimitiveType** (e.g. **String**), a UML **DataType** such as an internal representation of a Real number or as a UML **Class**, potentially with complex structure.

The AML Profile addresses the notion of a class that maps between the AML Data Types and the corresponding Reference Model Classes.

The abstract primitive object constraint has no properties, but those of its specializations follow common patterns.

These patterns include properties defining allowed values via explicit enumeration, range definition, and matching patterns. These allowed value properties are to be treated as permissive rather than restrictive, with the result that if more than one of these properties is defined, the set of valid values for an instance is the union of values so defined. Inclusion and exclusion of interval end points in the valueRanges properties is governed by the interval data types.

Another pattern comprises the ‘assumed value’ properties. In an archetype containing optional data elements, assumed values can be used to specify values that can reliably be inferred by receivers. For example, an archetype for ‘blood pressure measurement’ might include an optional data element describing the patient position, with choices ‘lying’, ‘sitting’ and ‘standing’. Since this element is optional, conformant data could be created that does not contain it. The ‘assumed value’ property allows a value to be explicitly stated so that all users/systems know what value to assume when optional items are not included in the data, in cases where such an assumption is appropriate. In the ‘blood pressure measurement’ example, one might define such a value as ‘sitting’—the most common value—with the understanding that such an assumption makes it important for instances in which the value is different, or not known, to say so, notwithstanding that the value is technically optional.

Assumed values are definable on any primitive type. If the archetype does not define an assumed value, no reliable assumption can be made by the receiver of an archetype instance concerning the value of the property.

Date and Time constraints follow the string syntax of the ISO 8601 Representation of dates and times.

**Diagrams**

[Object Constraint References](#_960b4dbbd4fb12cd1ac74b16e654b984)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

**Direct Known Subclasses (Specialization)**

[BooleanConstraint](#_72963a51b64d10f1a0ab72b99a7d95f7), [DateConstraint](#_05406539ea2335c7e43c699988cdb385), [DateTimeConstraint](#_2e2f2a2affb8ef26918289abc068cad2), [DurationConstraint](#_ac453484c1a5116843147a38bd4020c0), [IntegerConstraint](#_05ad521706377b116680c20824646d90), [RealConstraint](#_c4e09895097a057d7e6ce4b0d10c8967), [StringConstraint](#_e78c0feb207cbea2ca9911ec94e2a83e), [TimeConstraint](#_20b17cf4d1a1f7228a809f6ed68b3a0c)

**Associations**

**• public constrains :** [**DataType**](#_d5914eb0da42172989bbe57f23fc4310)**[1]**

The data type whose possible values are constrained.

**Known other classes**

[Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [Classifier](#_31f3bed9860f1a34043799bd12ffe873), [ComplexObjectConstraint](#_abfab8c8e983a73b4981f6fcfdd16134), [DataType](#_d5914eb0da42172989bbe57f23fc4310), [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e)

### <Package> Template Metamodel

This section shows the subset of UML 2.5 Template Model that is applicable to the AML specification.

**Note:** The first version of the UML specification will not include the ability to author constraints on UML templates. This model is hsere strictly for future reference.



**Template Metamodel**

This diagram shows the aspects of UML Templates and UML template bindings that would be applicable to the AML constraint specification were it to address template and bindings. It is anticipated that the specification may be expanded at some future time to include these artifacts.

#### <Class> ClassifierTemplateParameter

**Description**

A **ClassifierTemplateParameter** exposes a **Classifier** as a formal template parameter.

**Diagrams**

[Template Metamodel](#_d132633918c4cbb0c3c69e16de3df9c2)

**Associations**

**• public ownedParameteredElement :** [**Classifier**](#_31f3bed9860f1a34043799bd12ffe873)**[1]**

The **ParameterableElement** that is owned by this **TemplateParameter** for the purpose of exposing it as the *parameteredElement*.

#### <Class> RedefinableTemplateSignature

**Description**

A **RedefinableTemplateSignature** associates an ordered list of **ClassifierTemplateParameters** with an owning **Class**. The owning **Class** typically, but not always has one or more *ownedAttributes* that reference one of the *ownedParameters* of the **RedefinableTemplateSignature**.

**Diagrams**

[Template Metamodel](#_d132633918c4cbb0c3c69e16de3df9c2)

**Associations**

**• public ownedParameter :** [**ClassifierTemplateParameter**](#_3d9b09fe9052c8305d90ab92bc37d26b)**[1..\*]**

The formal parameters that are owned by this **TemplateSignature**.

#### <Class> TemplateBinding

**Description**

A **TemplateBinding** is a **DirectedRelationship** between a **TemplateableElement** and a template. A **TemplateBinding** specifies the **TemplateParameterSubstitutions** of actual parameters for the formal parameters of the template.

**Diagrams**

[Template Metamodel](#_d132633918c4cbb0c3c69e16de3df9c2)

**Associations**

**• public parameterSubstitution :** [**TemplateParameterSubstitution**](#_c2a122fef357367888fcb3768852586c)**[1..\*]**

The **TemplateParameterSubstitutions** owned by this **TemplateBinding**.

#### <Class> TemplateParameterSubstitution

**Description**

A *RMTemplateParameterSubstitution* indicates that the *actual* *RMClass* or *RMPrimitiveType* is to be substituted as the type for the *ownedParameteredElement* owned by the *formal* *RMClassifierTemplateParameter*.

**Diagrams**

[Template Metamodel](#_d132633918c4cbb0c3c69e16de3df9c2)

**Associations**

**• public formal :** [**ClassifierTemplateParameter**](#_3d9b09fe9052c8305d90ab92bc37d26b)**[1]**

The formal **TemplateParameter** that is associated with this **TemplateParameterSubstitution**.

**• public signature :** [**RedefinableTemplateSignature**](#_6afdd25f5589999ef5ae78a4eab8563d)**[1]**

The **TemplateSignature** for the template that is the target of this **TemplateBinding**.

**• public actual :** [**Classifier**](#_31f3bed9860f1a34043799bd12ffe873)**[1]**

The **ParameterableElement** that is the actual parameter for this **TemplateParameterSubstitution**.

**Known other classes**

[Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [Classifier](#_31f3bed9860f1a34043799bd12ffe873), [Property](#_652433ba6af347b5ab3d4b0b4b2931c9)

### <Package> Instance Metamodel

Instance specifications represent one of the bigger challenges for the AML specification. The UML Reference Model consists of a set of Classes, which instances of the UML 2.5 Class, DataTypes which are instances of the UML 2.5 DataType and/or PrimitiveTypes which are instances of the UML 2.5 PrimitiveType.

The archetype constraint language requires the ability to specify assumed values for **NamedObjectConstraints**, which will take the form of an instance of the Reference Model **Class** (or an instance of an instance of a UML class. The same pattern applies to **DataTypes** and (new) **PrimitiveTypes**.

The UML 2.5 modeling language supports **Class**, **DataType**, **PrimitiveType** and **Enumeration** an the metaclass level, and the models built using the modeling language support instances of **Class** (RMClass), **DataType** (RMDataType), **PrimitiveType** (RMPrimitiveType) and **Enumeration**. In addition, the language supports a small set of instances of instances of primitive types (**String**, **Real**, **Integer**, etc.) as well as instances of instances of the UML **Enumeration** Class (**EnumerationLiterals**).

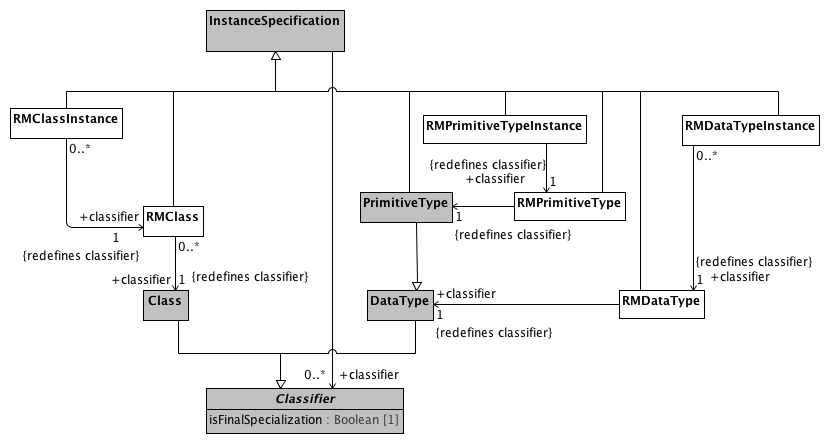
The modeling language, however, does not support a mechanism for specifying and exchanging instances of reference model classes, (new) primitive types or data types. This presents an issue when it becomes necessary to represent instances in constraints (permissible values) or assumed values.

The AML specification addresses the DataType and Primitive type issue by defining string representations for their values and constraints. It still faces the following issues:

1. How to represent instances of reference model classes as assumed values?
2. How to represent instances of the AML **ConceptReference** class, which consists of a URI, a namespace and name and other elements
3. How to represent the Designatable item, which consists of instances of a language/sign/description structure.

The AML profile has resolved these issues by:

1. Not allowing assumed values for non "primitive" types. This is a limitation on the ADL specification, but there currently aren't many, if any use cases.
2. ConceptReference is modelled as an extension to EnumerationLiteral, which allows additional instance properties.
3. Designations are represented as ordered collections of primitive types, where occurrence 0 of the language, sign, description represents the first element in the structure, occurrence 1 the second, etc.



**Instance Metamodel**

#### <Class> InstanceSpecification

**Description**

An **InstanceSpecification** represents the possible or actual existence of instances in a modeled system and completely or partially describes those instances.

**Diagrams**

[Instance Metamodel](#_4e8cc50476c42b847d941a87dd3d1d98)

**Direct Known Subclasses (Specialization)**

[PrimitiveType](#_2fd719f1d4a4fafd390224b10cd96510), [RMClass](#_977fccbc34231ad0bea8fc5ff3c8addc), [RMClassInstance](#_935cd7de4b22d47dd2c6aef93bed5c7a), [RMDataType](#_429c36b2e228f2b3422c128c3c5aa1bf), [RMDataTypeInstance](#_44fd267adf8202d91aafd96398da0a13), [RMPrimitiveType](#_32ef7946a6a95bf945efd5b41a1b81ec), [RMPrimitiveTypeInstance](#_9746082ed99de216190f4d2ea3b7a492)

**Associations**

**• public classifier :** [**Classifier**](#_31f3bed9860f1a34043799bd12ffe873)**[0..\*]**

#### <Class> PrimitiveType

**Description**

A PrimitiveType defines a predefined DataType, without any substructure

**Diagrams**

[Instance Metamodel](#_4e8cc50476c42b847d941a87dd3d1d98)

**Direct Known Superclasses (Generalization)**

[DataType](#_d5914eb0da42172989bbe57f23fc4310), [InstanceSpecification](#_14aa28a97b7753c26b32de25fe333e41)

#### <Class> RMClass

**Description**

An instance of *an instance* of an *RMClass.* As an example, if the RMClass "Automobile" were an instance of an RMClass, with the ownedAttributes "model" and "year", an RMClassInstance might be named "Ford", with the model attribute set to "Fairlane" and the year to "1965".

**Diagrams**

[Instance Metamodel](#_4e8cc50476c42b847d941a87dd3d1d98)

**Direct Known Superclasses (Generalization)**

[InstanceSpecification](#_14aa28a97b7753c26b32de25fe333e41)

**Associations**

**• public classifier :** [**Class**](#_a75c06fc93e516ccf92a1e38e18c46f3)**[1]**

#### <Class> RMClassInstance

**Diagrams**

[Instance Metamodel](#_4e8cc50476c42b847d941a87dd3d1d98)

**Direct Known Superclasses (Generalization)**

[InstanceSpecification](#_14aa28a97b7753c26b32de25fe333e41)

**Associations**

**• public classifier :** [**RMClass**](#_977fccbc34231ad0bea8fc5ff3c8addc)**[1]**

#### <Class> RMDataType

**Diagrams**

[Instance Metamodel](#_4e8cc50476c42b847d941a87dd3d1d98)

**Direct Known Superclasses (Generalization)**

[InstanceSpecification](#_14aa28a97b7753c26b32de25fe333e41)

**Associations**

**• public classifier :** [**DataType**](#_d5914eb0da42172989bbe57f23fc4310)**[1]**

#### <Class> RMDataTypeInstance

**Diagrams**

[Instance Metamodel](#_4e8cc50476c42b847d941a87dd3d1d98)

**Direct Known Superclasses (Generalization)**

[InstanceSpecification](#_14aa28a97b7753c26b32de25fe333e41)

**Associations**

**• public classifier :** [**RMDataType**](#_429c36b2e228f2b3422c128c3c5aa1bf)**[1]**

#### <Class> RMPrimitiveType

**Diagrams**

[Instance Metamodel](#_4e8cc50476c42b847d941a87dd3d1d98)

**Direct Known Superclasses (Generalization)**

[InstanceSpecification](#_14aa28a97b7753c26b32de25fe333e41)

**Associations**

**• public :** [**PrimitiveType**](#_2fd719f1d4a4fafd390224b10cd96510)**[1]**

#### <Class> RMPrimitiveTypeInstance

**Diagrams**

[Instance Metamodel](#_4e8cc50476c42b847d941a87dd3d1d98)

**Direct Known Superclasses (Generalization)**

[InstanceSpecification](#_14aa28a97b7753c26b32de25fe333e41)

**Associations**

**• public classifier :** [**RMPrimitiveType**](#_32ef7946a6a95bf945efd5b41a1b81ec)**[1]**

**Known other classes**

[Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [Classifier](#_31f3bed9860f1a34043799bd12ffe873), [DataType](#_d5914eb0da42172989bbe57f23fc4310)

### <Package> Package Metamodel

This section describes the Namespace and Package aspects of the UML 2.5 model that are used in the AML Object Model Specification. Note that, while this model is faithful to the actual UML metamodel, it has been flattened for the purposes of simplification, showing only the classes and attributes that are relevant to the AML Object Model Specification.



**Package Metamodel**

This diagram shows the subset of the UML 2.5 Package specification that is used in the AML Object Model. In particular, this diagram is meant to call attention to the fact that every Class is a NamedElement and, at such, has a memberNamespace which, for our purposes is a UML Package.

AML requires that all archetypes within a given archetype library must directly or indirectly constrain an class that is a member of the Package referenced by the containing library.

**Known other classes**

[Class](#_a75c06fc93e516ccf92a1e38e18c46f3), [Classifier](#_31f3bed9860f1a34043799bd12ffe873), [NamedElement](#_527fd9eb1e787c36a3748854a9431816), [Namespace](#_f762e4ef59f1948849a49d421126c16b), [Package](#_a0a843d7d41881592e31e887cebd6da4)

### <Package> Enumeration Metamodel

ADL 2.0 provides the ability to constrain UML Enumeration types. Section 6.2.3.5 of the AOM specification describes how the AOM model represents constraints as instances of C\_INTEGER or C\_STRING to document the value assigned to an enumeration literal in a transformation to a programming language. The AML specification addresses the need to assign specific representations by noting that a given **Enumeration** type can have attributes, allowing the **EnumerationLiteral** instances to include both names and numeric or string values.

The diagram below shows the UML **Enumeration** and **EnumerationLiteral** data types and the AML extensions that blend them with the corresponding ISO 11179 Enumerated\_Value\_Domain and Permissible\_Value types.



**Enumeration Metamodel**

This diagram shows the relationship between the UML Enumeration and EnumerationLiteral data types and the EnumeratedValueDomain and PermissibleValue extensions.

#### <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.

EnumeratedValueDomain extends both the UML Enumeration class and the ISO 11179 Enumerated\_Value\_Domain.

**Diagrams**

[Enumeration Metamodel](#_c323459faa5aa97d8abc3c64dcd86661)

**Direct Known Superclasses (Generalization)**

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

**Associations**

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

The set of valid or "permissible" values for a given domain.

**• public valueDomainMeaning :** [**ValueSetAndDefinitionReference**](#_25ca98283400275c47128a3c11af1824)**[0..1]**

A reference to a value set and optional definition that, when interpreted, produces a set of value meaning references.

#### <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**

[Enumeration Metamodel](#_c323459faa5aa97d8abc3c64dcd86661)

**Direct Known Superclasses (Generalization)**

[Value\_Domain](#_9b81960c26ee442be19393f5a5fc3ef2)

**Direct Known Subclasses (Specialization)**

[EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51)

**Associations**

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

A value in the domain.

#### <Class> EnumerationLiteral

**Description**

An **EnumerationLiteral** defines an element of the run-time extension of an **Enumeration**. Values corresponding to **EnumerationLiterals** are immutable and may be compared for equality. **EnumerationLiterals** may not change during their existence, so any attributes on an **Enumeration** shall be read-only.

An **EnumerationLiteral** has a name that shall be used to identify it within its **Enumeration**. The **EnumerationLiteral** name is scoped within and shall be unique within its **Enumeration**. **EnumerationLiteral** names shall be qualified for general use.

**Diagrams**

[Enumeration Metamodel](#_c323459faa5aa97d8abc3c64dcd86661)

**Direct Known Superclasses (Generalization)**

[NamedElement](#_527fd9eb1e787c36a3748854a9431816)

**Direct Known Subclasses (Specialization)**

[PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)

**Associations**

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

The **Enumeration** that this **EnumerationLiteral** is a member of.

**• public classifier :** [**Enumeration**](#_190e24bd48f094ad9ad981ac0b4eb47e)

The classifier of this **EnumerationLiteral** derived to be equal to its **Enumeration**.

#### <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**

[Enumeration Metamodel](#_c323459faa5aa97d8abc3c64dcd86661)

**Direct Known Superclasses (Generalization)**

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

**Associations**

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

The scoping domain for the permissible value.

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

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

A reference to the meaning associated with the **PermissibleValue.**

**Known other classes**

[DataType](#_d5914eb0da42172989bbe57f23fc4310), [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e)

## <Package> Constraint Meta Model

The constraint model is the core of the archetype design. It specifies how constraints are defined, showing the object-attribute-object pattern characteristic of object constraints. Because objects are composed of properties, and properties consist of objects, the archetype definition consists of alternate layers of objects and attributes.

### <Package> Atomic Data Type Constraints

This section describes the atomic or "primitive type constraints" -- constraints that restrict the possible values of the built in AML data types and/or specify assumed values when the element is not included in a data instance.



**Atomic Data Type Constraints**

Primitive object constraints define patterns that archetypes use to constrain the values of primitive or atomic types. All complex constraints are composed of primitive constraints at the leaf level.

#### <Class> BooleanConstraint

**Description**

A constraint on a BooleanPrimitive type.

**Diagrams**

[Atomic Data Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

**• public possibleValues :** [**BooleanPrimitive**](#_19737cdaaaee2179c50553b52361808b) **[0..\*]**

A set of allowed BooleanPrimitive values (e.g., True, False).

**• public assumedValue :** [**BooleanPrimitive**](#_19737cdaaaee2179c50553b52361808b) **[0..1]**

A Boolean value to be assumed to apply if no value is provided.

#### <Class> DateConstraint

**Description**

A constraint on a property of DatePrimitive type.

**Diagrams**

[Atomic Data Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

**• public possibleValues :** [**DatePrimitive**](#_db5d020506d0af7a330f0b4fe1cb870a) **[0..\*]**

A set of allowed Date values (e.g., ‘2000-01-01’).

**• public valueRanges :** [**DateInterval**](#_69ee53d558bd7c80a52df2f5ae7c8ab0) **[0..\*]**

A set of Date value ranges, any value of which is considered valid (e.g., ‘|>= 2000-01-01|’, ’2000-01-01..2005-06-30’).

**• public matchPattern :** [**DateMatchPattern**](#_3cae9b111208dfafe95b548c45720f8e) **[0..1]**

A string pattern that implies a set of valid Date values (e.g., ‘2000-??-xx’, ‘yyyy-xx-xx’).

**• public assumedValue :** [**DatePrimitive**](#_db5d020506d0af7a330f0b4fe1cb870a) **[0..1]**

A DatePrimitive value to be assumed to apply if no value is provided.

#### <Class> DateTimeConstraint

**Description**

A constraint on a property of DateTimePrimitive type.

**Diagrams**

[Atomic Data Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

**• public possibleValues :** [**DateTimePrimitive**](#_8347879bb381db17040637cc3ba0a25c) **[0..\*]**

A set of allowed DateTime values.

