Initiative for Modeling the Legal Analysis Methodology

LAM-SKOS-AP: an application profile for the Legal Analisys Methodology description

Deliverable: WP 1.2

Author: Eugeniu Costetchi

Date: 30 September 2019

Version 1.0

Disclaimer

Contents

1 Introduction

This document defines the application profile for Legal Analisys Methodology assets managed using VocBench3[1]. VocBench3 is a web-based, multilingual, collaborative development platform for managing OWL ontologies, SKOS(XL) thesauri and generic RDF datasets. SKOS is a common data model for sharing and linking knowledge organization systems via the Web. The SKOS data model provides a standard, low-cost migration path for porting existing knowledge organization systems to the Semantic Web. SKOS also provides a lightweight, intuitive language for developing and sharing new knowledge organization systems. It may be used on its own, or in combination with formal knowledge representation languages such as the Web Ontology language (OWL)[2].

An Application Profile (AP) is a specification that re-uses terms from one or more base standards, adding more specificity by identifying mandatory, recommended and optional elements to be used for a particular application, as well as recommendations for controlled vocabularies to be used.

The Application Profile specified in this document is based on the specification of the Simple Knowledge Organization System (SKOS). SKOS is an RDF[3] vocabulary designed to facilitate interoperability between controlled vocabularies published on the Web as Linked Open Data. Additional classes and properties from other well-known vocabularies are re-used where necessary.

The work does not cover implementation issues like mechanisms to edit or publish metadata assets and expected behaviour of systems implementing the Application Profile other than what is defined in the Conformance Statement.

The Application Profile is intended to facilitate controlled vocabularies exchange and therefore the classes and properties defined in this document are only relevant for the controlled vocabularies to be exchanged; there are no requirements for communicating systems to implement specific technical environments. The only requirement is that the systems can export and import data in RDF in conformance with this Application Profile.

2 Context

This section briefly explains the general context and the processes using and impacted by LAM. This description reflects the situation to date and may serve as a basis for deriving improvements.

EUR-Lex Legal Analysis Methodology (LAM) presents and describes the use of metadata

[1]Stellato, A., Fiorelli, M., Turbati, A., Lorenzetti, T., Van Gemert, W.,Dechandon, D., Laaboudi-Spoiden, C., Gerencser, A., Waniart, A.,Costetchi, E., and Keizer, J. (forthcoming). VocBench 3: a Collab-orative Semantic Web Editor for Ontologies, Thesauri and Lexicons.Semantic Web journal. link

[2]Bechhofer, S., & Miles, A. (2009). SKOS Simple Knowledge Organization System Reference. `https://www.w3.org/TR/skos-reference/`

[3]Wood, D., Lanthaler, M., & Cyganiak, R. (2014). RDF 1.1 Concepts and Abstract Syntax.

elements that are relevant for the legal and documentary analysis of the EUR-Lex website's content.

The metadata elements employed in LAM are taken from the Common Data Model (CDM) of the CELLAR repository of the Publications Office.

CELLAR is an electronic database which contains the documents and their related metadata diffused on one of the websites of the Publications Office. The CDM is an ontology that describes the concepts and relationships (properties/elements) that can exist for the data stored in the CELLAR.

LAM documentation contains descriptions of classes of legal documents and a selection of metadata suitable for describing each document class. LAM aims at facilitating the understanding and the use of relevant CDM properties.

LAM gives some basic Definition for the metadata elements, determines their cardinality and lists the related properties. It also gives some methodological rules concerning the use of the elements in different contexts during the legal analysis. It also describes which kind of data has to be used when filling in the metadata elements. If a metadata element has to be completed with a value coming from a controlled vocabulary, it is indicated. If there is no indication, it means that the metadata element can be filled in with free text.

The process involving LAM starts with the legal documents (LD) published in CELLAR. Then the legal analysis team receives an XML Notice and access to the HTML content. At this stage, the notice contains a minimal set of metadata which may or may not be correct.

The goal is to (a) verify the correctness of the existing metadata, (b) classify the document according to LAM methodology (c) enrich the document with the corresponding legal metadata.