**• public valueRanges :** [**DateTimeInterval**](#_3145967a9ac97f161c2d4ff9e296537e) **[0..\*]**

A set of DateTime value ranges, any value of which is considered valid (e.g., ‘|>=2000-01-01T 09:30:00|’, ’ 2000-01-01T09:30:00.. 2000-01-01T11:30:00’).

**• public matchPattern :** [**DateTimeMatchPattern**](#_b501b0aa7bbcd9ac89d6ddbf9f0bf3d9) **[0..1]**

A string pattern that implies a set of valid DateTime values (e.g., ‘yyyy-mm-ddT13:??:xx’, ‘yyyy-mm-ddThh:xx:xx’)

**• public assumedValue :** [**DateTimePrimitive**](#_8347879bb381db17040637cc3ba0a25c) **[0..1]**

A DateTime value to be assumed to apply if no value is provided.

#### <Class> DurationConstraint

**Description**

A constraint on a property of Duration type.

**Diagrams**

[Atomic Data Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

**• public possibleValues :** [**DurationPrimitive**](#_64b0026498682fb721ccdb38c186bba2) **[0..\*]**

A set of allowed Duration values (e.g., ‘P5d’).

**• public valueRanges :** [**DurationInterval**](#_8370a7823f617661d9ddb083556dfa3f) **[0..\*]**

A set of Duration value ranges, any value of which is considered valid (e.g., ‘|>= P5d|’, ’P5d..P8d’).

**• public matchPattern :** [**DurationMatchPattern**](#_baaa358403d3312c0779bc256e1050bd) **[0..1]**

A string pattern that implies a set of valid Duration values (e.g., ‘Pd’, ‘PThm’).

**• public assumedValue :** [**DurationPrimitive**](#_64b0026498682fb721ccdb38c186bba2) **[0..1]**

A DurationPrimitive value to be assumed to apply if no value is provided.

#### <Class> IntegerConstraint

**Description**

A constraint on a property of IntegerPrimitive type.

**Diagrams**

[Atomic Data Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

**• public possibleValues :** [**IntegerPrimitive**](#_89e2b2b9de405e6d05c4c5259fc8ffd6) **[0..\*]**

A set of allowed Integer values (e.g., 2, 5).

**• valueRanges :** [**IntegerInterval**](#_5218d9b9782bc6f925ed66b3e2bc529b) **[0..\*]**

A set of Integer value ranges, any value of which is considered valid.

**• public assumedValue :** [**IntegerPrimitive**](#_89e2b2b9de405e6d05c4c5259fc8ffd6) **[0..1]**

An IntegerPrimitive value to be assumed to apply if no value is provided.

#### <Class> RealConstraint

**Description**

A constraint on a property of RealPrimitive type.

**Diagrams**

[Atomic Data Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

**• public possibleValues :** [**RealPrimitive**](#_c596f10fb93cb4f697f2f1b0b64b43ce) **[0..\*]**

A set of allowed Real values (e.g., 37.2, 100.265).

**• public valueRanges :** [**RealInterval**](#_27fcdc4a73f595dca8b4e4fa28fd59a6) **[0..\*]**

A set of Real value ranges, any value of which is considered valid.

**• public assumedValue :** [**RealPrimitive**](#_c596f10fb93cb4f697f2f1b0b64b43ce) **[0..1]**

A RealPrimitive value to be assumed to apply if no value is provided.

#### <Class> StringConstraint

**Description**

A constraint on a property of String type.

Most text values that can be constrained are best constrained by language-independent terminology constraints, but this tactic is supported.

**Diagrams**

[Atomic Data Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

**• public possibleValues :** [**StringPrimitive**](#_6a90be7cfa784ea4b4e8cad8f4a47e82) **[0..\*]**

A set of allowed String values (e.g., ‘foo’).

**• public matchPattern :** [**RegularExpression**](#_5618f0ea3d5c7b6941d4be66259e0d28) **[0..1]**

A Perl regular expression defining allowed string constructions.

**• public assumedValue :** [**StringPrimitive**](#_6a90be7cfa784ea4b4e8cad8f4a47e82) **[0..1]**

A StringPrimitive value to be assumed to apply if no value is provided.

#### <Class> TimeConstraint

**Description**

A constraint on a property of TimePrimitive type.

**Diagrams**

[Atomic Data Type Constraints](#_2ca64b6794a95c8188c5478872196c54)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

**Attributes**

**• public possibleValues :** [**TimePrimitive**](#_05e0f2a221f0733d693058e9253b0017) **[0..\*]**

A set of allowed TimePrimitive values (e.g., ‘09:30:00’).

**• public valueRanges :** [**TimeInterval**](#_31d7d28b74d99d489c8e65e4a57e9ca9) **[0..\*]**

A set of Time value ranges, any value of which is considered valid (e.g., ‘|>= 09:30:00|’, ’09:30:00..11:30:00’).

**• public matchPattern :** [**TimeMatchPattern**](#_a28995c5bf2253d86854a7a97e55ba1d) **[0..1]**

A string pattern that implies a set of valid Time values (e.g., ‘13:??:xx’, ‘hh:xx:xx’).

**• public assumedValue :** [**TimePrimitive**](#_05e0f2a221f0733d693058e9253b0017) **[0..1]**

A TimePrimitive value to be assumed to apply if no value is provided.

**Known other classes**

[PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

### <Package> Terminology Constraints

This section describes how instances of the **TerminologyCode** data type are constrained via a **TerminologyCodeConstraint**. The AML TerminologyCode acts as a bridge between the Common Terminology Services 2 (CTS2), the ISO 11179-3 designation, definition and identification sections and the

requirements imposed by various reference model targets, including ISO 21090 Coded Data (CD) types and its derivatives, FHIR Coding data type, openEHR DV\_CODED\_TEXT, CIMI CODED\_TEXT etc. The **TerminologyCode** target allows a given instance to include, at a bare minimum, an identifier of the scoping namespace (*terminologyId*) and the unique code within that namespace(*codeString*). While this information may or may not be sufficient to transform to an external target, it supports the minimum requirement that AML allow terminology codes to be restricted by simple, internal enumerations as represented by the "ac" ADL codes and the corresponding "at" code value sets.

Constraints are expressed as sets of **PermissibleValues** -- either directly via the *possibleValues* attribute or indirectly through the *valueSetConstraint* attribute which, possibly with the assistance of a terminology service, also results in a collection of **PermissibleValues**.

A PermissibleValue can represent:

* a simple **EnumerationLiteral**
* an **PermissibleValue** with one or more *identifiers* of type **Scoped\_Identifier**
* an **PermissibleValue** with an associated *valueMeaning* of type **ConceptReference**

Each of these situations has a different analog in terms of the target TerminologyCode.

**Simple EnumerationLiteral**

* The *name* of the containing **Enumeration** maps to *terminologyId*
* The *name* of the **EnumerationLiteral** maps to *codeString*

The uri attribute may be populated via the namespace map.

Note that the ADL specification may require that enumeration values may need to be represented as integers and strings. This may be accomplished by adding an inteter or string to the scoping **Enumeration** type and then adding a corresponding slot to each **EnumerationLiteral** instance. The mechanism and mapping for this approach are outside of the scope of the AML specification.

**PermissibleValue with an *identifier***

If the **PermissibleValue** has an associated *identifier*, the *identifier* attribute of the target *identifier* maps to codeString and, if present, the shorthand\_prefix of the identifier *scope* maps to *terminologyId*. If *shorthand\_prefix* is absent, then the *name* of the *scope* **Namespace** is used. If more than one *identifier* is present, an implementation may choose which identifier will be mapped.

The *uri* attribute may be populated via the namespace map.

**PermissibleValue with a *valueMeaning***

The *valueMeaning* attribute is of type **ConceptReference** and carries the equivalent of a "term binding" for an ADL "at" code. In this case, the target **ConceptReference** is mapped to the **TerminologyCode** as:

* The *conceptReference* *uri* maps to the *TerminologyCode* *uri*
* If the *conceptReference* has a *name* attribute (type **ScopedEntityName**), the *namespace* maps to the *terminologyId* attribute and *name* to *codeString*
* If the *conceptReference* does not have a *name* attribute, the identifier rules above should be employed or, lacking an identifier, the simple **EnumerationLiteral** rules.
* If the *conceptReference* has a *describingCodeSystemAndVersion* attribute thenit maps to the corresponding *terminologyVersion*. If the *conceptReference* does not have a *describingCodeSystemAndVersion* attribute, then the *terminologyVersion* attribute should be left empty.



**TerminologyConstraints**

#### <Class> ObjectConstraint

**Description**

**ObjectConstraint** is the base class of all AML classifier constraints.

**Diagrams**

[TerminologyConstraints](#_2c065eee9fec4768da07422243566f39)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_4cfed31aa26e11dff4000228bcd222ae)

**Direct Known Subclasses (Specialization)**

[EnumerationConstraint](#_42c2e4f902eddd2a1629a431a96cd94f), [NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9), [ObjectConstraintProxy](#_6da4a9bc7db41a2b89064f79f0c4ed36), [PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da), [TerminologyCodeConstraint](#_642a5d8e79d72684f85a1ef76261f3ad)

**Associations**

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

All **ObjectConstraints** must ultimately be applied to an instance of a UML Classifier. **ObjectConstraints** can add additional restrictions to an existing "parent" constraint. If present, *parent* represents the existing constraint that is being further specialized.

A reference to the meaning associated with the **PermissibleValue.**

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

#### <Class> TerminologyCode

**Description**

A reference to a class, category or "concept", optionally as described in a specified code system or code system version. **TerminologyCode** represents the set of information needed to populate various target reference model coded elements, including a code, the scoping namespace, the corresponding URI, the code system or code system version from which the description was derived and a informative "term" which communicates the intent of the code to a human reader.

Two TerminologyCodes are considered to be identical if and only if:

1. They both have *uri* attributes and the uris are identical
2. One or both do not have a uri attribute and the *terminologyId* and *codeStrings* are identical
3. Both *codeStrings* are identical and the *terminologyId's* map to the same terminology *uri*.

The *terminologyVersion* and *term* attributes are strictly informative and are not part of the identity of a terminology code.

**Diagrams**

[TerminologyConstraints](#_2c065eee9fec4768da07422243566f39)

**Attributes**

**• public terminologyId : String [1]**

The name of the namespace that scopes *codeString*.

**• public codeString : String [1]**

A code that uniquely references a class, category or "concept" within the context of the scoping *terminologyId*.

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

The code system or code system or code system version that contained a description of the terminology code at the point in time it was referenced.

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

A URI that uniquely names the terminology code.

**• public term : String [0..1]**

An informative string that is intended to communicate the referent of the terminology code to a human reader. Term may be set to any appropriate language and context specific signifier that is deemed appropriate in a particular implementation. It is not part of the identity of the terminology code itself and should not be used as an identifier.

**Constraints**

* **codeOrURI**

A **TerminologyCode** instance must specify a *codeString* or a *uri*.

[OCL]

not(self.uri.oclIsUndefined()) or not(self.terminologyId.oclIsUndefined() or self.codeString.oclIsUndefined())

#### <Class> TerminologyCodeConstraint

**Description**

**TerminologyCodeConstraint** defines an list of possible PermissibleValues that restrict the set of corresponding TerminologyCode instances.

**Diagrams**

[TerminologyConstraints](#_2c065eee9fec4768da07422243566f39)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

**Attributes**

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

A list of **PermissibleValues** that may be represented as **TerminologyCodes**. If empty, there is no constraint.

**• public valueSetConstraint :** [**EnumeratedValueDomain**](#_ad639ee3d4cd535b2d3e55238d69cc51) **[0..1]**

An **EnumeratedValueDomain**, the members of which restrict the possible **TerminologyCode** values. If absent, there is no constraint.

If both possibleValues and a valueSetConstraint is specified, TerminologyCode instances are restricted to the intersection of the two fields.

**• public assumedValue :** [**PermissibleValue**](#_66976d5fcaf3eff9df49b6e5dab4ad12) **[0..1]**

The PermissibleValues, whose TerminologyCode interpretation is assumed to be the value for the containing attribute if absent in a data instance.

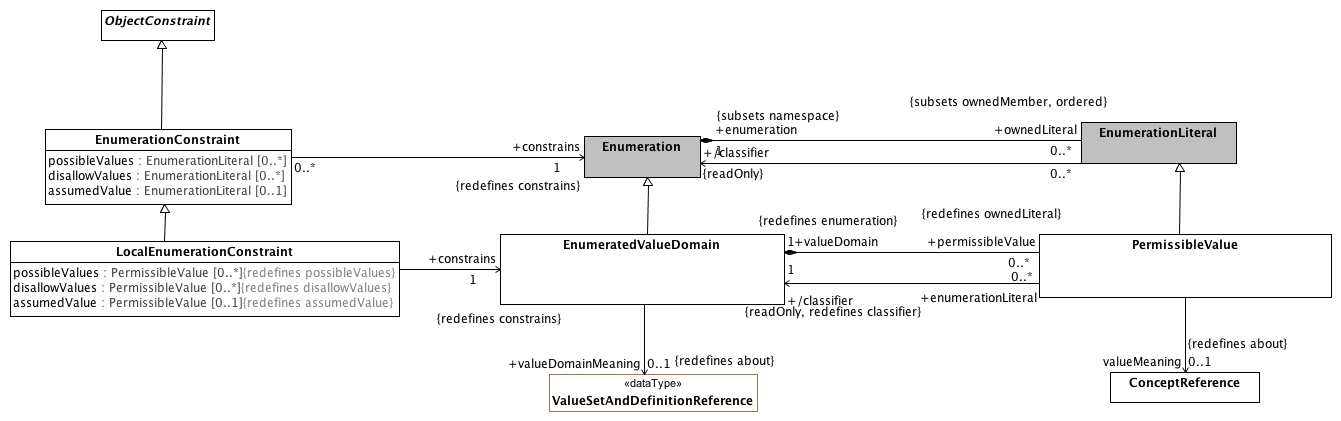
**Associations**

**• public constrains :** [**TerminologyCode**](#_3fde1641e9d2557d85d9c7faa67a58f7)**[1]**

The **TerminologyCode** constrained by the **TerminologyCodeConstraint**.

### <Package> Enumeration Constraints

This section describes the AML artifacts that constrain UML Enumerations and the ISO 11179-3 extension, EnumeratedValueDomain.



**EnumerationConstraints**

#### <DataType> ValueSetAndDefinitionReference

**Description**

A reference to a value set and optional definition.

**Diagrams**

[EnumerationConstraints](#_9a0e8e1c8e8cdbf68da35592986a39eb)

**Direct Known Superclasses (Generalization)**

[ValueSetReference](#_5e70a8ee4206e2396a68ca679af92933)

**Attributes**

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

#### <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**

[EnumerationConstraints](#_9a0e8e1c8e8cdbf68da35592986a39eb)

**Direct Known Superclasses (Generalization)**

[URIAndEntityName](#_4d1f571ab5e9384786ffe39444e822b4)

**Associations**

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

The code system and optional version that carried the description of the ConceptReference that was used to determine its meaning/validity.

#### <Class> LocalEnumerationConstraint

**Description**

A **LocalEnumerationConstraint** is a constraint on an EnumeratedValueDomain -- an extension of the UML enumeration that supports designation, definition, identification and meaning references.

**Diagrams**

[EnumerationConstraints](#_9a0e8e1c8e8cdbf68da35592986a39eb)

**Direct Known Superclasses (Generalization)**

[EnumerationConstraint](#_42c2e4f902eddd2a1629a431a96cd94f)

**Attributes**

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

The set of enumeration literals that are valid instances of the referenced **EnumeratedValueDomain**.

The members of *possibleValues* must be instances of the *constrained* **EnumeratedValueDomain**.

If *possibleValues* is non-empty, only the **PermissibleValues** in the set are considered valid. If it is empty, all non-disallowed values are considered valid.

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

The set of **PermissableValues** that are valid instances of the referenced **EnumeratedValueDomain**. The members of *disallowValues* must be instances of the constrains **EnumeratedValueDomain**.

If both *possibleValues* and *disallowValues* are specified, the disallowed values take precedence -- if the same literal appears in *possibleValues* and *disallowValues*, it is not valid for the target type.

**• public assumedValue :** [**PermissibleValue**](#_66976d5fcaf3eff9df49b6e5dab4ad12) **[0..1]**

The value to be assumed in the case that no instance is present in a message or database.

**Associations**

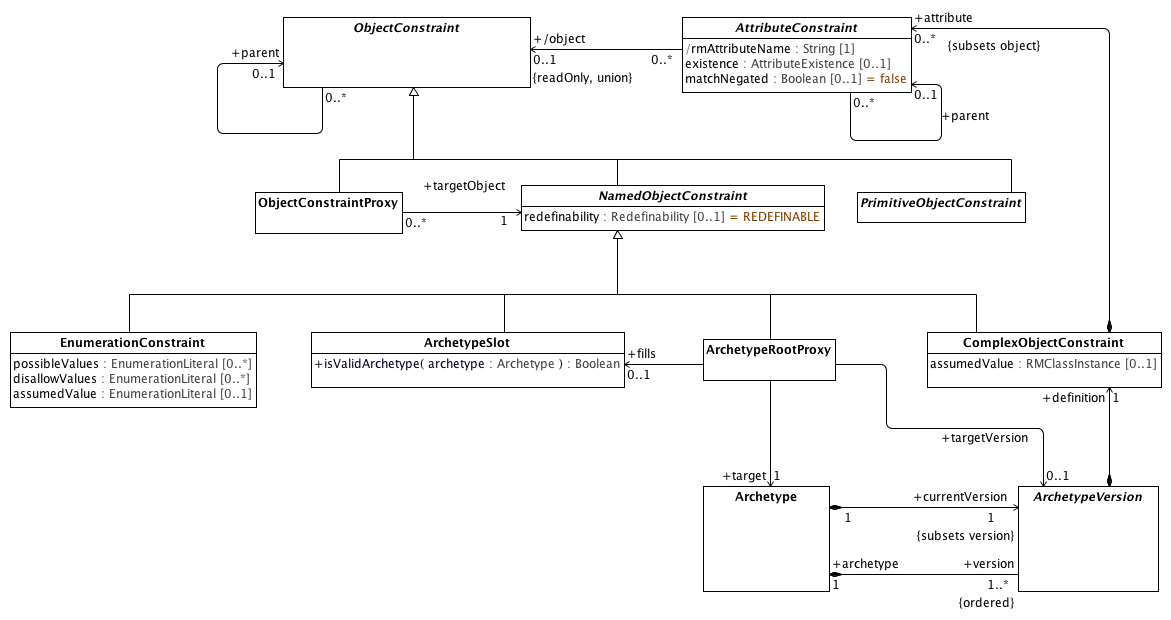
**• public constrains :** [**EnumeratedValueDomain**](#_ad639ee3d4cd535b2d3e55238d69cc51)**[1]**

A constraint on the set of possible **PermissableValues**.

**Known other classes**

[EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51), [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e), [EnumerationConstraint](#_42c2e4f902eddd2a1629a431a96cd94f), [EnumerationLiteral](#_41faf6a7041d7068bbbf4f9ff3924d22), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855), [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)

### <Package> Object Constraints



**Object Constraints**

#### <Class> ArchetypeRootProxy

**Description**

A reference to a target **Archetype** that constrains the same subclass or descendant thereof as that referenced by the *constrains* attribute of the **NamedObjectConstraint** itself.

ArchetypeRootProxy implements the *use\_archetype* construct in ADL. Note that an **ArchetypeRootProxy** may or may not reference a specific version of the *target* **Archetype**. If the *targetVersion* attribute is absent, the **ArchetypeRootProxy** is understood to reference the ArchetypeVersion that is considered to be "current" at whatever time that the proxy is referenced.

**Diagrams**

[Object Constraints](#_778c20eb327fafc2160b6a98a6014e5d)

**Direct Known Superclasses (Generalization)**

[NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9)

**Associations**

**• public target :** [**Archetype**](#_f45a7b68ecac449e953ff8a65d6eff75)**[1]**

The **Archetype** whose constraints are to be applied to the *constrained* **Classifier**.

**• public fills :** [**ArchetypeSlot**](#_e518b2b75b6f66417345772b8440e6f2)**[0..1]**

An *ArchetypeRootProxy* can fill an *ArchetypeSlot* that occurs in a *parent* or ancestor of the *Archetype specialize* association.

A reference to an **Archetype** that *constrains* the same **Class** *constrained* by the *fills* **ArchetypeSlot**.