The document classification, for instance, is performed by considering the structure of the document title (i.e. presence/absence of keywords), structure of the CELEX number (if present), author and other metadata. For example, the title which contains string "communication of the commission" required that the author is the European Commission (EC) and is classified as a certain LAM document class.

Enrichment of the corresponding legal metadata is manually performed under the guidance of the LAM documentation.

The diagram below depicts operations performed using the published Legal Document and the XML notice & metadata. Document classification, validation of publishing metadata and enrichment with legal metadata are performed by an external contractor. The legal analysis team further performs a quality control of the legal document and XML notice and metadata and ensure the dissemination of the metadata dissemination on EuroLex website and other channels.

In addition, the legal analysis team is the LAM owner and main editor. This role implies collecting feedback from various stakeholders and partners; initiating projects to

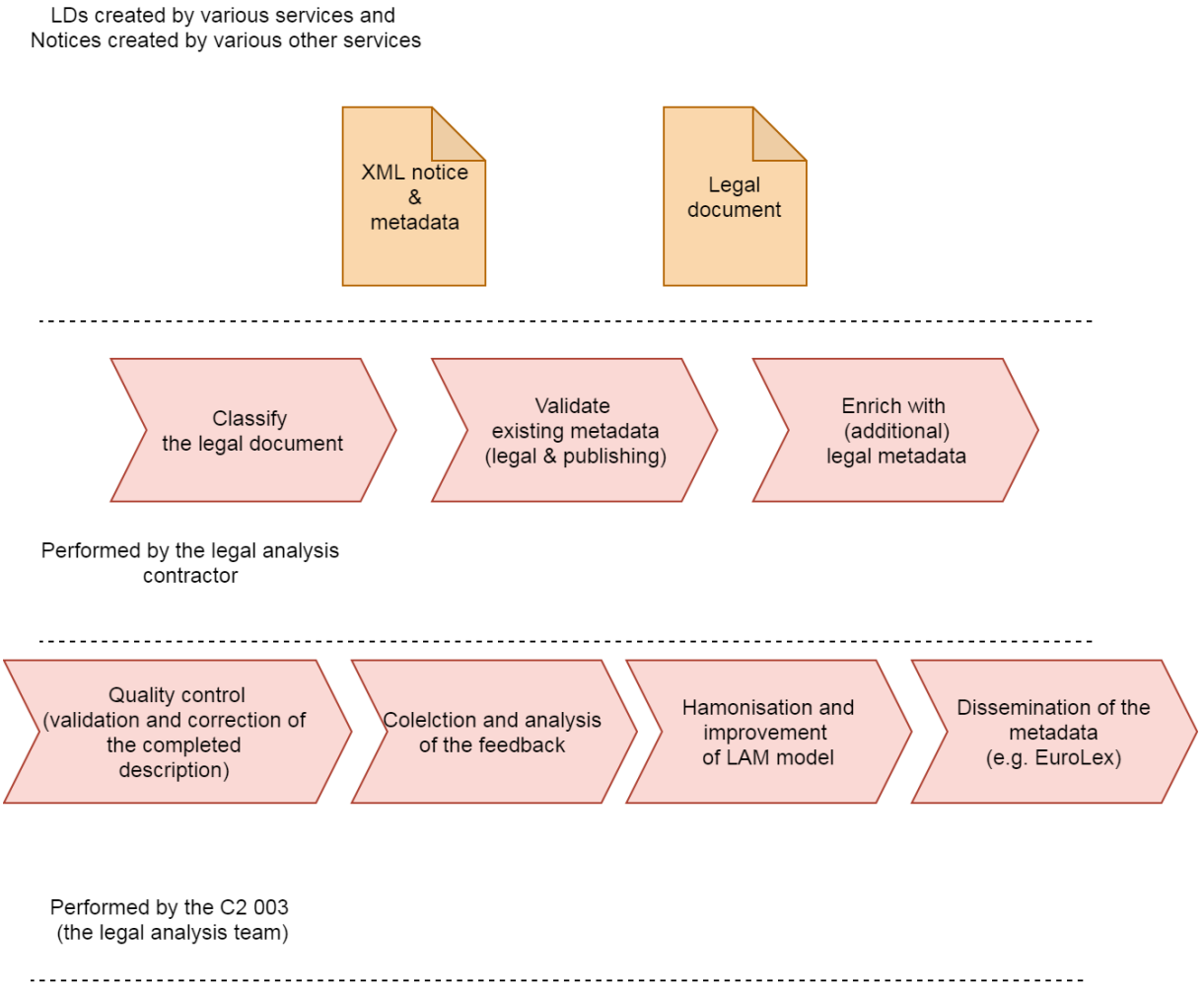

Figure 1: Digital assets and operations related to legal analysis methodology

harmonise and improve the quality of LAM itself.