**• public targetVersion :** [**ArchetypeVersion**](#_1de96fa71501cf96b27b14f3f9f1bb99)**[0..1]**

The particular version of the *target* **Archetype** to be applied as a constraint. If absent, the constraints in the *currentVersion* of the *target* **Archetype** is to be applied.

**Constraints**

* **validSlot**

[Analysis]

If there is a target fills ArchetypeSlot, the ArchetypeRootProxy archetypeId must be a member of one or more includes value sets and not be a member of one or more excludes value sets as identified in the ArchetypeSlot or any of its parents or ancestors

* **validTarget**

[OCL]

target.archetypeId = archetypeId and target.archetypeName = archetypeName -- Also

#### <Class> ArchetypeSlot

**Description**

An **ArchetypeSlot** represents a set of rules that can be used to determine whether a given **ArchetypeRootProxy** can validly *constrain* the **Class** referenced by the *constrains* by the referencing slot. **ArchetypeSlot** implements the *allow\_archetype* and *exclude\_archetype* constructs in ADL.

The actual rules embodied by ArchetypeSlot are implementation specific -- implementation would be expected to specialize this class.

**Diagrams**

[Object Constraints](#_778c20eb327fafc2160b6a98a6014e5d)

**Direct Known Superclasses (Generalization)**

[NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9)

**Operations**

**• public isValidArchetype (archetype :** [**Archetype**](#_f45a7b68ecac449e953ff8a65d6eff75)**) : Boolean**

Returns **true** if *archetype* is allowed to further restrict the UML **Class** and, if present constraint *parent* referenced by the *constrains* attribute of this slot.

#### <Class> ObjectConstraintProxy

**Description**

An **ObjectConstraintProxy** asserts that a constraint defined elsewhere in the archetype is to apply to the Classifier instance referenced by the proxy *constrains* attribute.

**Diagrams**

[Object Constraints](#_778c20eb327fafc2160b6a98a6014e5d)

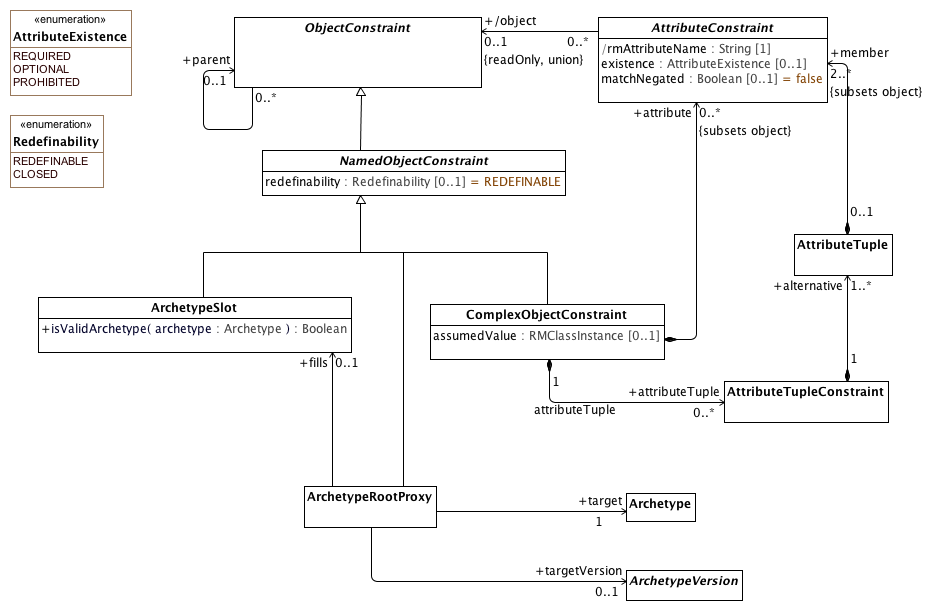
**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

**Known other classes**

[Archetype](#_f45a7b68ecac449e953ff8a65d6eff75), [ArchetypeVersion](#_1de96fa71501cf96b27b14f3f9f1bb99), [AttributeConstraint](#_11f887fb6f19248bf7193bca31772c05), [ComplexObjectConstraint](#_abfab8c8e983a73b4981f6fcfdd16134), [EnumerationConstraint](#_42c2e4f902eddd2a1629a431a96cd94f), [NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855), [PrimitiveObjectConstraint](#_db9df3b10e304d809393da4afc9a91da)

#### <Package> Named Object Constraints



**Named Object Constraints**

##### <Class> AttributeTuple

**Description**

A set of constraints on related attributes to be used to differentiate scenarios where the value of one attribute affects the valid values of another

**Diagrams**

[Named Object Constraints](#_9247c31608028a4b9ce5cbb4664b4baa)

**Associations**

**• public member :** [**AttributeConstraint**](#_11f887fb6f19248bf7193bca31772c05)**[2..\*]**

An *attribute* in the referenced **Class** or **DataType** constrained by an **AttributeTuple**

##### <Class> AttributeTupleConstraint

**Description**

An **AttributeTupleConstraint** presents a set of two or more alternative tuples, each of which simultaneously restricts of two or more attributes. The containing **ComplexObjectConstraint** is satisfied when all of the constraints in one of the *alternative* **AttributeTuples** are satisfied.

**Diagrams**

[Named Object Constraints](#_9247c31608028a4b9ce5cbb4664b4baa)

**Associations**

**• public alternative :** [**AttributeTuple**](#_6d5bfb351e19f61e0327587b0ff5fd4f)**[1..\*]**

A set of one or more alternative constraints, at least one of which must be satisfied if the **AttributeTupleConstraint** is to be satisfied.

**Known other classes**

[Archetype](#_f45a7b68ecac449e953ff8a65d6eff75), [ArchetypeRootProxy](#_12e58855caae51d65fb43e2837534f63), [ArchetypeSlot](#_e518b2b75b6f66417345772b8440e6f2), [ArchetypeVersion](#_1de96fa71501cf96b27b14f3f9f1bb99), [AttributeConstraint](#_11f887fb6f19248bf7193bca31772c05), [ComplexObjectConstraint](#_abfab8c8e983a73b4981f6fcfdd16134), [NamedObjectConstraint](#_ab5b3b01964560abb1047edd9efa4eb9), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

##### <Enumeration> Redefinability

**Description**

Determines whether a NamedObjectConstraint can be further specialized.

**Diagrams**

[Named Object Constraints](#_9247c31608028a4b9ce5cbb4664b4baa)

**Enumeration Literals**

* **CLOSED**

The **NamedObjectConstraint** cannot be a *parent* of another **ObjectConstraint**.

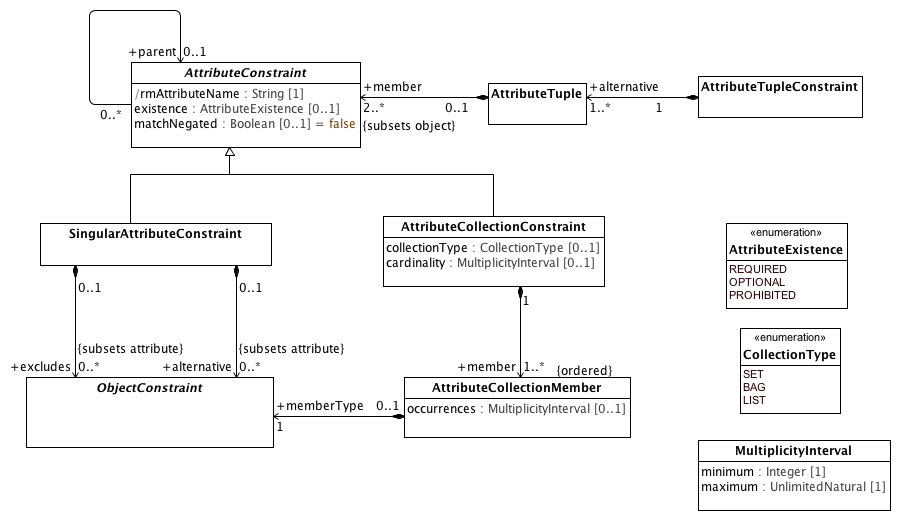
* **REDEFINABLE**

The **NamedObjectConstraint** can be the *parent* of another ObjectConstraint.

**Known other enumerations**

[AttributeExistence](#_4f99fbfcf9617d7ad55eca111d84fb67)

### <Package> Attribute Constraints



**Attribute Constraints**

#### <Class> AttributeCollectionMember

**Description**

An association that matches members of a collection of attributes with specific ObjectConstraints

**Diagrams**

[Attribute Constraints](#_a74fdcc000d4318dcb5580a5fa8fbfce)

**Attributes**

**• public occurrences :** [**MultiplicityInterval**](#_c810ec7fa381fa249b7a7d9fecae85b6) **[0..1]**

The minimum and maximum times that the members of this subset may appear. If absent, it may appear any number of times as long as it doesn't violate the *cardinality* constraint of the **AttributeCollectionConstraint** of which it is a *member*.

**Associations**

**• public memberType :** [**ObjectConstraint**](#_aa52f11e5760ad2f47030803962bb855)**[1]**

An **ObjectConstraint** that restricts the possible values of the particular **AttributeCollectionMember**. Note that the UML **Classifier** constrained by the *memberType* **ObjectConstraint** may reference the same **Type** as the *constrained* attribute or a (UML) specialization thereof.

#### <Class> MultiplicityInterval

**Description**

The minimum and maximum number of times that an element may occur.

**Diagrams**

[Attribute Constraints](#_a74fdcc000d4318dcb5580a5fa8fbfce)

**Attributes**

**• public minimum : Integer [1]**

The least number of elements.

**• public maximum : UnlimitedNatural [1]**

The greatest number of elements if maximum is an integer.

**Known other classes**

[AttributeCollectionConstraint](#_5eefba8eca7402f09bd5619804038771), [AttributeConstraint](#_11f887fb6f19248bf7193bca31772c05), [AttributeTuple](#_6d5bfb351e19f61e0327587b0ff5fd4f), [AttributeTupleConstraint](#_f6da15c71717330ae1b56f8b41e3dd51), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855), [SingularAttributeConstraint](#_48ee2586ffa14e5bb1cf8ad893969da7)

**Known other enumerations**

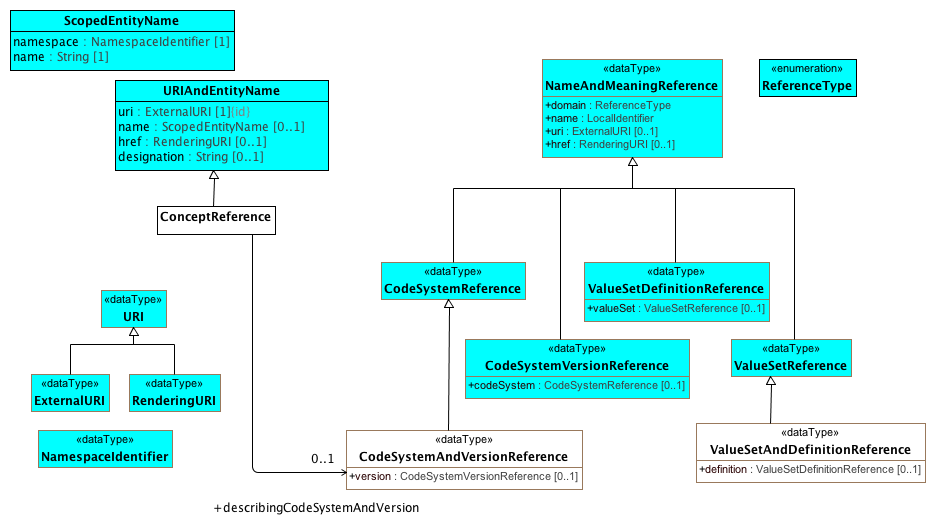
[AttributeExistence](#_4f99fbfcf9617d7ad55eca111d84fb67), [CollectionType](#_ac8c7771bf7f68d6747022e6924749ca)

## <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)

**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> 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)

**Known other Data Types**

[ValueSetAndDefinitionReference](#_25ca98283400275c47128a3c11af1824)

#### <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 (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 *URI* that identifies the intended meaning of the identifier.

**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.

[OCL]

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

**Known other classes**

[ConceptReference](#_4082bf99060eea349dab8f548d297f3d)

#### <Enumeration> ReferenceType

**Description**

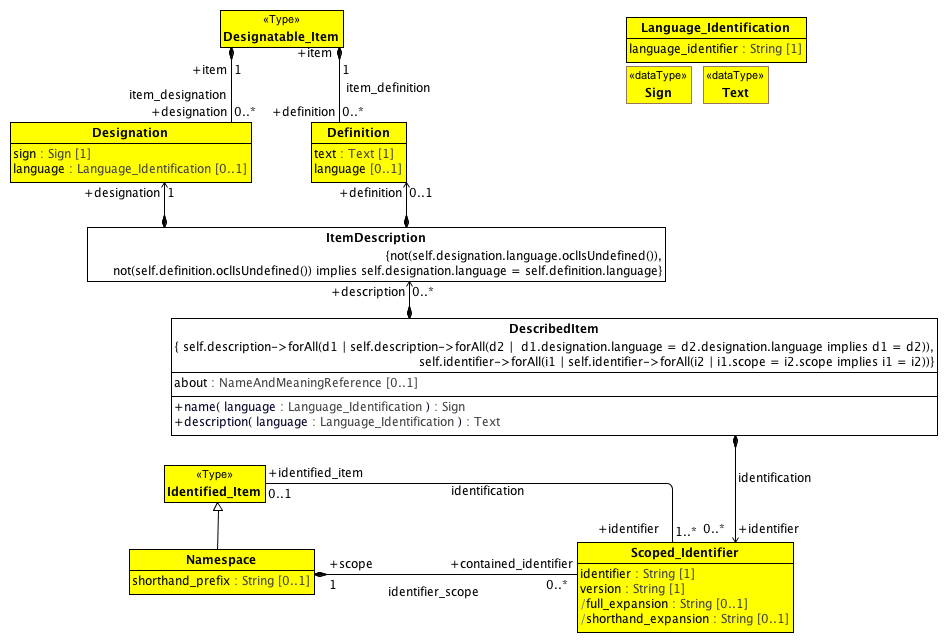
**Diagrams**

[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]**

The text of the Definition.

**• public language [0..1]**

language (3.2.68) used to write the definition text.

**Associations**

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

##### <Class> DescribedItem

**Description**

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

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**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Direct Known Subclasses (Specialization)**

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

**Attributes**

**• public about :** [**NameAndMeaningReference**](#_7677b94abdbbdc2c76d49314388386a7) **[0..1]**

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

**Operations**

**• public name (language :** [**Language\_Identification**](#_290b504024c5117158636715bff68ef8)**) :** [**Sign**](#_3e3bff45f2bcef12ecdff2975e0f26aa)

Returns the name of the artifact in the supplied language.

**• public description (language :** [**Language\_Identification**](#_290b504024c5117158636715bff68ef8)**) :** [**Text**](#_b7253cf7959d2481cd50a89513cd395b)

Returns the description/definition of the artifact in the supplied language

**Associations**

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

A unique identifier for the item. This attribute supports the ADL "id", "at" and "ac" identifiers, where the *scope* Namespace can be on the Archetype, Archetype\_Version or referencing component level.

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

A designation and optional description in a specified language.

**Constraints**

* **sameLanguage**

[OCL]

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

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

[OCL]

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

* **oneIdPerNamespace**

[OCL]

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

* **oneDescriptionPerLanguage**

[OCL]

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..\*]**

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

##### <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]**

##### <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..\*]**

An identifier for the Identified\_Item within a specific Namespace.

##### <Class> ItemDescription

**Diagrams**

[Designation and Definition metamodel region](#_39265938304977d9288c6eb410872362)

**Associations**

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

An optional definition in a specified language

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

A designation in a specified language

**Constraints**

* **mandatoryLanguage**

All designations must have a language

[OCL]

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

* **matchingDescriptionLanguage**

[OCL]

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..\*]**

##### <Class> Scoped\_Identifier

**Description**

An identifier with a particular scope provided by a Namespace.

**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]**

The 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.

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

short\_expansion (sic) is formed by prefixing the short\_prefix 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]**

Identifier\_scope *references* a Namespace which provides the scope for the

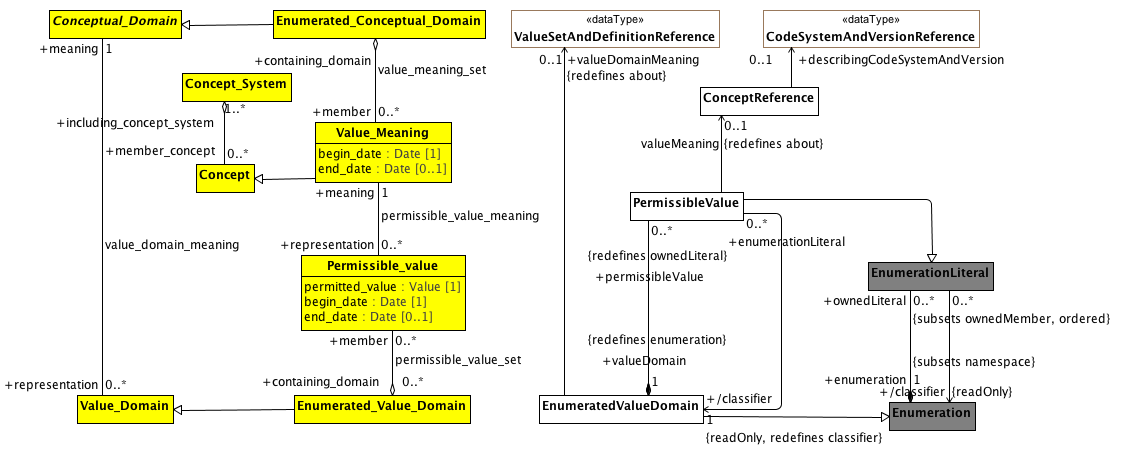
Scoped\_Identifier.

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

An identifier for the Identified\_Item within a specific Namespace.

#### <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)

**Direct Known Subclasses (Specialization)**

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

**Associations**

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

**<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..\*]**

It provides representation for a conceptual domain.

**<Class> Concept\_System**

**Description**

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

**Diagrams**

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

**Associations**

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

**<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..\*]**

**<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]**

The date at which the Permissible\_Value became valid.

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

The date at which the Permissible\_Value ceased to be valid.

**Associations**

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

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

**<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]**

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

**<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]**

The 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..\*]**

**Known other classes**

[ConceptReference](#_4082bf99060eea349dab8f548d297f3d), [EnumeratedValueDomain](#_ad639ee3d4cd535b2d3e55238d69cc51), [Enumerated\_Value\_Domain](#_f7ddc2c93ef0e7318e8f00f08a71d10c), [Enumeration](#_190e24bd48f094ad9ad981ac0b4eb47e), [EnumerationLiteral](#_41faf6a7041d7068bbbf4f9ff3924d22), [PermissibleValue](#_66976d5fcaf3eff9df49b6e5dab4ad12)

##### 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.

**Diagrams**

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

**Associations**

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

It gives meaning to the data element.