3 Terminology used

In the following sections, classes and properties are grouped under headings ”mandatory”, ”recommended” and ”optional”. These terms have the following meaning.

- Mandatory class: a receiver of data MUST be able to process information about instances of the class; a sender of data MUST provide information about instances of the class.
- Recommended class: a receiver of data MUST be able to process information about instances of the class; a sender of data MUST provide information about instances of the class, if it is available.

- Optional class: a receiver MUST be able to process information about instances of the class; a sender MAY provide the information but is not obliged to do so.
- Mandatory property: a receiver MUST be able to process the information for that property; a sender MUST provide the information for that property.
- Recommended property: a receiver MUST be able to process the information for that property; a sender SHOULD provide the information for that property if it is available.
- Optional property: a receiver MUST be able to process the information for that property; a sender MAY provide the information for that property but is not obliged to do so.

The meaning of the terms MUST, MUST NOT, SHOULD and MAY in this section and in the following sections are as defined in RFC 2119[4].

In the given context, the term "processing" means that receivers must accept incoming data and transparently provide these data to applications and services. It does neither imply nor prescribe what applications and services finally do with the data (parse, convert, store, make searchable, display to users, etc.).

4 Used vocabularies

This application profile reuses classes and properties from various existing specifications. Classes and properties specified in the next sections have been taken from the following namespaces.

Ontology	Prefix	URI
Simple Knowledge Organization System	skos	http://www.w3.org/2004/02/skos/core#
Shapes Constraint Language (SHACL)	sh	http://www.w3.org/ns/shacl#
DCMI Metadata Terms	dct	http://purl.org/dc/terms/
Publications Office Extensions Ontology	euvoc	http://publications.europa.eu/ontology/euvoc
OWL 2 Web Ontology Language	owl	http://www.w3.org/2002/07/owl#
RDF Schema Vocabulary	rdfs	http://www.w3.org/2000/01/rdf-schema#
Resource Description Framework	rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
XML Schema Definition	xsd	http://www.w3.org/2001/XMLSchema#

Table 1: List of namespace definitions

[4]IETF. RFC 2119. Key words for use in RFCs to Indicate Requirement Levels. `http://www.ietf.org/rfc/rfc2119.txt`

5 Graphical representation

The graphical representation of the Application Profile is provided in the form of an UML class diagram[5] and is depicted in the figure below. The boxes represent classes while the arrow connections represent properties establishing relations to other classes. The attributes inside boxes represent properties providing either literal data values or relation to other classes that omitted from the diagram. Both, arrows and attributes, are labelled with the property names prefixed with the namespace where they are defined.

The cardinality specifications on the connector (_ .. _) arrows and next to the attributes, marked in square brackets ([ ]) represent constraints on how the property may be employed on the class instances and has a normative meaning. The first number means minimum cardinality constraint and the second means maximum cardinality constraint. The minimum cardinality constraint is zero (0 .. _) for optional properties and one (1 .. _) for mandatory properties. The maximum cardinality constraint is usually unspecified (_ .. *) or limited to one (_ .. 1) and has no normative value in this application profile. If the cardinality is not specified then the implied meaning is exactly one (1..1).

The stereotypes, marked in the double angle brackets (<< >>), used in the diagram are based on RDFS model. They have an indicative and not a semantic value. The "mandatory", "optional" and "recommended" stereotypes are normative and have the meaning defined in the terminology section .

In curly brackets ({ }) are provided the value constraints on the property values. These constraints refer to controlled list of values that should meet a minimum set of requirements.

6 Application profile

This section presents the application profile describing each class and data type. The class descriptions also comprise a list of relevant properties and the configuration in which they may appear in the data.