**• 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..\*]**

**Known other classes**

[Concept](#_2f48f391605b5ae297a56304194c2a30), [ConceptReference](#_4082bf99060eea349dab8f548d297f3d), [Concept\_System](#_3c62b4de493817e9b573867c82d00f2a), [ObjectConstraint](#_aa52f11e5760ad2f47030803962bb855)

### <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)

# AML Profiles

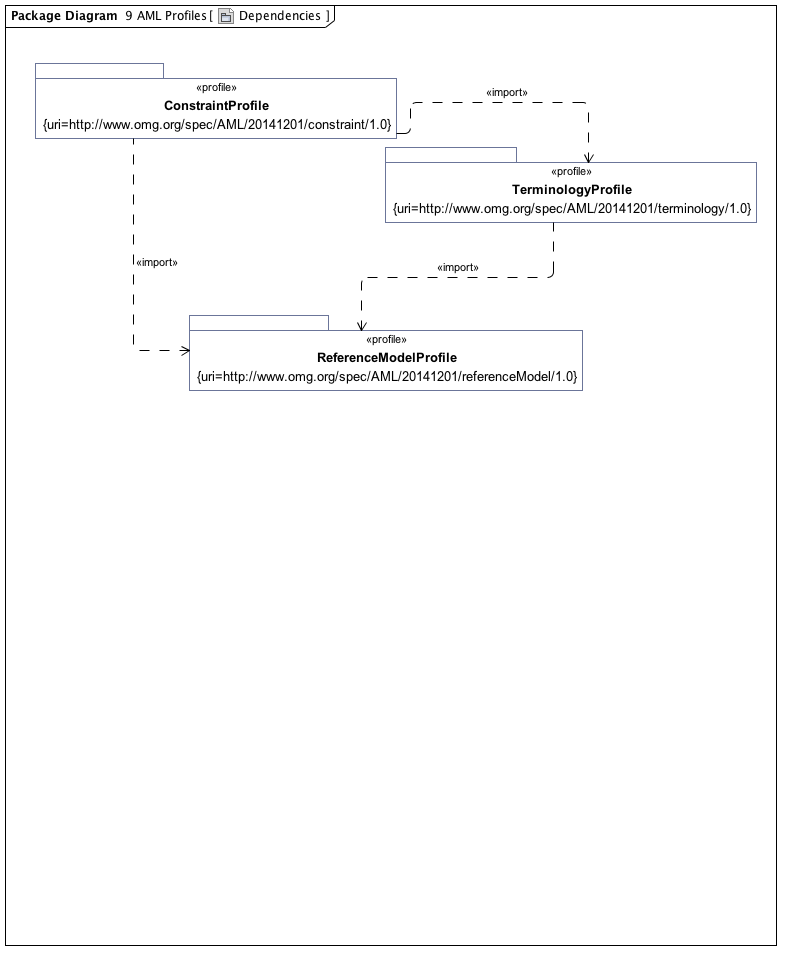
The accurate and timely communication within and across health information systems of hospitals presents an important challenge and opportunity to improving the quality of health care and patient safety. Health information by its very nature is distributed across independent and disparate solutions that are often syntactically or semantically inconsistent. This impedes users from accessing information in a normalized and meaningful manner. The objective of this submission is to provide a standard for modeling Archetype Models (AMs) using UML, to support the representation of Clinical Information Modeling Initiative (CIMI) artifacts in UML.

Archetypes are Platform Independent Models (PIMs), which are developed as a set of constraints on a specific Reference Model (RM). These archetypes are detailed and domain-specific definitions of concepts defined in terms of structured and constrained combinations of model elements within a reference model or parent archetype. These models refer to and represent clinical concepts such as heart rate, blood pressure, examination, etc. The ISO EN 13606 and OpenEHR communities define Archetypes utilizing the Archetype Definition Language (ADL).

The following models define the UML Profile “Archetype Modeling Language” (AML). The AML Profile was developed as an aggregation of three sub-profiles, which together meet the requirements of archetype modeling. The three sub-profiles of the AML Profile will include:

* Reference Model Profile (RMP): Enables the specification of reference models, upon which archetypes can be based.
* Constraint Model Profile (CMP): Support the specification of constraints on a given reference model, to enable the development of archetypes, including Clinical Information Models (CIMs).
* A Terminology Binding Profile (TBP): Provides support the binding of information models to terminology, with optional support for binding to CTS2. Terminology bindings will include:

1. Value Bindings: Linkage of the data model to value domains, which restrict the valid value of an attribute to a set of values that corresponds to a set of meanings recorded in an external terminology;
2. Semantic Bindings: Defining the meaning of model elements, using concepts in an external terminology; and
3. Constraint Bindings: Specifying constraints on the information model, using concepts and relationships defined in an external terminology.



**Dependencies**

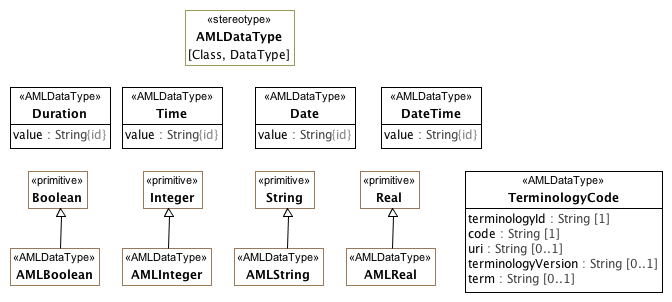
## <Package> Reference Model Profile

This section defines the set of data types whose values can be directly constrained by an AML Model. It also specifies a small set of stereotypes that are used to "decorate" a reference model and its various components.

### <Package> Assumed Data Types

This section defines the <<AMLDataType>> stereotype which is used to identify the "primitive" AML types -- Integers, Strings, Dates, Durations, etc. whose values can be constrained in an AML model.

One of the steps in constraining a Reference Model is to define how the set of AMLDataTypes map to corresponding elements in the Reference Model itself. If the Reference Model already uses the UML String, Integer, Real, and Boolean types, no further steps are necessary. While Reference Models can reference the other AML Types directly (Duration, Time, Date, DateTime, and TerminologyCode), models will typically need to define a transformation from the AMLDataTypes into the model equivalent.



**Assumed Data Types**

The AMLDataType stereotype serves two purposes:

1. To identify the set of atomic types whose possible values can be constrained in the AML Constraint Profile
2. To identify the set of types whose value will be treated as "data types" from the AML perspective.

The AMLDataType stereotype can extend both Class and DataType elements. The target reference model may choose to represent some or all of the AML DataTypes in a different fashion. A reference model may define its own String DataType rather than using the UML Native String type directly. Similarly, it may choose to represent a Date as a complex object consisting of year, month, day, granularity, zone, etc.

One of the tasks for a reference model implementer is to create maps from the appropriate AML Data Types and the corresponding reference model types.

#### <Class> Date

**Description**

Represents an absolute point in time, as measured on the Gregorian calendar, and specified only to the day.

**Diagrams**

[Assumed Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d), [Sample Data Binding](#_8548f4699761fde908c0fa2fe95f29ba)

**Direct Known Subclasses (Specialization)**

[MappedDate](#_2a37a0f4cd69c5ee1a32f787b23de875)

**Attributes**

**• public value : String**

ISO8601 string for date, in format **YYYYMMDD** or **YYYY-MM-DD**, or a partial invariant.

#### <Class> DateTime

**Description**

Represents an absolute point in time, specified to the second.

**Diagrams**

[Assumed Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d)

**Attributes**

**• public value : String**

A valid ISO8601 string for date/time, in format

**YYYYMMDDThhmmss[,sss][Z | ±hh[mm]]** or in extended format **YYYY-MM-DDThh:mm:ss[,sss][Z | ±hh[mm]]** or a partial variant.

Note that this class includes 2 deviations from ISO 8601:2004:

• for partial date/times, any part of the date/time up to the month may be missing, not just seconds and minutes as in the standard;

• the time 24:00:00 is not allowed, since it would mean the date was really on the next day.

#### <Class> Duration

**Description**

Represents a period of time corresponding to a difference between two time- points.

**Diagrams**

[Assumed Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d)

**Attributes**

**• public value : String**

Value is a valid ISO 8601 duration, i.e. takes the form:

• **P[nnY][nnM][nnW][nnD][T[nnH][nnM][nnS]]** Where each nn represents a number of years, months, etc. nnW represents a number of 7- day weeks.

Note: allowing the **W** designator in the same expression as other designators is an exception to the published standard, but necessary in clinical information (typically for representing pregnancy duration).

#### <Class> TerminologyCode

**Description**

**TerminologyCode** represents the contents of either an **EnumerationLiteral** or a **PermissibleValue**. It corresponds to the AOM **Terminology\_Code**, and its function is to gather sufficient information that, in combination with the KnownNamespace Enumeration extension, it will be possible to populate a variety of target coded types, including the ISO 21090 CD type, the openEHR DV\_CODED\_TEXT, the CIMI CODED\_TEXT, etc.

**Diagrams**

[Assumed Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d)

**Attributes**

**• public terminologyId : String [1]**

The *name* or *uri* of the *namespace* that scopes *code.*

Either the *name* of the **Enumeration** that owns the **EnumerationLiteral** represented by the *code* attribute or the *uri* tag of the **Enumeration** if it is extended by the **ScopedIdentifier** stereotype and the tag is present.

**• public code : String [1]**

The representation of an **EnumerationLiteral.** This will usually be the name of the **EnumerationLiteral**, but it should be possible for implementations to associate other numeric or string values with literals through either the *slot* or *id* of an **IdentifiedItem** extension that will become the code value instead.

**• public uri : String [0..1]**

The URI of a **ConceptReference**. This should be populated in the case where the selected value is a **PermissibleValue** and either (a) the owning **Enumeration** is stereotyped by **ScopedIdentifier** and it has a *uriPattern* attribute or (b) the **PermissibleValue** includes a *meaning* tag of type **ConceptReference**, in which case *uri* equals the *uri* of the **ConceptReference**.

If two TerminologyCodes have the same *uri*, they are considered to be identical. If a uri is absent in one or both TerminologyCodes, they are considered to be identical only if the *terminologyId* and *code* are the same.

**• public terminologyVersion : String [0..1]**

The uri of a CodeSystem or CodeSystemVersion that carried a description of the particular code at some point in time. This attribute is strictly informative and should is not part of the identity of the terminology code.

If the source **EnumerationLiteral** is stereotyped by **PermissibleValue**, the meaning tag is present and the **ConceptReference** in the meaning tag has a **CodeSystemAndVersion**, this is set to:

* The *uri* of the **CodeSystemVersionReference** if it is present
* The *uri* of the **CodeSystemReference** otherwise.

**• public term : String [0..1]**

A string designating the intended target of the **TerminologyCode** in a given language and context. This is strictly informative and is not part of the part of the identity of a **TerminologyCode** as it may change as data moves through different languages or contexts.

#### <Class> Time

**Description**

Represents an absolute point in time from an origin usually interpreted as meaning the start of the current day, specified to the second.

**Diagrams**

[Assumed Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d)

**Attributes**

**• public value : String**

Value representation is ISO8601 string for time, i.e. in form: **hhmmss[,sss][Z|±hh[mm]]** or the extended form: **hh:mm:ss[,sss][Z|±hh[mm]]**, or a partial invariant.

A small deviation to the ISO 8601:2004 standard in this class is that the time 24:00:00 is not allowed, for consistency with **DateTime**.

#### <Stereotype> AMLDataType

**Description**

The **AMLDataType** stereotype represents a built in AML data type. Instances of classifiers that are extended by **AMLDataType** classifierstypes have the same identity semantics as the UML **PrimitiveType**.

**Diagrams**

[Assumed Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d)

#### <Primitive Type> AMLBoolean

**Description**

Boolean type used for two-valued mathematical logic.

**Diagrams**

[Assumed Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d)

**Direct Known Superclasses (Generalization)**

Boolean

#### <Primitive Type> AMLInteger

**Description**

The AML Integer type.

**Diagrams**

[Assumed Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d), [Sample Data Binding](#_8548f4699761fde908c0fa2fe95f29ba)

**Direct Known Superclasses (Generalization)**

Integer

**Direct Known Subclasses (Specialization)**

[MappedInteger](#_f30eca8742491a4fd68cc825a7079d28)

#### <Primitive Type> AMLReal

**Description**

Type used to represent decimal numbers. AMLReal typically corresponds to a single-precision floating point value in most languages.

**Diagrams**

[Assumed Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d)

**Direct Known Superclasses (Generalization)**

Real

#### <Primitive Type> AMLString

**Description**

The AML String type.

**Diagrams**

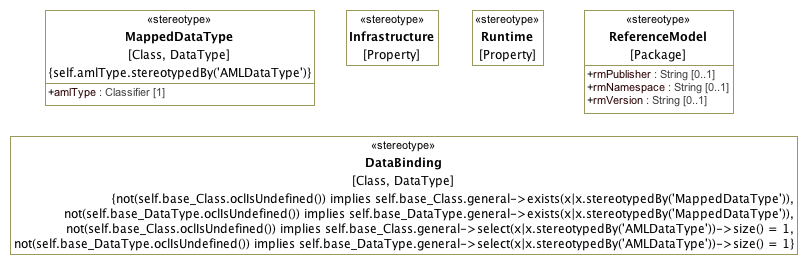
[Assumed Data Types](#_c6849d5d118a9832b3df7e75a9d1a83d)

**Direct Known Superclasses (Generalization)**

String

### <Package> Reference Model Decorators

One of the steps in constraining a Reference Model is to define how the set of AMLDataTypes map to corresponding elements in the Reference Model itself. If the Reference Model already uses the UML String, Integer, Real, and Boolean types, no further steps are necessary. While Reference Models can reference the other AML Types directly (Duration, Time, Date, DateTime, and TerminologyCode), models will typically need to define a transformation from the AMLDataTypes into the model equivalent.



**Reference Model Decorators**

The following clauses define the profiles allowing various forms of constraints on UML **Classes**, **Properties**, **Enumerations** and **EnumerationLiterals**.

#### <Stereotype> DataBinding

**Description**

The **DataBinding** stereotype identifies a "bridge" class providing a mapping between an internal AML data type identified by the **AMLDataType** stereotype and a corresponding Reference Model **DataType** or **Class** identified by the **MappedDataType** stereotype.

The Class or DataType extended by a DataBinding functions as the base point for AML constraints.

**Diagrams**

[Reference Model Decorators](#_549d0bc2f9c4ac0312fc619b71bf4d08)

**Constraints**

* **oneAMLDataType1**

This Class must specialized an <<AMLDataType>>

[OCL]

not(self.base\_Class.oclIsUndefined()) implies self.base\_Class.general->select(x|x.stereotypedBy('AMLDataType'))->size() = 1

* **oneAMLDataType2**

This DataType must specialize an <<AMLDataType>>

[OCL]

not(self.base\_DataType.oclIsUndefined()) implies self.base\_DataType.general->select(x|x.stereotypedBy('AMLDataType'))->size() = 1

* **atLeastOneMappedDataType1**

This Class must be stereotyped by <<MappedDataType>>

[OCL]

not(self.base\_Class.oclIsUndefined()) implies self.base\_Class.general->exists(x|x.stereotypedBy('MappedDataType'))

* **atLeastOneMappedDataType2**

This DataType must be specialized from <<MappedDataType>>

[OCL]

not(self.base\_DataType.oclIsUndefined()) implies self.base\_DataType.general->exists(x|x.stereotypedBy('MappedDataType'))

#### <Stereotype> Infrastructure

**Description**

A stereotype indicating a base Property represents an aspect of an Archetype implementation such as a specific Archetype identifier or other element. Properties with the Infrastructure cannot be constrained in AML.

**Diagrams**

[Reference Model Decorators](#_549d0bc2f9c4ac0312fc619b71bf4d08)

#### <Stereotype> MappedDataType

**Description**

The **MappedDataType** stereotype extends a Class or DataType in the Reference Model. It indicates the base Class or DataType corresponds to the assumed AMLDataType referenced by amlType.

Reference Model implementors will need to define a mapping from the referenced amlType and the corresponding base Class or DataType that will transform the AML values into the corresponding target values.

**Diagrams**

[Reference Model Decorators](#_549d0bc2f9c4ac0312fc619b71bf4d08)

**Attributes**

**• public amlType : Classifier [1]**

The assumed or built-in AML Data Type corresponding to the extended Class or DataType.

**Constraints**

* **isAMLDataType**

The amlType must reference a classifier (Class or DataType) that has a AMLDataType stereotype.

[OCL]

self.amlType.stereotypedBy('AMLDataType')

#### <Stereotype> ReferenceModel

**Description**

This stereotype identifies a package as a Reference Model -- a package which contains the collection of UML Classes that can be constrained by the Archetypes in an Archetype Library. The Reference Model stereotype also allows the specification of the publisher, namespace and version of a Reference Model in a form compatible with a modeling language such as ADL.

**Diagrams**

[Reference Model Decorators](#_549d0bc2f9c4ac0312fc619b71bf4d08)

**Attributes**

**• public rmPublisher : String [0..1]**

The name of the Reference Model publisher. Corresponds to *rm\_publisher* in AOM 2.0

**• public rmNamespace : String [0..1]**

The owning domain name of the archetype. Corresponds to the *namespace* attribute in AOM2.0.

**• public rmVersion : String [0..1]**

Designates the version id of the reference model on which the archetype is based. Corresponds to *rm\_release* in AOM 2.0

#### <Stereotype> Runtime

**Description**

A stereotype indicating a base Property represents an identifier, date or other element that is a part of the identity of an instance and cannot be constrained in the AML.

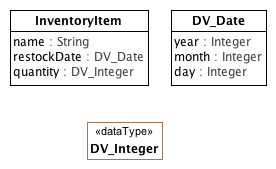
**Diagrams**

[Reference Model Decorators](#_549d0bc2f9c4ac0312fc619b71bf4d08)

#### <Package> DataBinding Example

The following section is a non-normative example of how the AMLDataType, MappedDataType and DataBinding stereotypes are used to connect an AML Archetype to a Reference Model.

##### <Package> Sample Reference Model



**Sample Reference Model**

The figure above shows a sample reference model. The class, InventoryItem has three attributes:

* *name*: type **String** (a UML **PrimitiveType**)
* *restockDate* : type **DV\_Date** (a **Class**)
* *quantity* : type **DV\_Integer** (a **DataType**)

**<DataType> DV\_Integer**

**Description**

A specialized Integer data type used by the sample reference model.

**Diagrams**

[Sample Reference Model](#_ee9ffd00f21ae726d591ff29117ce0b2)

**<Class> DV\_Date**

**Description**

A specialized Date type used by the sample reference model.

**Diagrams**

[Sample Reference Model](#_ee9ffd00f21ae726d591ff29117ce0b2)

**Attributes**

**• public year : Integer**

**• public month : Integer**

**• public day : Integer**

**<Class> InventoryItem**

**Diagrams**

[Sample Reference Model](#_ee9ffd00f21ae726d591ff29117ce0b2)

**Attributes**

**• public name : String**

**• public restockDate :** [**DV\_Date**](#_84cbb28f8eb841516835fc6301503e04)

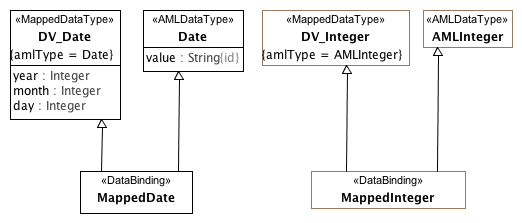
**• public quantity :** [**DV\_Integer**](#_8c16749c116ad386c9e05ce56a59a116)

**<Class> DateItem**

**Diagrams**

[Sample Reference Model](#_ee9ffd00f21ae726d591ff29117ce0b2),

##### <Package> Sample DataBinding



**Sample Data Binding**

This diagram shows the two data types defined in the Sample Reference Model data types with added MappedDataType stereotype. This identifies them as a mapped representation of an AML primitived type, which means AML constraints cannot be applied to their owned attributes (e.g. *year* cannot be constrained within the **DV\_Date** class because it is is mapped from the AML **Date** *value*.)

**<DataType> DV\_Integer**

**Description**

A specialized Integer data type used by the sample reference model.

**Diagrams**

[Sample Data Binding](#_8548f4699761fde908c0fa2fe95f29ba)

**Direct Known Subclasses (Specialization)**

[MappedInteger](#_f30eca8742491a4fd68cc825a7079d28)

**<Class> DV\_Date**

**Description**

The Date data type as used by the sample reference model.

**Diagrams**

[Sample Data Binding](#_8548f4699761fde908c0fa2fe95f29ba)

**Direct Known Subclasses (Specialization)**

[MappedDate](#_2a37a0f4cd69c5ee1a32f787b23de875)

**Attributes**

**• public year : Integer**

**• public month : Integer**

**• public day : Integer**

**<Class> MappedDate**

**Description**

The data binding class specializing both the **Date** data type and its corresponding **DV\_Date** class in the reference model.

**Diagrams**

[Sample Data Binding](#_8548f4699761fde908c0fa2fe95f29ba)

**Direct Known Superclasses (Generalization)**

[Date](#_78ee642abf9938398776ce11b2ae5595), [DV\_Date](#_acf6f25348d09f64751e3145a2a78d4a)

**Direct Known Subclasses (Specialization)**

[AcmeDate](#_3508926265c7d0fc2aff69e7498f5c24)

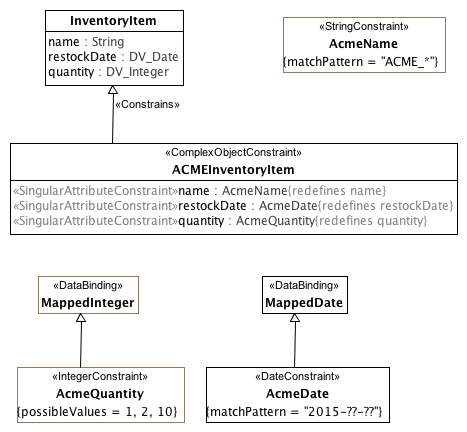
**Known other classes**

[Date](#_78ee642abf9938398776ce11b2ae5595)

**Known other primitive types**

[AMLInteger](#_8e49c0c434d339e5881155e37d0822a0)

##### <Package> Sample Constraint



**Sample Constraint**

The diagram above shows how attribute constraints are applied to the sample reference model mapping. The first constraint, **AcmeName** is a specialization of the UML PrimitiveType, **String**. As the *name* attribute in the **Inventory** class is of the same type, no data binding is needed -- the **ACMEInventoryItem** constraint can directly redefine it.

In the case of *restockDate* and *quantity*, the AML profile has no knowledge of the **DV\_Integer** and **DV\_Date** types. The **MappedInteger** and **MappedDate** classes allow AML constraints to be applied to the corresponding AML primitive types of **AMLInteger** and **Date** respectively, while simutaneously applying to and allowing redefinition of the **InventoryItem** *restockDate* and *quantity* attributes.

**<Class> AcmeDate**

**Description**

Instances of **MappedDate** whose year is 2015

**Diagrams**

[Sample Constraint](#_40567bb1c344f2e9ddb760c83dac78ce)

**Direct Known Superclasses (Generalization)**

[MappedDate](#_2a37a0f4cd69c5ee1a32f787b23de875)

**<Class> ACMEInventoryItem**

**Description**

A sample inventory item in the ACME Manufacturing database. An **ACMEInventoryItem** must have:

* A *name* that begins with the string "ACME\_"
* A *quantity* of 1, 2, or 10
* A *restockDate* that occurs in 2015

**Diagrams**

[Sample Constraint](#_40567bb1c344f2e9ddb760c83dac78ce)

**Direct Known Superclasses (Generalization)**

[InventoryItem](#_7a3cd0274dcc9ef68ed24910598c14f3)

**Attributes**

**• public name :** [**AcmeName**](#_04c8152f4b07cff6c2e1f26a2720266a)

**• public restockDate :** [**AcmeDate**](#_3508926265c7d0fc2aff69e7498f5c24)

**• public quantity :** [**AcmeQuantity**](#_0e0bd56f8557928d9e58d67df05a7c2d)

**<Class> InventoryItem**

**Description**

An inventory item in a sample reference model.

**Diagrams**

[Sample Constraint](#_40567bb1c344f2e9ddb760c83dac78ce)

**Direct Known Subclasses (Specialization)**

[ACMEInventoryItem](#_394193c1c648aef51e06567f267d32ef)

**Attributes**

**• public name : String**

The name of the item

**• public restockDate :** [**DV\_Date**](#_acf6f25348d09f64751e3145a2a78d4a)

The date the item should be restocked

**• public quantity :** [**DV\_Integer**](#_1ffc2af295659e44b62cead5148bcf04)

The number of items in the inventory.

**Known other classes**

[MappedDate](#_2a37a0f4cd69c5ee1a32f787b23de875)

## <Package> Terminology Profile

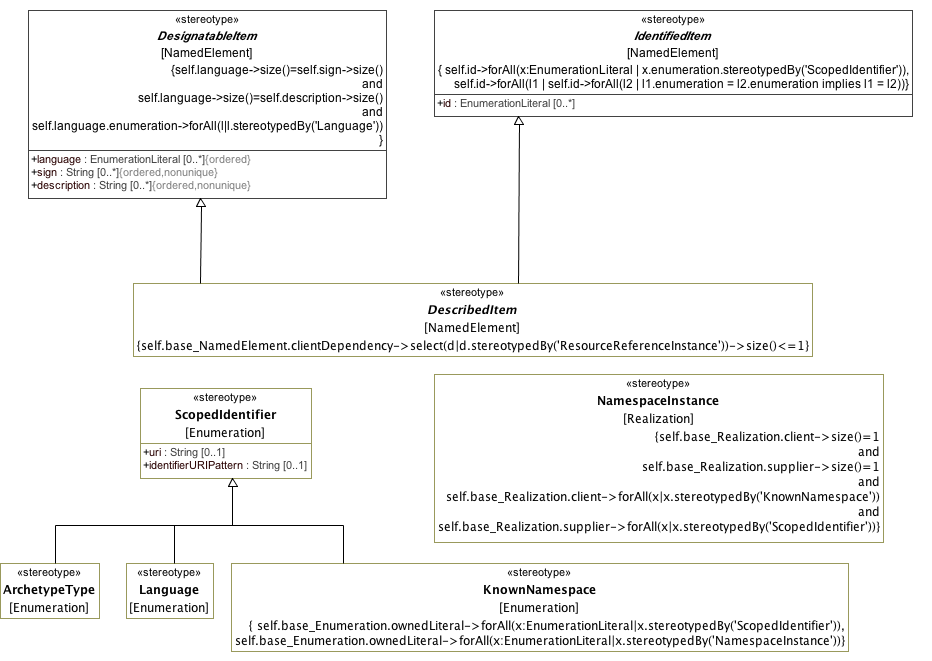
The AML Terminology Profile is the UML equivalent of the ADL 2.0 terminology section, including:

* Identifiers -- "id", "at" and "ac" identifiers may be assigned to Class constraints, EnumerationLiterals and Enumerations respectively.
* Term definitions -- multilingual designation ("text") / description tuples may be assigned to any named AML model artifact
* Term bindings
  + Class constraints may be associated with a reference to an external resource that the constraint is "about"
  + Enumeration constraints may be associated with a reference to an external value set constraining the set of possible values and value meanings
  + TerminologyCode (Permissible) values may be associated with "concept" (aka. class, category, term) references providing the meaning for the value
* Value Sets -- local enumerations may be defined that associate collections of individual codes (ADL "at" codes) with a local value set or enumeration (ADL "ac" code).

The Terminology Binding profile draws on the ISO 11179-3 model for the identification, designation, definition and value / meaning binding aspects and on the OMG Common Terminology Services 2 (CTS2) specification for the model of Concept, Code System, Code System Version, Value Set and Value Set Definition references.

### <Package> Identification and Designation

The first section in the AML Terminology Profile focuses on resource identifiers, the equivalent of which in the ADL/AOM 2.0 specification would be the "id", "at" and "ac" codes, their namespaces and their multilingual designations and descriptions.



**IdentificationAndDesignation**

#### <Stereotype> ArchetypeType

**Description**

An implementation specific enumeration of possible archetype types. In ADL, this would extend an enumeration that includes literals for *archetype*, *template, template\_overlay* and *operational\_template* and their flattened equivalents.

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Superclasses (Generalization)**

[ScopedIdentifier](#_59faf6918f4c546323d6df67392c366b)

#### <Stereotype> DescribedItem

**Description**

The **DescribedItem** stereotype allows multilingual designations/descriptions and scoped identifiers to be assigned a **NamedElement**.

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Superclasses (Generalization)**

[DesignatableItem](#_80448b03d480bba05b1e156796878f77), [IdentifiedItem](#_4b28f60cd7e8328f1d31dbcfa39d2ff3)

**Direct Known Subclasses (Specialization)**

[EnumeratedValueDomain](#_c7f411daaf64f83e013bec437cb8f30a), [ObjectConstraint](#_ad75af95f635bdf35f69d9db9b17aae2), [PermissibleValue](#_5bb7ce8128b60ee5eb2ca275444e9692), [ResourceReference](#_1b2eec63ad4ef6c72d57b9985e0346ff)

**Constraints**

* **validAbout**

There must be a maximum of one <<ResourceReferenceInstance>> Abstraction

[OCL]

self.base\_NamedElement.clientDependency->select(d|d.stereotypedBy('ResourceReferenceInstance'))->size()<=1

#### <Stereotype> DesignatableItem

**Description**

The **DesignatableItem** stereotype allows a set of language / sign / description tuples to be assigned to a **NamedElement**. This represents a flattened version of the **ItemDescription** in the AML Meta Model.

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Subclasses (Specialization)**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1), [ResourceReference](#_1b2eec63ad4ef6c72d57b9985e0346ff), [ResourceReferenceInstance](#_9d682f32f4917feea358e696d1fd146d)

**Attributes**

**• public language : EnumerationLiteral [0..\*]**

A reference to a spoken or written language. *language* must be an **EnumerationLiteral** in an **Enumeration** stereotyped by the **Language** stereotype. Only one *sign*/description tuple is allowed per language.

**• public sign : String [0..\*]**

A string designating or "signifying" the name of the *NamedElement* in the corresponding *language. sign* is a required field and cannot contain an empty String ("").

**• public description : String [0..\*]**

A definition or description of the **NamedElement** in the corresponding *language*. *description* is optional and its absence is indicated by the empty ("") String.

**Constraints**

* **language sign description tuples are designations**

The ordered sequences language, sign, description must be same length and language must be in a <<Language>>.

[OCL]

self.language->size()=self.sign->size() and self.language->size()=self.description->size() and self.language.enumeration->forAll(l|l.stereotypedBy('Language'))

#### <Stereotype> IdentifiedItem

**Description**

The **IdentifiedItem** stereotype allows the assignment of one or more optional identifiers to be assigned to a **NamedElement**.

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Subclasses (Specialization)**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1), [ResourceReferenceInstance](#_9d682f32f4917feea358e696d1fd146d)

**Attributes**

**• public id : EnumerationLiteral [0..\*]**

An identifier unique within the context of the containing **Enumeration**.

**Constraints**

* **scopedIdentifierLiteral**

Every *id* property is an instance of a ScopedIdentifier.

[OCL]

self.id->forAll(x:EnumerationLiteral | x.enumeration.stereotypedBy('ScopedIdentifier'))

* **uniqueScopes**

Every id must belong to a unique instance specification classifier. An identified Item cannot have two or more identifiers drawn from the same ScopedIdentifier enumeration.

[OCL]

self.id->forAll(l1 | self.id->forAll(l2 | l1.enumeration = l2.enumeration implies l1 = l2))

#### <Stereotype> KnownNamespace

**Description**

**KnownNamespace** extends the **Enumeration** listing the set of namespace identifiers recognized by a particular implementation or archetype library. The name of a member literal represents a local namespace identifier. Each local identifier can, in turn, be realized by a second namespace. As an example, an AML model might have 3 known namespaces:

1. "SCT"
2. "SNOMED-CT"
3. "LOINC"

The first two namespaces would reference the same realization target, a **ScopedIdentifier** for the SNOMED CT identifiers referenced by the model, while the third would reference another **ScopedIdentifier** **Enumeration** containing the LOINC identifiers referenced by the model.

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Superclasses (Generalization)**

[ScopedIdentifier](#_59faf6918f4c546323d6df67392c366b)

**Constraints**

* **hasScopedIdentifiers**

Set if knownNamespace has an identifier uniquely referencing its scoping namespace.

[OCL]

self.base\_Enumeration.ownedLiteral->forAll(x:EnumerationLiteral|x.stereotypedBy('ScopedIdentifier'))

* **namespaceInstances**

All owned EnumerationLiterals must be <<NamespaceInstance>>

[OCL]

self.base\_Enumeration.ownedLiteral->forAll(x:EnumerationLiteral|x.stereotypedBy('NamespaceInstance'))

#### <Stereotype> Language

**Description**

The set of languages referenced by an AML instance. Typically these will be drawn from ISO 639-1 or one of its derivatives.

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Superclasses (Generalization)**

[ScopedIdentifier](#_59faf6918f4c546323d6df67392c366b)

#### <Stereotype> NamespaceInstance

**Description**

NamespaceInstance is an instance of Namespace.

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Constraints**

* **mustBeScopedIdentifier**

There must be a single <<KnownNamespace>> client and a single <<ScopedIdentifier>> supplier.

[OCL]

self.base\_Realization.client->size()=1 and self.base\_Realization.supplier->size()=1 and self.base\_Realization.client->forAll(x|x.stereotypedBy('KnownNamespace')) and self.base\_Realization.supplier->forAll(x|x.stereotypedBy('ScopedIdentifier'))

#### <Stereotype> ScopedIdentifier

**Description**

A **ScopedIdentifier** is an enumeration that may include a uri referencing the scope of the identifier and a uri pattern describing how uri's are constructed from the member EnumerationLiterals.

As an example, the SNOMED CT identifier namespace might have a *uri* of "http://snomed.info/sct", which identifies the sole namespace and an *identifierURIPattern* of "http://snomed.info/id/$1" which indicates an EnumerationLiteral named 74400008 would be represented as "http://snomed.info/id/74400008".

**Diagrams**

[IdentificationAndDesignation](#_0628ce7e5381e7543ba417bb320e03fb)

**Direct Known Subclasses (Specialization)**

[ArchetypeType](#_7dc1530ae1ef855ecc3eb9bd5b555a14), [KnownNamespace](#_b9f78b93edc24bb3301ba69a57e4afc3), [Language](#_446e0591a4f825b22cd9e573c1239a72)

**Attributes**

**• public uri : String [0..1]**

A URI referencing the namespace associated with the ScopedIdentifier. Examples: <http://snomed.info/sct> http://loinc.org

**• public identifierURIPattern : String [0..1]**

A URI substitution pattern, where "$1" indicates where the name of an owned EnumerationLiteral would be substituted to create a URI. Example: http://loinc.org/id/$1.

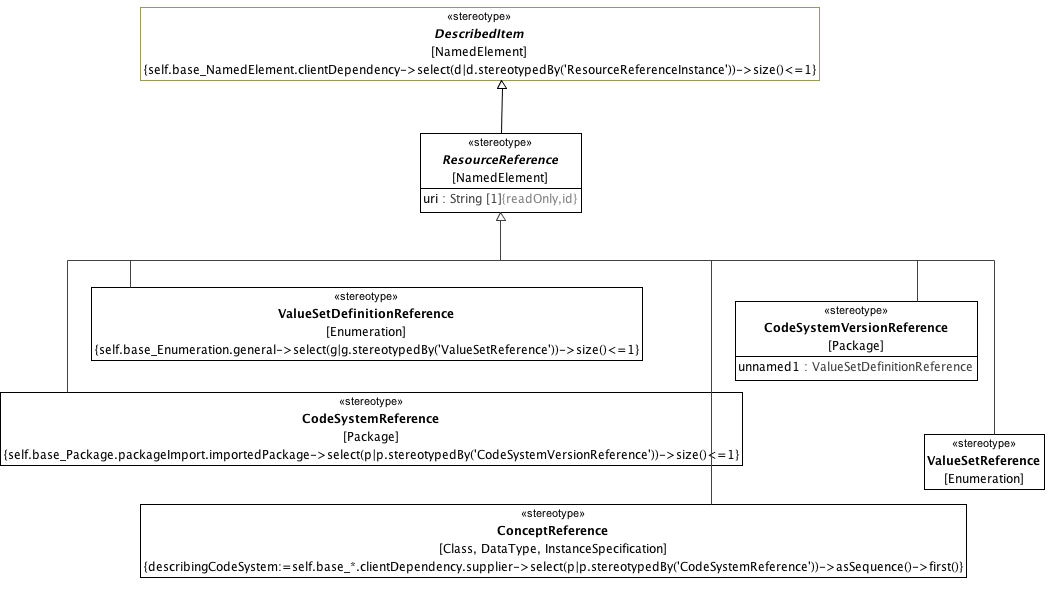
### <Package> Resource References

This clause describes the profiles corresponding to the MetaModel constructs:

* ResourceReference
* CodeSystemAndVersionReference
* ValueSetAndDefinitionReference

The first clause below defines how each of these elements are represented. The second clause defines a set of extensions to the Abstraction relationship providing links between model elements and their target ResourceReferences.

#### <Package> Resource References



**Resource References**

A URI naming the resource. This should conform to the semantics of the CTS2 **PersistentURI**.

##### <Stereotype> CodeSystemReference

**Description**

A ResourceReference whose *uri* identifies to a code system (aka. "concept system, "terminology" or "ontology"). Note, while the UML **Package** is the base for the **CodeSystemReference**, its intent is to represent a the *uri* and optional name of a code system and a single, optional *version* and should contain nothing else. The optional *version* of type **CodeSystemVersionReference** is represented as a *packageImport*.

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ResourceReference](#_1b2eec63ad4ef6c72d57b9985e0346ff)

**Constraints**

* **version**

There must be at most 1 imported <<CodeSystemVersionReference>> Package.

[Binary]

self.base\_Package.packageImport.importedPackage->select(p|p.stereotypedBy('CodeSystemVersionReference'))->size()<=1

##### <Stereotype> CodeSystemVersionReference

**Description**

A URI referencing a specific version of a code system. Note, while UML **Package** is used as the base for the **CodeSystemVersionReference** stereotype, a **Package** extended by **CodeSystemVersionReference** may not have any members. It is simply the name of a code system version and not the actual entity itself.

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ResourceReference](#_1b2eec63ad4ef6c72d57b9985e0346ff)

**Attributes**

**• public unnamed1 :** [**ValueSetDefinitionReference**](#_a4fedb7858ead8d2272640d51b53719a)

##### <Stereotype> ConceptReference

**Description**

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

**ConceptReference** can be used to bind a **Class** or **DataType** constraint to its terminology binding (aka "meaning" in ISO 11179-3). **ConceptReference** may also be associated with a describing code system and optional code system version by a usage relationship whose client is **ConceptReference** and whose supplier is **CodeSystemReference**. The purpose of this association is to identify the particular description that was used when selecting the concept reference.

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ResourceReference](#_1b2eec63ad4ef6c72d57b9985e0346ff)

**Constraints**

* **describingCodeSystem**

Definition of describingCodeSystem

describingCodeSystem:=self.base\_\*.clientDependency.supplier->select(p|p.stereotypedBy('CodeSystemReference'))->asSequence()->first()

[English]

describingCodeSystem:=self.base\_\*.clientDependency.supplier->select(p|p.stereotypedBy('CodeSystemReference'))->asSequence()->first()

##### <Stereotype> ResourceReference

**Description**

A ResourceReference couples a NamedElement with a uri referencing an external class, category, individual or "concept". It should be noted the uri in ResourceReference has the semantics associated with the PersistentURI in the CTS2 specification -- it is not intended to reference a resource directly, but to "name" a resource that has a description in one or more terminologies, (code systems, classification systems, ontologies).

The uri serves as the identity of a ResourceReference. It may be accompanied by additional language specific designations and descriptions as well as by a ScopedIdentifer that identifies the target as a namespace/name tuple.

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1), [DesignatableItem](#_80448b03d480bba05b1e156796878f77)

**Direct Known Subclasses (Specialization)**

[CodeSystemReference](#_7de70b4fbd9e6a164f7f00cde47dfd5a), [CodeSystemVersionReference](#_9a8de95c38ebe2d6ce506cbc9bef7b7a), [ConceptReference](#_57ae94153b82f28889d42ad4aa8fe1e0), [ValueSetDefinitionReference](#_a4fedb7858ead8d2272640d51b53719a), [ValueSetReference](#_1a1ca20b54028ee5e2eb20af35411f6e)

**Attributes**

**• public uri : String [1]**

URI of the resource.

**Constraints**

* **uniqueId**

Every identifier must come from a different namespace

[OCL]

self.id->forAll(i1 | self.id->forAll(i2 | i1.enumeration = i2.enumeration implies i1 = i2))

* **language sign description represent tuple**

The ordered sequences language, sign, description must be the same length and language must be part of a <<Language>> Enumeration.

[OCL]

self.language->size()=self.sign->size() and self.language->size()=self.description->size() and self.language.enumeration->forAll(l|l.stereotypedBy('Language'))

##### <Stereotype> ValueSetDefinitionReference

**Description**

A *uri* that names the value set that the extended **Enumeration** represents. The set of **EnumerationLiterals** in an **Enumeration** stereotyped by **ValueSetDefinitionReference** represent the set or a subset of the **ConceptReferences** returned by the interpretation of the referenced definition.

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ResourceReference](#_1b2eec63ad4ef6c72d57b9985e0346ff)

**Constraints**

* **specializesValueSet**

A ValueSetDefinitionReference is a specialization of a ValueSetReference.

[OCL]

self.base\_Enumeration.general->select(g|g.stereotypedBy('ValueSetReference'))->size()<=1

##### <Stereotype> ValueSetReference

**Description**

A *uri* that names the value set that the extended Enumeration represents. The set of EnumerationLiterals in an Enumeration stereotyped by ValueSetReference represent the set or a subset of the ConceptReferences returned by the "current" (in the CTS2 sense) definition of the ValueSet.