6.1 lam:DocumentProperty

This class aims at describing properties used for describing legal documents. It aims at capturing the rules and constraints on permitted annotations and controlled lists of values.

This is a proxy class for the formal properties. It is aimed at capturing the documental and partially the constraints imposed on this property.

Superclasses:

- *skos:Concept*
- *sh:PropertyShape*

[5]Class diagram, `https://en.wikipedia.org/wiki/Class_diagram`

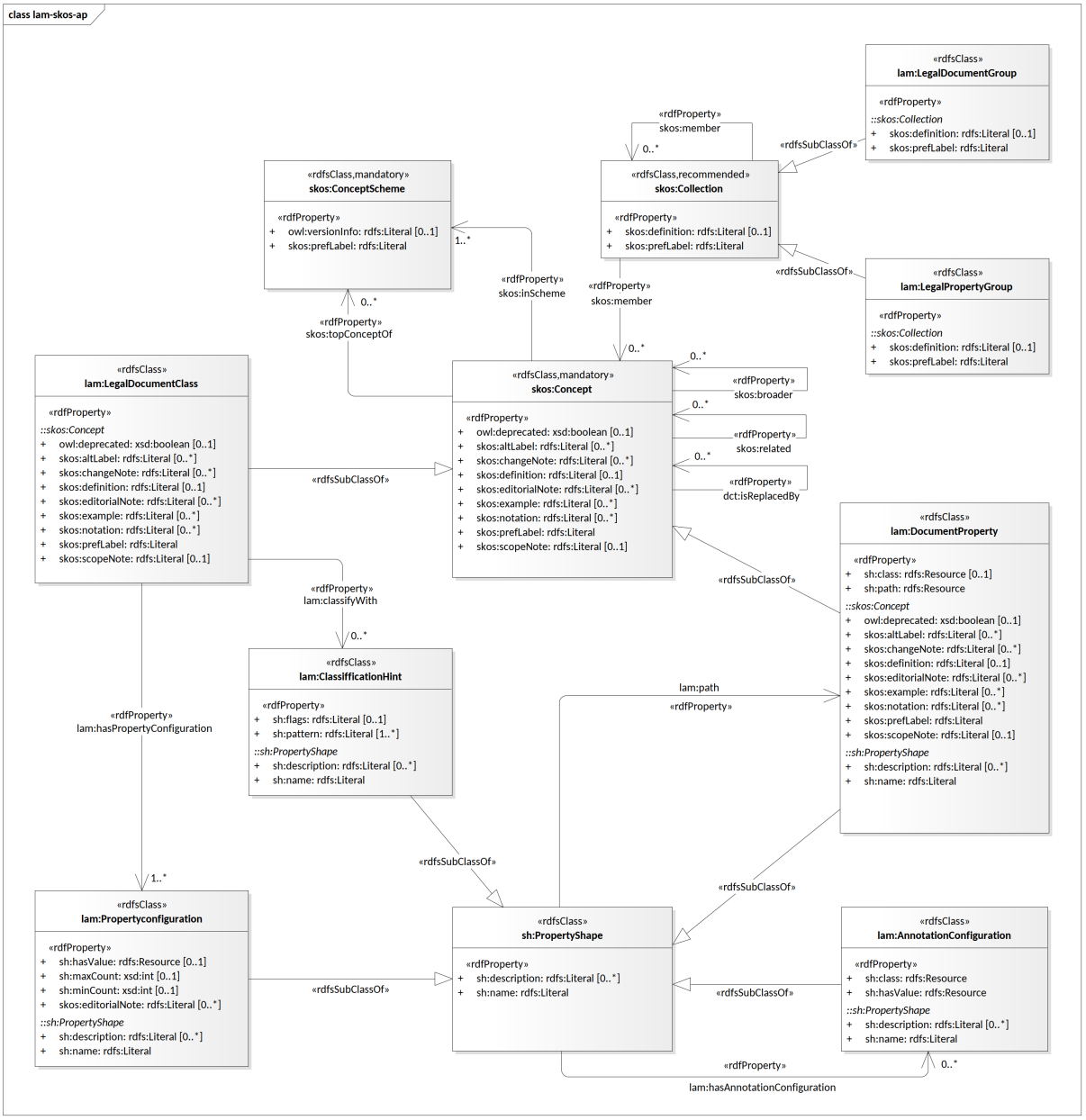

Figure 2: UML class diagram for the SRC-AP-VB3 application profile

Name	Type	Cardinality	Definition
sh:class	rdfs:Resource	0..1	The condition specified by sh:class is that each value node is a SHACL instance of a given type. This property constraints the range of the values that the property may take.
sh:path	rdfs:Resource	1..1	The property that is formally used in describing the legal documents. Usually a CDM property.