**Diagrams**

[Resource References](#_aa2597b87275ed4e1724a94a9c31d90b)

**Direct Known Superclasses (Generalization)**

[ResourceReference](#_1b2eec63ad4ef6c72d57b9985e0346ff)

**Constraints**

* **definition**

This Enumeration must have exactly one <<ValueSetDefinitionReference>> Generalization.

[OCL]

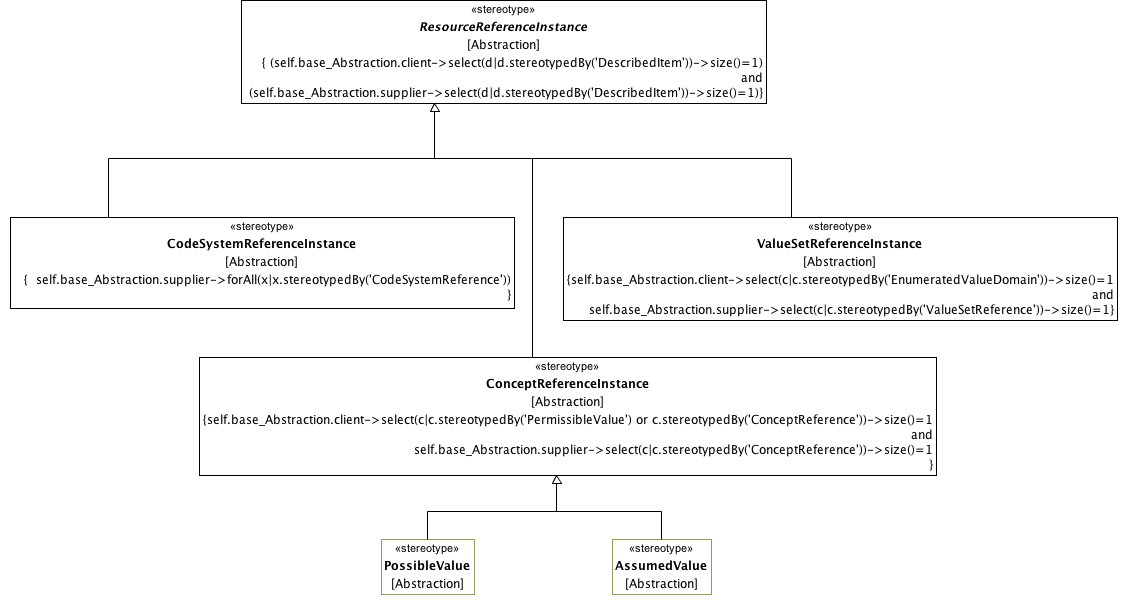
self.base\_Enumeration.general->select(g|g.stereotypedBy('ValueSetDefinitionReference'))->size()<=1

**Known other Stereotypes**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1)

#### <Package> Resource Reference Relationships

This section describes the set of associations that allow the various types of ResourceReference to be linked to their referencing elements.



**Resource Reference Relationships**

A URI that names the resource. This should conform to the semantics of the CTS2 **PersistentURI**.

##### <Stereotype> AssumedValue

**Description**

An instance of a link between a **ConceptReferenceConstraint** and an assumedValue.

**Diagrams**

[Resource Reference Relationships](#_6d065336864e77a240c6e7a3eb36e8cd)

**Direct Known Superclasses (Generalization)**

[ConceptReferenceInstance](#_6a2f733d1d3ea9bc232f96caadb113e1)

##### <Stereotype> CodeSystemReferenceInstance

**Description**

**CodeSystemReferenceInstance** represents the optional link between a **ConceptReference** and a describing code system and optional version.

**Diagrams**

[Resource Reference Relationships](#_6d065336864e77a240c6e7a3eb36e8cd)

**Direct Known Superclasses (Generalization)**

[ResourceReferenceInstance](#_9d682f32f4917feea358e696d1fd146d)

**Constraints**

* **isCodeSystemReference**

Must be a valid instance of a CodeSystemReference.

[OCL]

self.base\_Abstraction.supplier->forAll(x|x.stereotypedBy('CodeSystemReference'))

##### <Stereotype> ConceptReferenceInstance

**Description**

A **ConceptReferenceInstance** associates a referencing element with a corresponding **ConceptReference**. It is used to link a **PermissibleValue** with a value meaning, an **ObjectConstraint** with what the constraint is "about" (aka. "meaning in ISO 11179) and to identify the possible and assumed values for a **ConceptReferenceConstraint**.

**Diagrams**

[Resource Reference Relationships](#_6d065336864e77a240c6e7a3eb36e8cd)

**Direct Known Superclasses (Generalization)**

[ResourceReferenceInstance](#_9d682f32f4917feea358e696d1fd146d)

**Direct Known Subclasses (Specialization)**

[AssumedValue](#_7c76bb5bbde4643a957c898bfc8af67a), [PossibleValue](#_ef85d33c740061c0c756c4e535c34ccc)

**Constraints**

* **isConceptReference**

There must be one <<PermissibleValue>> or <<ConceptReference>> client and one <<ConceptReference>> supplier.

[OCL]

self.base\_Abstraction.client->select(c|c.stereotypedBy('PermissibleValue') or c.stereotypedBy('ConceptReference'))->size()=1 and self.base\_Abstraction.supplier->select(c|c.stereotypedBy('ConceptReference'))->size()=1

##### <Stereotype> PossibleValue

**Description**

An instance of a link between a **ConceptReferenceConstraint** and a set of possible values.

**Diagrams**

[Resource Reference Relationships](#_6d065336864e77a240c6e7a3eb36e8cd)

**Direct Known Superclasses (Generalization)**

[ConceptReferenceInstance](#_6a2f733d1d3ea9bc232f96caadb113e1)

##### <Stereotype> ResourceReferenceInstance

**Description**

ResourceReferenceInstance is an abstraction that associates a NamedElement that is stereotyped by DescribedItem with an instance of its reference.

**Diagrams**

[Resource Reference Relationships](#_6d065336864e77a240c6e7a3eb36e8cd)

**Direct Known Superclasses (Generalization)**

[DesignatableItem](#_80448b03d480bba05b1e156796878f77), [IdentifiedItem](#_4b28f60cd7e8328f1d31dbcfa39d2ff3)

**Direct Known Subclasses (Specialization)**

[CodeSystemReferenceInstance](#_e055a6cce06d0838055b62dbfbf235f2), [ConceptReferenceInstance](#_6a2f733d1d3ea9bc232f96caadb113e1), [ValueSetReferenceInstance](#_f3184cb0f8e704f5122c5e97fb9f130c)

**Constraints**

* **isResourceReference**

Must be an instance of a resource reference class

[OCL]

(self.base\_Abstraction.client->select(d|d.stereotypedBy('DescribedItem'))->size()=1) and (self.base\_Abstraction.supplier->select(d|d.stereotypedBy('DescribedItem'))->size()=1)

##### <Stereotype> ValueSetReferenceInstance

**Description**

A link between a **ConceptReferenceConstraint** and a value set and optional definition, where the value set resolves to a set of **ConceptReferences**.

**Diagrams**

[Resource Reference Relationships](#_6d065336864e77a240c6e7a3eb36e8cd)

**Direct Known Superclasses (Generalization)**

[ResourceReferenceInstance](#_9d682f32f4917feea358e696d1fd146d)

**Constraints**

* **isValueSetReference**

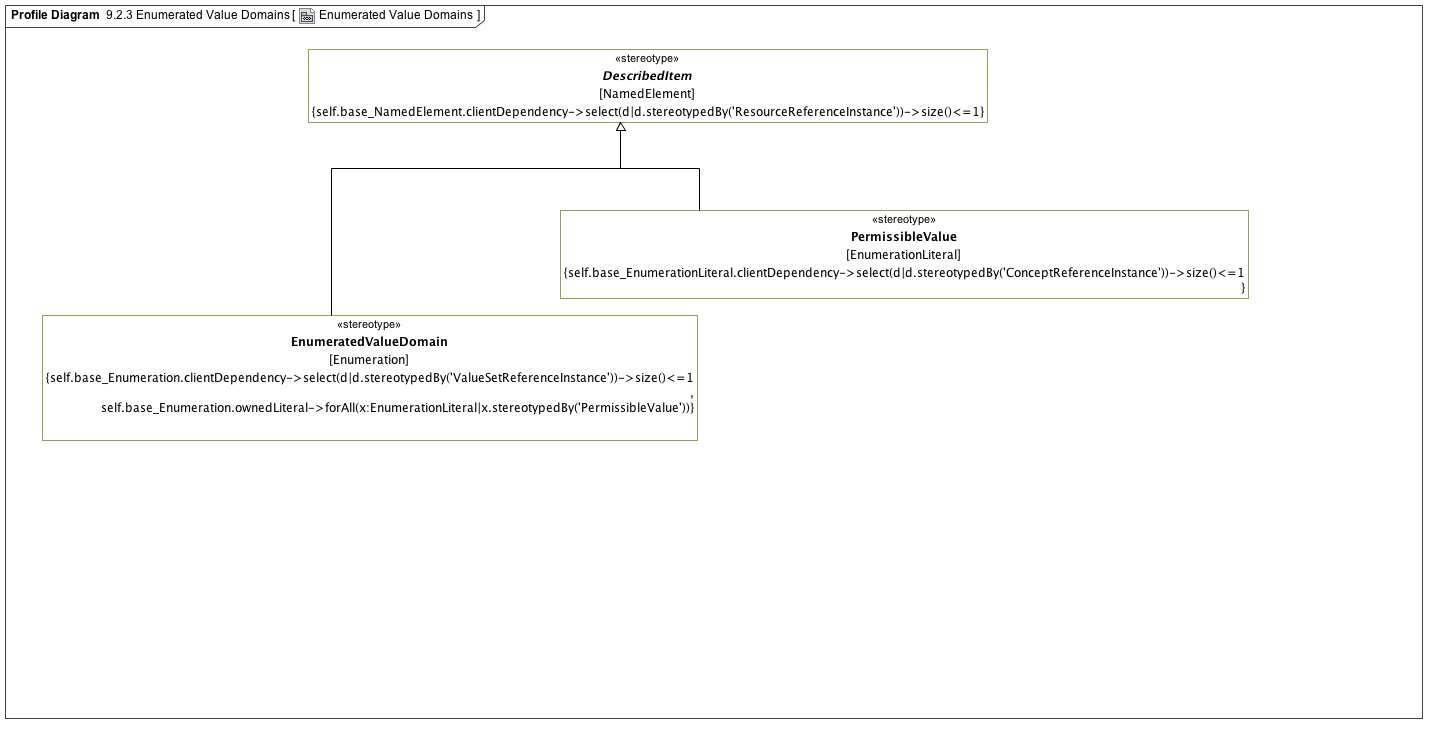
Must have one <<EnumeratedValueDomain>> client and one <<ValueSetReference>> supplier.

[OCL]

self.base\_Abstraction.client->select(c|c.stereotypedBy('EnumeratedValueDomain'))->size()=1 and self.base\_Abstraction.supplier->select(c|c.stereotypedBy('ValueSetReference'))->size()=1

### <Package> Enumerated Value Domains

This clause defines the AML extensions to Enumeration and EnumerationLiteral that represent the EnumeratedValueDomain and PermissibleValue elements respectively.



**Enumerated Value Domains**

#### <Stereotype> EnumeratedValueDomain

**Description**

An **EnumeratedValueDomain** represents a discrete set of possible values (**PermissibleValues**) for a particular field or data element. Each **PermissibleValue** identifies a unique value and (optionally) its intended meaning. An **EnumeratedValueDomain** may be associated with a scoped identifier, which, in the ADL case would be an "ac" code. The set of **PermissibleValues** for the **EnumeratedValueDomain** would represent an ADL value\_set, which would bind an "ac" code to a set of "at" codes.

An **EnumeratedValueDomain** may also reference a **ValueSetReference** using the **ValueSetReferenceInstance** relationship which would resolve to a set of **PermissibleValues**.

**Diagrams**

[Enumerated Value Domains](#_bbceab59bf026d3b0616400766b2619e)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1)

**Constraints**

* **meaningIsConceptReference**

There is a maximum of one <<ValueSetReferenceInstance>> Abstractions.

[OCL]

self.base\_Enumeration.clientDependency->select(d|d.stereotypedBy('ValueSetReferenceInstance'))->size()<=1

* **permissibleValues**

All instances must be type permissibleValue

[OCL]

self.base\_Enumeration.ownedLiteral->forAll(x:EnumerationLiteral|x.stereotypedBy('PermissibleValue'))

#### <Stereotype> PermissibleValue

**Description**

A unique within the context of the owning **EnumeratedValueDomain**. A **PermissibleValue** can be associated with an identifier, which, in ADL would be an "at" code. A permissible value can be associated with one or more language specific designations and optional descriptions, which represent the ADL term\_definitions section (as applied to "at" codes). A **PermissibleValue** may also reference a **ConceptReference** via the **ConceptReferenceInstance** association,where the **ConceptReference** identifies the intended meaning for the value.

**Diagrams**

[Enumerated Value Domains](#_bbceab59bf026d3b0616400766b2619e)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1)

**Constraints**

* **valueMeaningIsConceptReference**

PermissibleValue.about must be a concept reference

[OCL]

self.base\_EnumerationLiteral.clientDependency->select(d|d.stereotypedBy('ConceptReferenceInstance'))->size()<=1

**Known other Stereotypes**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1)

## <Package> Constraint Profile

The constraint profile defines the modeling elements that may be applied to a reference model and archetype. These elements “constrain” the target model narrowing the semantics and syntax. These elements are drawn from ADL 2.0 and ISO 13606.

### <Package> Archetypes

This clause defines the profiles for Archetypes, Archetype Libraries and their identification, provenance and workflow. It corresponds to the *archetype*, *language*, and *description* sections of ADL 2.0.

#### <Package> Archetype Libraries

This clause defines the stereotypes that represent **Archetype**, **ArchetypeLibrary** and **ArchetypeVersion** as well as defining the link to the **AuthoredResource** metadata.



**Archetypes**

##### <Stereotype> Archetype

**Description**

An **Archetype** is a package that contains a versioned set of constraints on a **Class** that is a member of the **ReferenceModel** that is owned by the containing **ArchetypeLibrary**.

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

**Attributes**

**• public archetypeName : String [1]**

The human readable name of the Archetype. Example: "clinical data group", "laboratory test", "serum sodium", "cimi composition"

**• public originalLanguage : EnumerationLiteral [0..1]**

The original spoken or written language in which the archetype was authored (Example: EN, DE, ES). A language type drawn for the language numeration.

**• public archetypeType : EnumerationLiteral [0..\*]**

The implementation specific type or classification of the archetype. In the ADL context, this would include archetype, template\_overlay, etc.

**• public rmPackagePath : String [1]**

The qualifiedName of the package containing the root class constrained by this archetype.

**• public rmClassName : String [1]**

Name of the root Class constrained of this archetype.

**Constraints**

* **mustBeOwned**

The containing Package must be an <<ArchetypeLibrary>>.

[OCL]

self.base\_Package.owningPackage.stereotypedBy('ArchetypeLibrary')

* **ownsVersions**

All members of a package must be <<ArchetypeVersion>>.

[OCL]

self.base\_Package.ownedMember->select(x|x.oclIsKindOf(Classifier) and not(x.oclIsKindOf(Association)))->forAll(x|x.stereotypedBy('ArchetypeVersion'))

* **originalLanguage**

originalLanguage must be contained by a <<Language>> Enumeration.

[OCL]

self.originalLanguage.enumeration.stereotypedBy('Language')

* **archetypeType**

Each archetypeType must be an <<ArchetypeType>>.

[OCL]

self.archetypeType.enumeration->forAll(x|x.stereotypedBy('ArchetypeType'))

* **constrainsRMElement**

The Class or one of its generalization ancestors is a member of the ArchetypeLibrary Reference Model.

[OCL]

self.base\_Package.nestingPackage.packageImport.importedPackage.nestedPackage.packagedElement->select(x|x.oclIsKindOf(Classifier)).oclAsType(Classifier) ->exists(x|self.base\_Package.packagedElement->select(p|p.stereotypedBy('ArchetypeVersion')).oclAsType(Classifier).general->includes(x))

* **currentVersion**

The package must contain one <<ArchetypeCurrentVersion>>.

[OCL]

self.base\_Package.packagedElement->select(x|x.stereotypedBy('ArchetypeCurrentVersion'))->size()=1

* **specializesArchetype**

If an Archetype specializes another Archetype, they both constrain the same class.

[OCL]

self.base\_Package.packageMerge.mergedPackage.ownedType->select(t|t.stereotypedBy('ArchetypeVersion')).oclAsType(Classifier).general ->forAll(x|self.base\_Package.ownedType->select(t|t.stereotypedBy('ArchetypeVersion')).oclAsType(Classifier).general->includes(x))

* **archetypeIdType**

The Package must be stereotyped by ArchetypeId.

[OCL]

self.base\_Package.stereotypedBy('ArchetypeId')

##### <Stereotype> ArchetypeLibrary

**Description**

An **ArchetypeLibrary** is a **Package** that contains a collection of archetypes that constrain classes within the imported **ReferenceModel**. An **ArchetypeLibrary** must import exactly one **Package** that is stereotyped as a **ReferenceModel**.

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

**Constraints**

* **oneReferenceModel**

The must be one <<ReferenceModelImport>> PackageImport.

[OCL]

self.base\_Package.packageImport->select(stereotypedBy('ReferenceModelImport'))->size() = 1

* **onlyArchetypes**

All packaged elements must be <<Archetype>>.

[OCL]

self.base\_Package.packagedElement->forAll(p|p.stereotypedBy('Archetype'))

##### <Stereotype> ArchetypeType

**Description**

An implementation specific classification of Archetype types. In the ADL context, the ADL 2.0.5 specification describes the following types:

* archetype
* template
* template\_overlay
* operational\_template

This would be implemented with a specialization of the ArchetypeType enumeration with the four literals above. ArchetypeType does not affect the semantics of an AML implementation.

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

##### <Stereotype> ArchetypeVersion

**Description**

A set of constraints that can be applied as a predicate against instances of **Class** referenced by the constrains attribute of the containing Archetype.

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

**Direct Known Superclasses (Generalization)**

[AuthoredResource](#_13ad6987e15b787d385f0b30ff25d6c9), [ComplexObjectConstraint](#_bd9b14c4d7198d36c5a9dec9c2836b62)

**Attributes**

**• public amlVersion : String [1]**

The version of the AML specification used to define this version of the archetype.

**Constraints**

* **archetypeRoot**

This Class must have a generalization.

[OCL]

self.base\_Class.generalization->size()=1

* **ownedByArchetype**

The namespace owner must be <<Archetype>>

[OCL]

self.base\_Class.namespace.stereotypedBy('Archetype')

* **archetypeVersionIdType**

This Class must also be stereotyped by <<ArchetypeVersionId>>.

[OCL]

self.base\_Class.stereotypedBy('ArchetypeVersionId')

##### <Stereotype> AuthoredResource

**Description**

**AuthoredResource** carries a minimal set of information about the source and origin of an Archetype. Its intent is to be a "connection point" to attach additional workflow and other provenance information to the target **Archetype**.

**AuthoredResource** can be associated with **ResourceTranslations**, **Descriptions**, etc through **OpaqueBehaviors** stereotyped by **ResourceTranslation** and **ResourceDescription** respectively.

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

**Direct Known Subclasses (Specialization)**

[ArchetypeVersion](#_61831f1c446a753e3069251f603bfa37)

**Attributes**

**• public originalLanguage : EnumerationLiteral [1]**

The original language the model was authored in.

**• public isControlled : Boolean [1] = false**

A flag indicating whether the archetype is change-controlled or not can be included after the version. Archetypes that include the “controlled” flag should have the revision history section included, while those with the “uncontrolled” flag, or no flag at all, may omit the revision history. This enables archetypes to be privately edited in an early development phase without generating large revision histories of little or no value

**• public isGenerated : Boolean [1] = false**

A flag indicating whether the archetype was generated or authored. This marker is used to support the migration to differential archetype representation introduced in ADL 1.5, to enable proper representation of specialized archetypes.

**• public resourceSource : String [0..1]**

A URI that references the source document (if any) from which the original resource was derived.

**• public resourceDocumentLanguage : String [0..1]**

The language (e.g. AOM, CEM, ...) of the source of the constraints, if any.

**• public resourceDocumentSyntax : String [0..1]**

The syntax of the resource document (ADL, XML, XMI, ...)