Name	Type	Cardinality	Definition

Table 2: Properties of the lam:DocumentProperty class

6.2 lam:LegalDocumentClass

This class aims at describing a set of legal documents that have in common a set of properties. It aims at capturing the rules and constraints on legal document properties, metadata elements, and cardinality.

Superclasses:

- *skos:Concept*

Name	Type	Cardinality	Definition
lam:hasProperty Configuration	lam:PropertyConfiguration	1..*	The constraint definition for a legal document property.
lam:classifyWith	lam:Classiffication Hint	0..*	

Table 3: Properties of the lam:LegalDocumentClass class

6.3 lam:LegalDocumentGroup

A way of grouping legal document classes.

Superclasses:

- *skos:Collection*

6.4 lam:LegalPropertyGroup

A way of grouping properties of the legal documents.

Superclasses:

- *skos:Collection*

6.5 lam:AnnotationConfiguration

Annotation configurations define the constraints on the annotations of the legal documents.

Superclasses:

- *sh:PropertyShape*

Name	Type	Cardinality	Definition
sh:class	rdfs:Resource	1..1	The condition specified by sh:class is that each value node is a SHACL instance of a given type. This property constraints the range of the values that the property may take.
sh:hasValue	rdfs:Resource	1..1	sh:hasValue specifies the condition that at least one value node is equal to the given RDF term.

Table 4: Properties of the lam:AnnotationConfiguration class

6.6 lam:ClassificationHint

This class defines how a document may be automatically assigned as a member of a given class of legal documents.

Superclasses:

- *sh:PropertyShape*

Name	Type	Cardinality	Definition
sh:flags	rdfs:Literal	0..1	An optional string of flags accepting SPARQL RegEx flags.
sh:pattern	rdfs:Literal	1..*	sh:pattern specifies a regular expression that each value node matches to satisfy the condition. It can be plain text or RegEx pattern matching the value of a given lam:property.

Table 5: Properties of the lam:ClassificationHint class

6.7 lam:Propertyconfiguration

The Property Configuration define constraints on a lam:DocumentProperty for a given lam:DocumentClass

Superclasses:

- *sh:PropertyShape*

Name	Type	Cardinality	Definition
sh:hasValue	rdfs:Resource	0..1	sh:hasValue specifies the condition that at least one value node is equal to the given RDF term.
sh:maxCount	xsd:int	0..1	Maximum cardinality constraint. Formally sh:maxCount specifies the maximum number of value nodes that satisfy the condition.

Name	Type	Cardinality	Definition
sh:minCount	xsd:int	0..1	Minimum cardinality constraint. Formally sh:minCount specifies the minimum number of value nodes that satisfy the condition. If the minimum cardinality value is 0 then this constraint is always satisfied and so may be omitted.
skos:editorialNote	rdfs:Literal	0..*	A note for an editor, translator or maintainer of the vocabulary.

Table 6: Properties of the lam:Propertyconfiguration class

6.8 skos:ConceptScheme

A SKOS concept scheme can be viewed as an aggregation of one or more SKOS concepts. Semantic relationships (links) between those concepts may also be viewed as part of a concept scheme. This definition is, however, meant to be suggestive rather than restrictive, and there is some flexibility in the formal data model stated below.

Thesauri, classification schemes, subject heading lists, taxonomies, 'folksonomies', and other types of controlled vocabulary are all examples of concept schemes. Concept schemes are also embedded in glossaries and terminologies.

Name	Type	Cardinality	Definition
owl:versionInfo	rdfs:Literal	0..1	An owl:versionInfo statement generally has as its object a string giving information about this version. This statement does not contribute to the logical meaning of the resource.
skos:prefLabel	rdfs:Literal	1..1	The preferred lexical label for a resource, in a given language. No two concepts in the same concept scheme may have the same preferred label in a given language.

Table 7: Properties of the skos:ConceptScheme class

6.9 skos:Concept

A SKOS concept can be viewed as an idea or notion; a unit of thought. However, what constitutes a unit of thought is subjective, and this definition is meant to be suggestive, rather than restrictive.

The notion of a SKOS concept is useful when describing the conceptual or intellectual structure of a knowledge organization system, and when referring to specific ideas or meanings established within a KOS.

Note that, because SKOS is designed to be a vehicle for representing semi-formal KOS, such as thesauri and classification schemes, a certain amount of flexibility has been built in to the formal definition of this class.

Name	Type	Cardinality	Definition
owl:deprecated	xsd:boolean	0..1	States whether the resource is current or deprecated. By deprecating a resource, it means that it should not be used in new documents. Deprecation is a feature commonly used in versioning software to indicate that a particular feature is preserved for backward-compatibility purposes, but may be phased out in the future.
skos:altLabel	rdfs:Literal	0..*	An alternative lexical label for a resource. Acronyms, abbreviations, spelling variants, and irregular plural/singular forms may be included among the alternative labels for a concept.
skos:changeNote	rdfs:Literal	0..*	A note about a modification to a concept.
skos:definition	rdfs:Literal	0..1	A statement or formal explanation of the meaning of a concept.
skos:editorialNote	rdfs:Literal	0..*	A note for an editor, translator or maintainer of the vocabulary.
skos:example	rdfs:Literal	0..*	An example of the use of a concept.
skos:notation	rdfs:Literal	0..*	A notation is a string of characters such as "T58.5" or "303.4833" used to uniquely identify a concept within the scope of a given concept scheme or within a specified context.
skos:prefLabel	rdfs:Literal	1..1	The preferred lexical label for a resource, in a given language. No two concepts in the same concept scheme may have the same preferred label in a given language.
skos:scopeNote	rdfs:Literal	0..1	A note that helps to clarify the meaning of a concept.
skos:broader	skos:Concept	0..*	A concept that is more general in meaning. Broader concepts are typically rendered as parents in a concept hierarchy (tree).
dct:isReplacedBy	skos:Concept	0..*	A related resource that supplants, displaces, or supersedes the described resource.
skos:related	skos:Concept	0..*	A concept with which there is an associative semantic relationship.

Table 8: Properties of the skos:Concept class

6.10 skos:Collection

A meaningful collection of concepts. Labeled collections can be used with collectible semantic relation properties i.e. skos:narrower, where you would like a set of concepts to be displayed under a "node label" in the hierarchy.

Name	Type	Cardinality	Definition
skos:definition	rdfs:Literal	0..1	A statement or formal explanation of the meaning of a collection.
skos:prefLabel	rdfs:Literal	1..1	The preferred lexical label for a resource, in a given language. No two concepts in the same concept scheme may have the same preferred label in a given language.
skos:member	skos:Collection	0..*	A member of a collection.

Table 9: Properties of the skos:Collection class

6.11 rdfs:Resource

All things described by RDF are called resources, and are instances of the class rdfs:Resource. This is the class of everything.

6.12 sh:PropertyShape

A property shape is a shape in the shapes graph that is the subject of a triple that has sh:path as its predicate. A shape has at most one value for sh:path. Each value of sh:path in a shape must be a well-formed SHACL property path.

In LAM-SKOS-AP the role of sh:path is taken by lam:path.

Name	Type	Cardinality	Definition
sh:description	rdfs:Literal	0..*	Property shape may have values for sh:description to provide descriptions of the property in the given context.
sh:name	rdfs:Literal	1..1	Property shapes may have one or more values for sh:name to provide human-readable labels for the property in the target where it appears.

Table 10: Properties of the sh:PropertyShape class

6.13 euvoc:XlNotation

A notation is a string of characters used to uniquely identify a concept within a specified context.

Like the skosxl:Label class reifies SKOS label statements, XlNotation reifies SKOS notation statements. This class permits, if needed, to maintain the historical view of the values and add additional provenance descriptions.