**• public resourceSourceURI : String [0..1]**

An external identifier that uniquely identifies this Archetype. The format and structure of this identifier are determined by the rules of the *resourceDocumentLanguage* and/or *resourceDocumentSyntax*. This identifier cannot be used as an identifier within AML itself as it may not always be present. It must be preserved, however, for export to external resources.

**Constraints**

* **languageEnumeration**

originalLanguage must be in a <<Language>> Enumeration

[OCL]

self.originalLanguage.enumeration.stereotypedBy('Language')

##### <Stereotype> ComplexObjectConstraint

**Description**

A collection of constraints on an instance of an instance of a UML **Class.** A **ComplexObjectConstraint** may constrain the existence, cardinality or possible values on one or more of the constrained class attributes.

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_ad75af95f635bdf35f69d9db9b17aae2)

**Direct Known Subclasses (Specialization)**

[ArchetypeVersion](#_61831f1c446a753e3069251f603bfa37)

**Constraints**

* **singleParent**

Every constraint must specialize exactly one Class

[OCL]

self.base\_Class.generalization->size() = 1

* **allAttributeConstraints**

All owned attributes must be <<AttributeConstraint>>

[OCL]

self.base\_Class.ownedAttribute->forAll(x|x.stereotypedBy('AttributeConstraint'))

##### <Stereotype> ReferenceModelImport

**Description**

**ReferenceModelImport** is a **PackageImport** where the *importingNamespace* is an instance of an **ArchetypeLibrary** and the *importedPackage* is an instance of a **ReferenceModel**.

**Diagrams**

[Archetypes](#_1c8c4aa3906888e5c2c8895f5ed19903)

**Constraints**

* **libraryOnly**

importing namespace must be an <<ArchetypeLibrary>>

[OCL]

self.base\_PackageImport.importingNamespace.stereotypedBy('ArchetypeLibrary')

* **libraryReferenceModel**

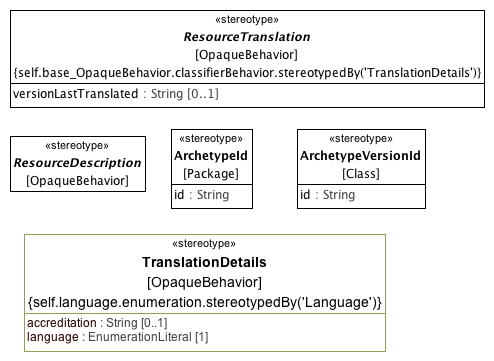
importedPackage must be <<ReferenceModel>> and importNamespace must be <<ArchetypeLibrary>>

[OCL]

self.base\_PackageImport.importedPackage.stereotypedBy('ReferenceModel') and self.base\_PackageImport.importingNamespace.stereotypedBy('ArchetypeLibrary')

#### <Package> Archetype Metadata

This clause identifies the stereotypes used to connect an **AuthoredResource** to metadata describing translations, ownership, copyright, state, etc.



**Archetype Metadata**

##### <Stereotype> ArchetypeId

**Description**

An artifact uniquely identifying an **Archetype** within a given community of use. The actual syntax and structure of the **ArchetypId** type should be established by a community of use, but all **ArchetypeId** implementations must support a String representation.

**Diagrams**

[Archetype Metadata](#_c8719d73828cfab6778459fc65cfee21)

**Attributes**

**• public id : String**

ID for the Archetype

##### <Stereotype> ArchetypeVersionId

**Description**

An **Archetype** identifier. **ArchetypeId** must uniquely identify an **Archetype** within the context of the containing **ArchetypeLibrary**. **ArchetypId** may be extended to support specific workflows and community needs, but all implementations must support a string representation.

**Diagrams**

[Archetype Metadata](#_c8719d73828cfab6778459fc65cfee21)

**Attributes**

**• public id : String**

The archetype identifier.

##### <Stereotype> ResourceDescription

**Description**

A detailed description of the source, provenance, copyright, etc of the ADL Resource.

**Diagrams**

[Archetype Metadata](#_c8719d73828cfab6778459fc65cfee21)

##### <Stereotype> ResourceTranslation

**Description**

A collection of translations for a resource.

**Diagrams**

[Archetype Metadata](#_c8719d73828cfab6778459fc65cfee21)

**Attributes**

**• public versionLastTranslated : String [0..1]**

The ArchetypeVersion identifier of the version that was last translated.

**Constraints**

* **hasOneTranslationDetails**

classifierBehavior must be <<TranslationDetails>>

[OCL]

self.base\_OpaqueBehavior.classifierBehavior.stereotypedBy('TranslationDetails')

##### <Stereotype> TranslationDetails

**Description**

A basic stereotype for adding translation details to a resource translation

**Diagrams**

[Archetype Metadata](#_c8719d73828cfab6778459fc65cfee21)

**Attributes**

**• public accreditation : String [0..1]**

**• public language : EnumerationLiteral [1]**

**Constraints**

* **languageEnumeration**

language must be contained by a <<Language>> Enumeration

[OCL]

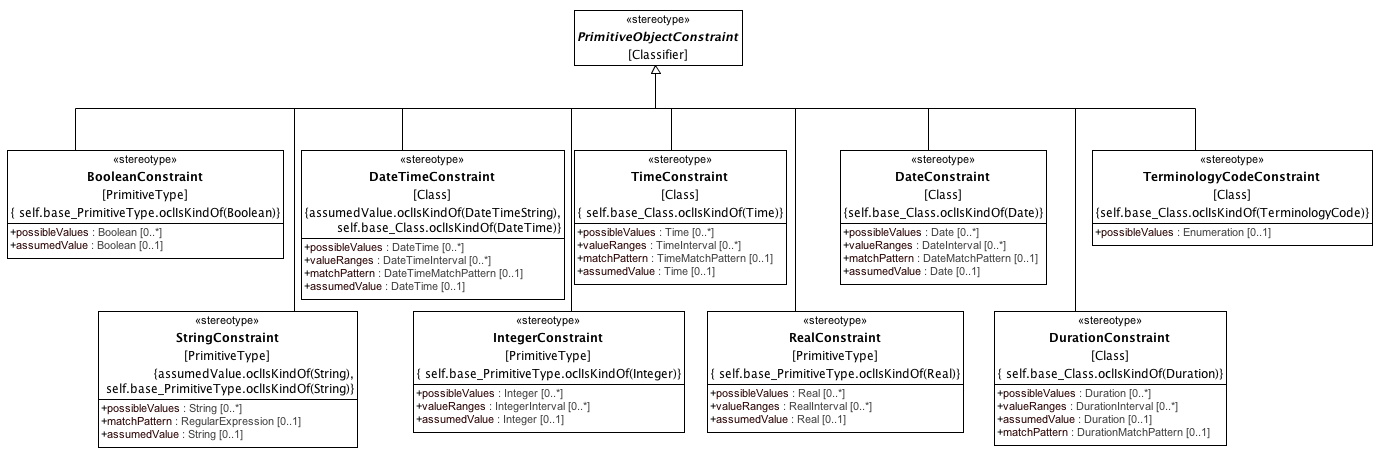
self.language.enumeration.stereotypedBy('Language')

### <Package> Data Type Constraints

This clause defines the stereotypes that are used to constrain the "primitive" or assumed types that are built in to the AML Profile.

#### <Package> Primitive Type Constraints

This clause identifies the basic "primitive" type constraints



**Primitive Type Constraints**

##### <Stereotype> BooleanConstraint

**Description**

A constraint on a property of Boolean type.

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

**• public possibleValues : Boolean [0..\*]**

A set of allowed Boolean values (e.g., True, False).

**• public assumedValue : Boolean [0..1]**

A Boolean value to be assumed to apply if no value is provided.

**Constraints**

* **constrainsBoolean**

The constraint is applied to a Boolean.

[English]

self.base\_PrimitiveType.oclIsKindOf(Boolean)

##### <Stereotype> DateConstraint

**Description**

A constraint on a property of Date type

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

**• public possibleValues :** [**Date**](#_78ee642abf9938398776ce11b2ae5595) **[0..\*]**

A set of allowed Date values (e.g., ‘2000-01-01’)

**• public valueRanges :** [**DateInterval**](#_eb65cb2938a6220d8f4a10f0d8aba136) **[0..\*]**

A set of Date value ranges, any value of which is considered valid (e.g., ‘|>= 2000-01-01|’, ’2000-01-01..2005-06-30’)

**• public matchPattern :** [**DateMatchPattern**](#_d029702cd1184b62db7fbbb690187cd5) **[0..1]**

A string pattern implying a set of valid Date values (e.g., ‘2000-??-xx’, ‘yyyy-xx-xx’)

**• public assumedValue :** [**Date**](#_78ee642abf9938398776ce11b2ae5595) **[0..1]**

A Date value to be assumed to apply if no value is provided

**Constraints**

* **constrainsDate**

Constraint must be applied to a DateType

[English]

self.base\_Class.oclIsKindOf(Date)

##### <Stereotype> DateTimeConstraint

**Description**

A constraint on a property of DateTime type.

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

**• public possibleValues :** [**DateTime**](#_7ba7e85df09d292033e869c3e8664062) **[0..\*]**

A set of allowed DateTime values (e.g., ‘2000-01-01T09:30:00’).

**• public valueRanges :** [**DateTimeInterval**](#_956e6c028c830c0453b74cbd2204109e) **[0..\*]**

A set of DateTime value ranges, any value of which is considered valid (e.g., ‘|>=2000-01-01T 09:30:00|’, ’ 2000-01-01T09:30:00.. 2000-01-01T11:30:00’).

**• public matchPattern :** [**DateTimeMatchPattern**](#_0fe09862b0893b7f5636966c344bc986) **[0..1]**

A string pattern implying a set of valid DateTime values (e.g., ‘yyyy-mm-ddT13:??:xx’, ‘yyyy-mm-ddThh:xx:xx’).

**• public assumedValue :** [**DateTime**](#_7ba7e85df09d292033e869c3e8664062) **[0..1]**

A DateTime value to be assumed to apply if no value is provided.

**Constraints**

* **assumedDateTime**

AssumeDateTime is applied to a DateTimeString.

[English]

assumedValue.oclIsKindOf(DateTimeString)

* **constrainsDateTime**

Base class is a DateTime.

[English]

self.base\_Class.oclIsKindOf(DateTime)

##### <Stereotype> DurationConstraint

**Description**

A constraint on a property of Duration type

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

**• public possibleValues :** [**Duration**](#_6f1e8a2b40ce6a6203e07d9c5daded71) **[0..\*]**

A set of allowed Duration values (e.g., ‘P5d’).

**• public valueRanges :** [**DurationInterval**](#_3d4fd0ce80d2a2e88d2f42b3cb7dbec5) **[0..\*]**

A set of Duration value ranges, any value of which is considered valid (e.g., ‘|>= P5d|’, ’P5d..P8d’).

**• public assumedValue :** [**Duration**](#_6f1e8a2b40ce6a6203e07d9c5daded71) **[0..1]**

A Duration value to be assumed to apply if no value is provided.

**• public matchPattern :** [**DurationMatchPattern**](#_f432daf955c8241a03bff56a5a7d8e66) **[0..1]**

A string pattern that implies a set of valid Duration values (e.g., ‘Pd’, ‘PThm’).

**Constraints**

* **constrainsDuration**

The constrain is applied to a Duration.

[English]

self.base\_Class.oclIsKindOf(Duration)

##### <Stereotype> IntegerConstraint

**Description**

A constraint on a property of Integer type.

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

**• public possibleValues : Integer [0..\*]**

A set of allowed Integer values (e.g., 2, 5).

**• public valueRanges :** [**IntegerInterval**](#_86618450de28d822bd6b57b67a32ab2b) **[0..\*]**

A set of Integer value ranges, any value of which is considered valid.

**• public assumedValue : Integer [0..1]**

An Integer value to be assumed to apply if no value is provided.

**Constraints**

* **assumedInteger**

assumedValue is an integer.

[English]

assumedValue.oclIsKindOf(Integer)

* **constrainsInteger**

The constraint is applied to an Integer.

[English]

self.base\_PrimitiveType.oclIsKindOf(Integer)

##### <Stereotype> PrimitiveObjectConstraint

**Description**

A constraint on a UML or AML primitive type

The abstract primitive object constraint has no properties, but those of its specializations follow common patterns. (It also seems to have no relationships, e.g., composition to complex constraint.)

These patterns include properties defining allowed values via explicit enumeration, range definition, and matching patterns. These allowed value properties are to be treated as permissive rather than restrictive, with the result that if more than one of these properties is defined, the set of valid values for an instance is the union of values so defined. Inclusion and exclusion of interval end points in the valueRanges properties is governed by the interval data types.

Another pattern comprises the ‘assumed value’ properties. In an archetype containing optional data elements, assumed values can be used to specify values that can reliably be inferred by receivers. For example, an archetype for ‘blood pressure measurement’ might include an optional data element describing the patient position, with choices ‘lying’, ‘sitting’ and ‘standing’. Since this element is optional, conformant data could be created that does not contain it. The ‘assumed value’ property allows a value to be explicitly stated so that all users/systems know what value to assume when optional items are not included in the data, in cases where such an assumption is appropriate. In the ‘blood pressure measurement’ example, one might define such a value as ‘sitting’—the most common value—with the understanding that such an assumption makes it important for instances in which the value is different, or not known, to say so, notwithstanding that the value is technically optional.

Assumed values are definable on any primitive type. If the archetype does not define an assumed value, no reliable assumption can be made by the receiver of an archetype instance concerning the value of the property.

Date and Time constraints follow the string syntax of the ISO 8601 Representation of dates and times.

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_ad75af95f635bdf35f69d9db9b17aae2)

**Direct Known Subclasses (Specialization)**

[BooleanConstraint](#_40ee863e6fd02692437dae1d81ba12de), [DateConstraint](#_ff8930d68c378c02c221704764a5c9d4), [DateTimeConstraint](#_7dde1322feeec9c32a95df44c39d8e48), [DurationConstraint](#_384c080719f5bd1b45eae1293215b466), [IntegerConstraint](#_2219fb1dcaf5f26a0ed07de77d69cd5e), [RealConstraint](#_b921bc493035fb4e067213114372e254), [StringConstraint](#_982033c222702fafb1d4d3ed7b399317), [TerminologyCodeConstraint](#_ef76317db67a290898f39af3c51eee9c), [TimeConstraint](#_d8c772ca77efc45bee8711f1de17afc0)

##### <Stereotype> RealConstraint

**Description**

A constraint on a property of Real type.

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

**• public possibleValues : Real [0..\*]**

A set of allowed Real values (e.g., 37.2, 100.265).

**• public valueRanges :** [**RealInterval**](#_d4f7314ff920dd15ee0e834cfbd4e6f2) **[0..\*]**

A set of Real value ranges, any value of which is considered valid.

**• public assumedValue : Real [0..1]**

Real value to be assumed to apply if no value is provided.

**Constraints**

* **constrainsReal**

Constraint is applied to a real.

[English]

self.base\_PrimitiveType.oclIsKindOf(Real)

##### <Stereotype> StringConstraint

**Description**

A constraint on a property of String type. Most constrainable text values are best constrained by language-independent terminology constraints, but this tactic is supported.

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

**• public possibleValues : String [0..\*]**

A set of allowed String values (e.g., ‘Robert’)

**• public matchPattern :** [**RegularExpression**](#_7dba62a18c5b1da31d9d2e3df675a297) **[0..1]**

A Perl regular expression defining allowed string constructions.

**• public assumedValue : String [0..1]**

A String value to be assumed to apply if no value is provided.

**Constraints**

* **assumedString**

The assumed value must be a String

[English]

assumedValue.oclIsKindOf(String)

* **constrainsString**

The constraint is applied to a String.

[English]

self.base\_PrimitiveType.oclIsKindOf(String)

##### <Stereotype> TerminologyCodeConstraint

**Description**

A **TerminologyCodeConstraint** is a constraint on the possible values of the AML **TerminologyCode** type. It constrains the possible values of a terminology code by referencing an **Enumeration** whose member **EnumerationLiterals** represent the possible values for the constrained **TerminologyCode.** *possibleValues* can reference a simple **Enumeration**, a **ScopedIdentifier** or an **EnumeratedValueDomain**. The referenced **Enumeration** or **EnumeratedValueDomain** may be extended by one or more **EnumerationConstraints** or **EnumeratedValueDomainConstraints** respectively.

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

**• public possibleValues : Enumeration [0..1]**

An **Enumeration** whose (possibly constrained) instances represent the possible values for the constrained **TerminologyCode**. If absent, the TerminologyCode values are not constrained.

**Constraints**

* **mustBeEnumeratedValueDomain**

valueSet, if defined, must be an <<EnumeratedValueDomain>>

[OCL]

not(self.valueSet.oclIsUndefined()) implies self.valueSet.stereotypedBy('EnumeratedValueDomain')

* **constrainsConceptReference**

This Class must be a <<ConceptReference>>

[OCL]

self.base\_Class.oclIsKindOf(TerminologyCode)

##### <Stereotype> TimeConstraint

**Description**

A constraint on a property of Time type.

**Diagrams**

[Primitive Type Constraints](#_ae5d1776f07899a7af4725351a338b35)

**Direct Known Superclasses (Generalization)**

[PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Attributes**

**• public possibleValues :** [**Time**](#_cba83b2c77167c96697f3caaa1886f5c) **[0..\*]**

A set of allowed Time values (e.g., ‘09:30:00’).

**• public valueRanges :** [**TimeInterval**](#_2db4f3574d756c0312a2a6559efd3ad9) **[0..\*]**

A set of Time value ranges, any value of which is considered valid (e.g., ‘|>= 09:30:00|’, ’09:30:00..11:30:00’).

**• public matchPattern :** [**TimeMatchPattern**](#_4b701856b8ef798f1dd89caa4d9efe11) **[0..1]**

A string pattern implying a set of valid Time values (e.g., ‘13:??:xx’, ‘hh:xx:xx’).

**• public assumedValue :** [**Time**](#_cba83b2c77167c96697f3caaa1886f5c) **[0..1]**

A Time value to be assumed to apply if no value is provided.

**Constraints**

* **constrainsTime**

The constraint is applied to a Time type.

[English]

self.base\_Class.oclIsKindOf(Time)

#### <Package> Date and Time Match Types

This clause describes the various matching types derived from ISO 8601 *Data elements and interchange formats – Information interchange – Representation of dates and times*



**Date and Time Match Types**

#### <Package> Intervals

This section contains the definition of the intervals used in the primitive type constraints.



**Intervals**

##### <Class> DateInterval

**Description**

A set of contiguous Dates.

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Superclasses (Generalization)**

[Interval](#_3ae971b2839139d9692e47ec472148b6)

**Attributes**

**• public lower :** [**Date**](#_78ee642abf9938398776ce11b2ae5595) **[0..1]**

The earliest Date that may be included in the interval.

**• public upper :** [**Date**](#_78ee642abf9938398776ce11b2ae5595) **[0..1]**

The latest Date that may be included in the interval.

##### <Class> DateTimeInterval

**Description**

A set of contiguous DateTimes.

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Superclasses (Generalization)**

[Interval](#_3ae971b2839139d9692e47ec472148b6)

**Attributes**

**• public lower :** [**DateTime**](#_7ba7e85df09d292033e869c3e8664062) **[0..1]**

The earliest DateTime that may be included in the interval.

**• public upper :** [**DateTime**](#_7ba7e85df09d292033e869c3e8664062) **[0..1]**

The latest DateTime that may be included in the interval.

##### <Class> DurationInterval

**Description**

A set of contiguous Duration magnitudes.

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Superclasses (Generalization)**

[Interval](#_3ae971b2839139d9692e47ec472148b6)

**Attributes**

**• public lower :** [**Duration**](#_6f1e8a2b40ce6a6203e07d9c5daded71) **[0..1]**

The shortest Duration that may be included in the interval.

**• public upper : Duration [0..1]**

The longest Duration that may be included in the interval.

##### <Class> IntegerInterval

**Description**

A set of contiguous integers.

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Superclasses (Generalization)**

[Interval](#_3ae971b2839139d9692e47ec472148b6)

**Attributes**

**• public lower : Integer [0..1]**

The smallest integer that may be included in the interval.

**• public upper : Integer [0..1]**

The largest integer that may be included in the interval.

##### <Class> Interval

**Description**

A set of contiguous values bounded by upper and lower limits. Inclusion of the upper and lower limits is governed by the upperIncluded and lowerIncluded properties.

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Subclasses (Specialization)**

[DateInterval](#_eb65cb2938a6220d8f4a10f0d8aba136), [DateTimeInterval](#_956e6c028c830c0453b74cbd2204109e), [DurationInterval](#_3d4fd0ce80d2a2e88d2f42b3cb7dbec5), [IntegerInterval](#_86618450de28d822bd6b57b67a32ab2b), [RealInterval](#_d4f7314ff920dd15ee0e834cfbd4e6f2), [TimeInterval](#_2db4f3574d756c0312a2a6559efd3ad9)

**Attributes**

**• public lowerIncluded : Boolean [1]**

A Boolean value indicating whether the value asserted to be the lower limit is included in the set it bounds.

**• public upperIncluded : Boolean [1]**

A Boolean value indicating whether the value asserted to be the upper limit is included in the set it bounds.

##### <Class> RealInterval

**Description**

A set of contiguous Real numbers

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Superclasses (Generalization)**

[Interval](#_3ae971b2839139d9692e47ec472148b6)

**Attributes**

**• public lower : Real [0..1]**

The smallest Real that may be included in the interval

**• public upper : Real [0..1]**

The largest Real that may be included in the interval

##### <Class> TimeInterval

**Description**

A set of contiguous Times.

**Diagrams**

[Intervals](#_8566828d89ca01e8919a51d19aa6cd8b)

**Direct Known Superclasses (Generalization)**

[Interval](#_3ae971b2839139d9692e47ec472148b6)

**Attributes**

**• public lower :** [**Time**](#_cba83b2c77167c96697f3caaa1886f5c) **[0..1]**

The earliest Time that may be included in the interval.

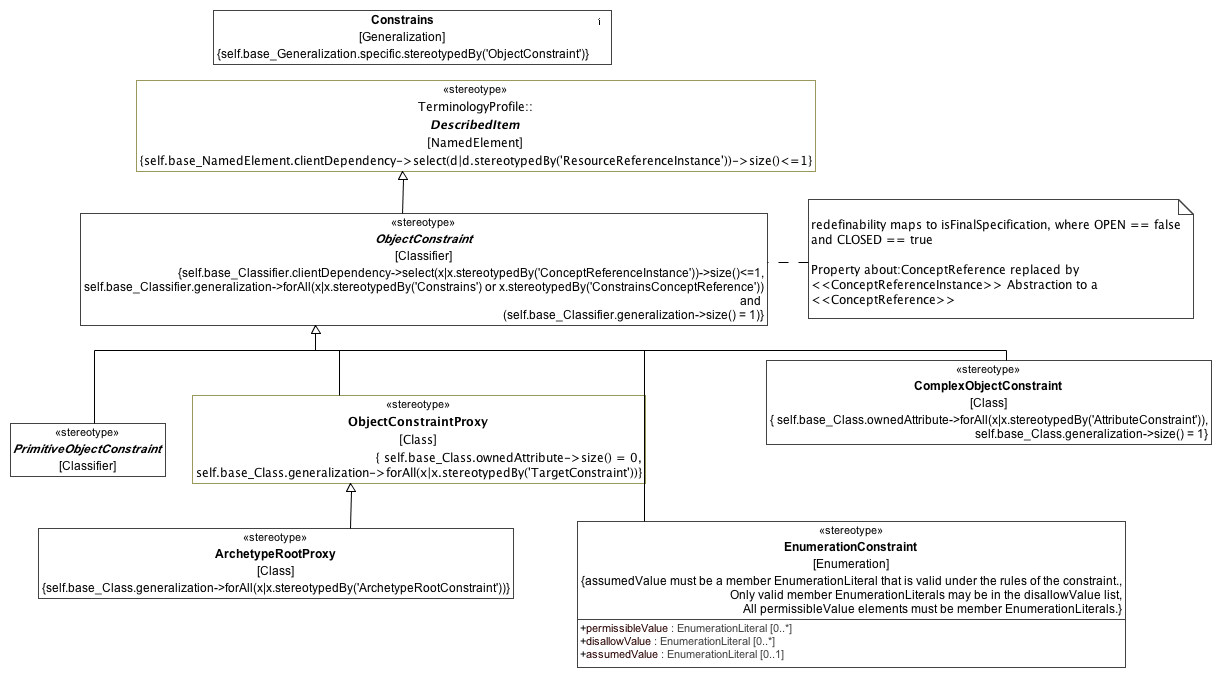
**• public upper :** [**Time**](#_cba83b2c77167c96697f3caaa1886f5c) **[0..1]**

The latest Time that may be included in the interval.

### <Package> Object and Property Constraints

#### <Package> Object Constraints

This clause describes the various forms of constraint that can be applied to the UML **Class.**



**Object Constraints**

##### <Stereotype> ArchetypeRootProxy

**Description**

A reference to a target Archetype that constrains the same subclass or descendant thereof as that referenced by the constrains attribute of the NamedObjectConstraint itself.

**ArchetypeRootProxy** implements the use\_archetype construct in ADL. Note that an ArchetypeRootProxy may or may not reference a specific version of the target Archetype. If the targetVersion attribute is absent, the ArchetypeRootProxy is understood to reference the ArchetypeVersion that is considered to be "current" at whatever time that the proxy is referenced.

**Diagrams**

[Object Constraints](#_f139650e10793c0005c93b604992495c)

**Direct Known Superclasses (Generalization)**

[ObjectConstraintProxy](#_c8ae60f7f44b70cf5dce7db03aa6ac1e)

**Constraints**

* **redefinesConstrains**

Any generalization must be <<ArchetypeRootConstraint>>

[OCL]

self.base\_Class.generalization->forAll(x|x.stereotypedBy('ArchetypeRootConstraint'))

##### <Stereotype> Constrains

**Description**

Constraining relationship between a generalization and an **ObjectConstraint**.

**Diagrams**

[Object Constraints](#_f139650e10793c0005c93b604992495c)

**Direct Known Subclasses (Specialization)**

[TargetConstraint](#_6de94cd3c6736f017766fe61020a5a13)

**Constraints**

* **specificObjectConstraint**

Specific must be an <<ObjectConstraint>>

[OCL]

self.base\_Generalization.specific.stereotypedBy('ObjectConstraint')

##### <Stereotype> EnumerationConstraint

**Description**

A constraint on a possible values in a UML **Enumeration**. **EnumerationConstraint** allows a modeler to:

1. List which **EnumerationLiterals** owned by the **Enumeration** are permitted.
2. List which **EnumerationLiterals** owned by the **Enumeration** are not permitted.
3. Add an **EnumerationLiteral** that is the assumed value for this instance.
4. Add additional **EnumerationLiterals**, extending the parent **Enumeration**.

**Diagrams**

[Object Constraints](#_f139650e10793c0005c93b604992495c)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_ad75af95f635bdf35f69d9db9b17aae2)

**Direct Known Subclasses (Specialization)**

[EnumeratedValueDomainConstraint](#_7b4688dbd3826f33c726c87847ae4a72)

**Attributes**

**• public permissibleValue : EnumerationLiteral [0..\*]**

The **Enumeration** *members* allowed in this constrained instance. If not empty, only the **EnumerationLiterals** in this list are valid values. Only **EnumerationLiterals** are valid for the general **Enumeration** or **EnumerationConstraint** can be listed in *permissibleValue*.

**• public disallowValue : EnumerationLiteral [0..\*]**

The **Enumeration** *members* not allowed in this constrained instance. An **EnumerationLiteral** in the disallowValue takes precedence over a *member* in the *permissibleValue* -- if it appears in both places it is not allowed.

**• public assumedValue : EnumerationLiteral [0..1]**

The value to be assumed if it is absent in a data instance. *assumedValue* must be a valid value for the constrained enumeration -- if there is a non-empty list of *permissibleValues*, it must appear in the list and it may not appear in the *disallowValue* list.

**Constraints**

* **validPermissibleValue**

[English]

All permissibleValue elements must be member EnumerationLiterals.

* **validDisallowValue**

[English]

Only valid member EnumerationLiterals may be in the disallowValue list

* **validDefaultValue**

[English]

assumedValue must be a member EnumerationLiteral that is valid under the rules of the constraint.

##### <Stereotype> ObjectConstraint

**Description**

An **ObjectConstraint** is a specialization of a **Classifier** where there are no new *ownedAttributes* that do not *subset* or *redefine* an inherited *attribute*.

The **Classifier** stereotyped by an **ObjectConstraint** must participate in exactly one **Generalization** relationship which must be stereotyped by the **Constrains** stereotype.

**Diagrams**

[Object Constraints](#_f139650e10793c0005c93b604992495c)

**Direct Known Superclasses (Generalization)**

[DescribedItem](#_d45578f848d02aad83980903e5bde7d1)

**Direct Known Subclasses (Specialization)**

[ComplexObjectConstraint](#_bd9b14c4d7198d36c5a9dec9c2836b62), [EnumerationConstraint](#_1bc74c3698f61990aff3aec96088f0a9), [ObjectConstraintProxy](#_c8ae60f7f44b70cf5dce7db03aa6ac1e), [PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

**Constraints**

* **redefinesGeneralization**

This Classifier must have exactly one generalization, and that is a <<Constrains>> or <<ConstrainsConceptReference>> generalization.

[OCL]

self.base\_Classifier.generalization->forAll(x|x.stereotypedBy('Constrains') or x.stereotypedBy('ConstrainsConceptReference')) and (self.base\_Classifier.generalization->size() = 1)

* **about**

At most one "about" reference via <<ConceptReferenceInstance>> Abstraction

[OCL]

self.base\_Classifier.clientDependency->select(x|x.stereotypedBy('ConceptReferenceInstance'))->size()<=1

##### <Stereotype> ObjectConstraintProxy

**Description**

An *ObjectConstraintProxy* asserts a constraint defined elsewhere in the archetype is to apply to the **Classifier** instance referenced by the proxy *constrains* attribute.

**Diagrams**

[Object Constraints](#_f139650e10793c0005c93b604992495c)

**Direct Known Superclasses (Generalization)**

[ObjectConstraint](#_ad75af95f635bdf35f69d9db9b17aae2)

**Direct Known Subclasses (Specialization)**

[ArchetypeRootProxy](#_f5f73ce565f73d8b4808997e54e8e698)

**Constraints**

* **redefinesConstrains**

All generals are <<TargetConstraint>>

[OCL]

self.base\_Class.generalization->forAll(x|x.stereotypedBy('TargetConstraint'))

* **noNewAttributes**

No attributes allowed on this Class

[OCL]

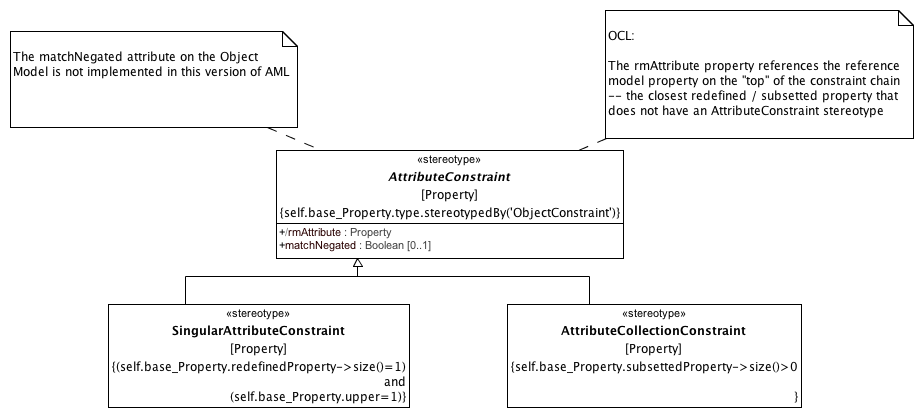
self.base\_Class.ownedAttribute->size() = 0

**Known other Stereotypes**

[ComplexObjectConstraint](#_bd9b14c4d7198d36c5a9dec9c2836b62), [DescribedItem](#_d45578f848d02aad83980903e5bde7d1), [PrimitiveObjectConstraint](#_c72b6d9c8a46b96f02fdfefe3b8b0568)

#### <Package> Attribute Constraints

This clause defines the constraints that can be applied to a UML **Property**.



**Attribute Constraints**

##### <Stereotype> AttributeCollectionConstraint

**Description**

A constraint on an attribute with an upper multiplicity of greater than one. An AttributeCollectionConstraint may assert an ordering on the member AttributeCollectionMembers and may assert the minimum and maximum number of instances that may occur for the referenced attribute.

**Diagrams**

[Attribute Constraints](#_88dce20413dd8833c4e90da9fe432855)

**Direct Known Superclasses (Generalization)**

[AttributeConstraint](#_1bf8a3231ae21af2dec84426b5618c38)

**Constraints**

* **mustHaveSubsettedProperty**

Must have at least one subsettedProperty.

[OCL]

self.base\_Property.subsettedProperty->size()>0

##### <Stereotype> AttributeConstraint

**Description**

In any information model, attributes are either single-valued or multiply-valued, i.e. of a generic container. Both have existence, while multiply-valued attributes also have cardinality

**Diagrams**

[Attribute Constraints](#_88dce20413dd8833c4e90da9fe432855)

**Direct Known Subclasses (Specialization)**

[AttributeCollectionConstraint](#_4bc615eb2707782fc8254702b7e0b435), [SingularAttributeConstraint](#_2d1a6d8b2806092b50ec3fd4cd2db35b)

**Attributes**

**• public rmAttribute : Property**

**• public matchNegated : Boolean [0..1]**

**Constraints**

* **objectConstraintTarget**

The Property type must be an <<ObjectConstraint>>

[OCL]

self.base\_Property.type.stereotypedBy('ObjectConstraint')

##### <Stereotype> SingularAttributeConstraint

**Description**

An AttributeConstraint that constrains the possible values for a singular property -- a property with an upper multiplicity of 1.

**Diagrams**

[Attribute Constraints](#_88dce20413dd8833c4e90da9fe432855)

**Direct Known Superclasses (Generalization)**

[AttributeConstraint](#_1bf8a3231ae21af2dec84426b5618c38)

**Constraints**

* **mustHaveRedefinedProperty**

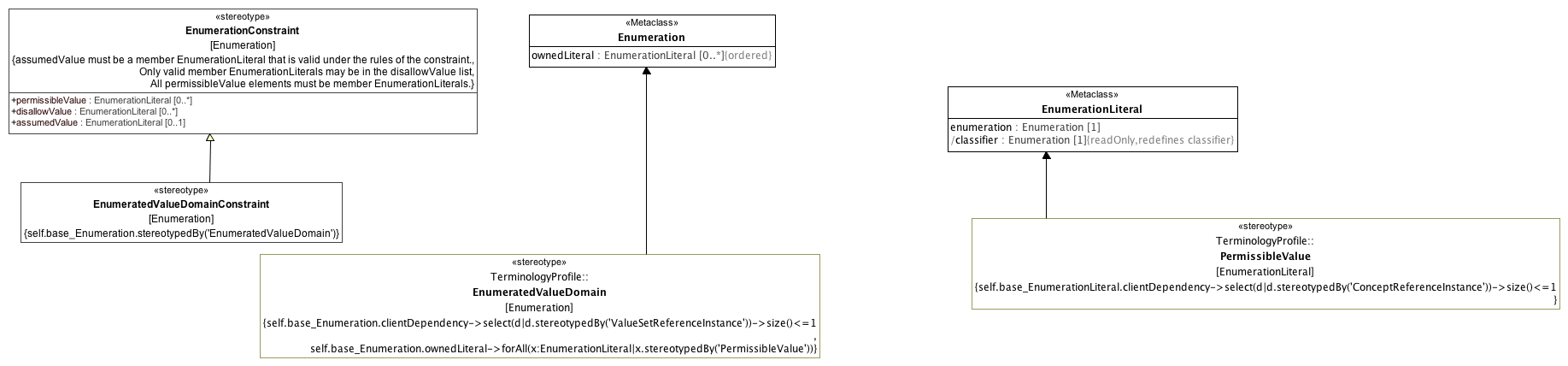
Property must have exactly one redefined Property and an upper bound of 1.

[OCL]

(self.base\_Property.redefinedProperty->size()=1) and (self.base\_Property.upper=1)

#### <Package> Enumeration Constraints

This clause defines the constraints that can be applied to UML **Enumeration** and its derivatives.



**Enumeration Constraints**

##### <Class> Enumeration

**Diagrams**

[Enumeration Constraints](#_4049ef2e39f1ca7b7abb65f10409d85f)

**Direct Known Superclasses (Generalization)**

DataType

**Attributes**

**• public ownedLiteral : EnumerationLiteral [0..\*]**

The ordered set of literals owned by this Enumeration.

##### <Class> EnumerationLiteral

**Diagrams**

[Enumeration Constraints](#_4049ef2e39f1ca7b7abb65f10409d85f)

**Direct Known Superclasses (Generalization)**

InstanceSpecification

**Attributes**

**• public enumeration : Enumeration [1]**

The Enumeration that this EnumerationLiteral is a member of.

**• public classifier : Enumeration [1]**

##### <Stereotype> EnumeratedValueDomainConstraint

**Description**

A constraint on a **EnumeratedValueDomain** that represents a discrete set of possible values for a particular field or data element.

**Diagrams**

[Enumeration Constraints](#_4049ef2e39f1ca7b7abb65f10409d85f)

**Direct Known Superclasses (Generalization)**

[EnumerationConstraint](#_1bc74c3698f61990aff3aec96088f0a9)

**Constraints**

* **constrainsEVD**

This Enumeration is an <<EnumeratedValueDomain>>

[OCL]

self.base\_Enumeration.stereotypedBy('EnumeratedValueDomain')

**Known other Stereotypes**

[EnumeratedValueDomain](#_c7f411daaf64f83e013bec437cb8f30a), [EnumerationConstraint](#_1bc74c3698f61990aff3aec96088f0a9), [PermissibleValue](#_5bb7ce8128b60ee5eb2ca275444e9692)

#### <Package> Constraint Proxies

This clause defines mechanisms for reusing and "importing" constraints.



**Constraint Proxies**

##### <Stereotype> ArchetypeRootConstraint

**Description**

Connects an <<ArchetypeVersion>> to an <<ArchetypeRootProxy>> to be constrained.

**Diagrams**

[Constraint Proxies](#_17547bff44c4353bd3a454a0c3c7e577)

**Direct Known Superclasses (Generalization)**

[TargetConstraint](#_6de94cd3c6736f017766fe61020a5a13)

**Constraints**

* **specificIsRootProxy**

Specific must be an <<ArchetypeRootProxy>>

[OCL]

self.base\_Generalization.specific.stereotypedBy('ArchetypeRootProxy')

* **generalIsArchetypeVersion**

The general must be an <<ArchetypeVersion>>

[OCL]

self.base\_Generalization.general.stereotypedBy('ArchetypeVersion')

##### <Stereotype> TargetConstraint

**Description**

Connects an <<ObjectConstraint>> to an <<ObjectConstraintProxy>>

**Diagrams**

[Constraint Proxies](#_17547bff44c4353bd3a454a0c3c7e577)

**Direct Known Superclasses (Generalization)**

[Constrains](#_f91b532413834ad1de94d0b0af526f5b)

**Direct Known Subclasses (Specialization)**

[ArchetypeRootConstraint](#_6eafe370f24f01390e7ab79d6568ea94)

**Constraints**

* **specificIsProxy**

Specific must be an <<ObjectConstraintProxy>>

[OCL]

self.base\_Generalization.specific.stereotypedBy('ObjectConstraintProxy')

* **generalIsObjectConstraint**

General must be <<ObjectConstraint>>

[OCL]

self.base\_Generalization.general.stereotypedBy('ObjectConstraint')

**Known other Stereotypes**

[ArchetypeRootProxy](#_f5f73ce565f73d8b4808997e54e8e698), [Constrains](#_f91b532413834ad1de94d0b0af526f5b), [ObjectConstraint](#_ad75af95f635bdf35f69d9db9b17aae2), [ObjectConstraintProxy](#_c8ae60f7f44b70cf5dce7db03aa6ac1e)

### <Package> Terminology Constraints

This clause defines the profiles used for constraining Enumerations, EnumeratedValueDomains and TerminologyCodes.



**TerminologyConstraints**

**Known other Stereotypes**

[EnumeratedValueDomainConstraint](#_7b4688dbd3826f33c726c87847ae4a72), [EnumerationConstraint](#_1bc74c3698f61990aff3aec96088f0a9)