Name	Type	Cardinality	Definition
dct:created	xsd:date	0..1	Date of creation of the resource.
dct:modified	xsd:date	0..1	Date of modification of the resource.
euvoc:endDate	xsd:date	0..1	End of the validity period. If a resource has an end date then it must be marked as deprecated.
euvoc:startDate	xsd:date	0..1	Beginning of the validity period.
owl:deprecated	xsd:boolean	0..1	States whether the resource is current or deprecated. By deprecating a resource, it means that it should not be used in new documents. Deprecation is a feature commonly used in versioning software to indicate that a particular feature is preserved for backward-compatibility purposes, but may be phased out in the future.
rdf:value	rdfs:Literal	1..1	The literal form of the notation.
dct:type	skos:Concept	1..1	Specify the context where a specified notation is considered unique.

Table 11: Properties of the euvoc:XlNotation class

6.14 euvoc:XlNote

Like the skosxl:Label class reifies SKOS label statements, XlNote reifies SKOS note statements (i.e. skos:editorialNote, skos:example, skos:historyNote, skos:definition, skos:scopeNote and skos:changeNote). This class permits, if needed, to maintain the historical view of the values and add additional provenance descriptions.

Name	Type	Cardinality	Definition
dct:created	xsd:date	0..1	Date of creation of the resource.
dct:modified	xsd:date	0..1	Date of modification of the resource.
dct:source	rdfs:Resource	0..1	A related resource from which the described resource is derived. The described resource may be derived from the related resource in whole or in part. Recommended best practice is to identify the related resource by means of a string conforming to a formal identification system.

Name	Type	Cardinality	Definition
owl:deprecated	xsd:boolean	0..1	States whether the resource is current or deprecated. By deprecating a resource, it means that it should not be used in new documents. Deprecation is a feature commonly used in versioning software to indicate that a particular feature is preserved for backward-compatibility purposes, but may be phased out in the future.
rdf:value	rdfs:Literal	1..1	The literal form of the note.

Table 12: Properties of the euvoc:XlNote class

6.15 rdfs:Literal

The class rdfs:Literal is the class of literal values such as strings and integers. Property values such as textual strings are examples of RDF literals.

6.16 xsd:boolean

The boolean data type is used to specify a true or false value.

6.17 xsd:date

The date data type is used to specify a date. The date is specified in the following form "YYYY-MM-DD" where:

- YYYY indicates the year
- MM indicates the month
- DD indicates the day

Note: All components are required!

6.18 xsd:int

7 Conformance statement

7.1 Provider requirements

In order to conform to this application profile an application must be able to provide data assets that:

- comprise statements using at least the mandatory classes and properties.
- in addition, any of the recommended and optional properties and classes may be provided.

- employ values from the specified controlled vocabularies on properties with restricted value ranges. Additional controlled vocabularies may be used on properties with unrestricted value ranges.

7.2 Receiver requirements

In order to conform to this application profile an application that receives a compliant data asset must be able to:

- process information for all mandatory classes and optionally for recommended and optional classes.
- process information for all mandatory properties and optionally for recommended and optional properties.
- process information for all controlled vocabularies.

"processing" means that receivers must accept incoming data and transparently provide these data to applications and services. It does neither imply nor prescribe what applications and services finally do with the data (parse, convert, store, make searchable, display to users, etc.).

8 Multilingual aspects

Multilingual aspects related to this Application Profile concern all properties whose contents are expressed as strings with human-readable text. Wherever such properties are used, the string values are of one of two types:

- The string is free text. Examples are descriptions and labels. Such text may be translated into several languages.
- The string is an appellation of a "named entity". Examples are names of organisations or persons. These names may have parallel versions in other languages but those versions don't need to be literal translations.

Wherever values of properties are expressed with either type of string, the property can be repeated with translations in the case of free text and with parallel versions in case of named entities. For free text, the language tag is mandatory. For named entities, the language tag is optional and should only be provided if the parallel version of the name is strictly associated with a particular language. For example, the name "European Union" has parallel versions in all official languages of the union, while a name like "W3C" is not associated with a particular language and has no parallel versions.

How multilingual information is handled by the receiver applications, for example in indexing and user interfaces, is outside of the scope of this Application Profile.