SKOS-AP-EU: an application profile for publishing SKOS-based metadata assets

Eugeniu Costetchi

13/09/2019

Publications Office of the European Union

Table of Contents

1. Introduction

This document defines the application profile of SKOS 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].

The scope of the application profile is originally intended for, but not limited to, the reference metadata assets managed by the Publications Office of the European Union. *Reference metadata* assets refer to thesauri, taxonomies, authority tables, reference tables, controlled vocabularies, etc. Examples of such assets maintained by the Publications Office are EuroVoc thesaurus, Corporate Body, Language and Country authority lists. The complete list of assets can be found on EU Vocabularies Website.

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. 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.

[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.

- *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.).

3. Context of use

4. External vocabularies

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).

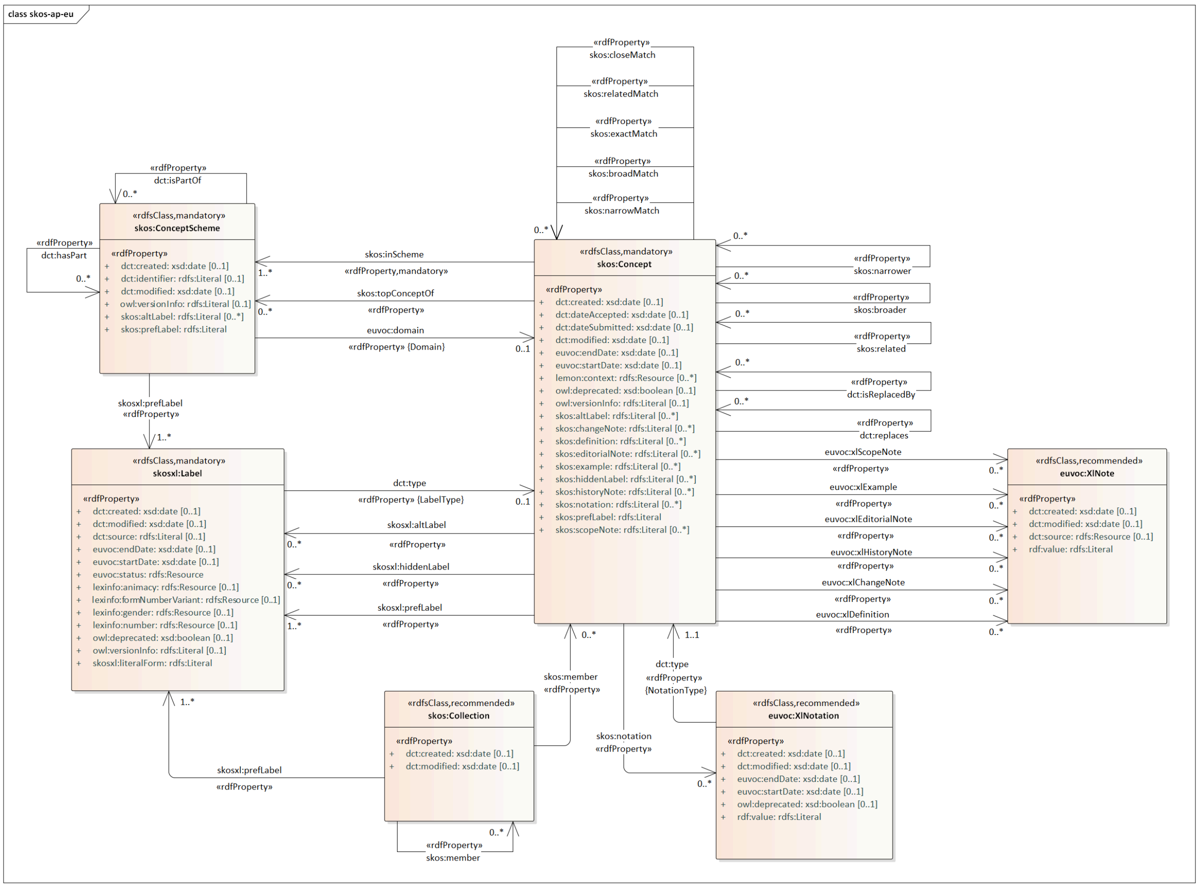
The stereotypes, marked in the double angle brackets (<< >>), used in the diagram are based on RDFS model. They have an indicative and not a semantic value. The "mandatory", "optional" and "recommended" stereotypes are normative and have the meaning defined in the terminology section .

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.

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

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

Figure 1. UML class diagram of SRC-AP-VB3 application profile


class skos-ap-eu
«rdfProperty»
skos:closeMatch
«rdfProperty»
skos:relatedMatch
«rdfProperty»
skos:exactMatch
«rdfProperty»
skos:broadMatch
«rdfProperty»
skos:narrowMatch
0..*
«rdfProperty»
dct:isPartOf
0..*
«rdfsClass,mandatory»
skos:ConceptScheme
«rdfProperty»
+ dct:created: xsd:date [0..1]
+ dct:identifier: rdfs:Literal [0..1]
+ dct:modified: xsd:date [0..1]
+ owl:versionInfo: rdfs:Literal [0..1]
+ skos:altLabel: rdfs:Literal [0..*]
+ skos:prefLabel: rdfs:Literal
«rdfProperty»
dct:hasPart
0..*
skos:inScheme
«rdfProperty,mandatory»
1..*
skos:topConceptOf
«rdfProperty»
0..*
euvoc:domain
«rdfProperty» {Domain}
0..1
«rdfsClass,mandatory»
skos:Concept
«rdfProperty»
+ dct:created: xsd:date [0..1]
+ dct:dateAccepted: xsd:date [0..1]
+ dct:dateSubmitted: xsd:date [0..1]
+ dct:modified: xsd:date [0..1]
+ euvoc:endDate: xsd:date [0..1]
+ euvoc:startDate: xsd:date [0..1]
+ lemon:context: rdfs:Resource [0..*]
+ owl:deprecated: xsd:boolean [0..1]
+ owl:versionInfo: rdfs:Literal [0..1]
+ skos:altLabel: rdfs:Literal [0..*]
+ skos:changeNote: rdfs:Literal [0..*]
+ skos:definition: rdfs:Literal [0..*]
+ skos:editorialNote: rdfs:Literal [0..*]
+ skos:example: rdfs:Literal [0..*]
+ skos:hiddenLabel: rdfs:Literal [0..*]
+ skos:historyNote: rdfs:Literal [0..*]
+ skos:notation: rdfs:Literal [0..*]
+ skos:prefLabel: rdfs:Literal
+ skos:scopeNote: rdfs:Literal [0..*]
0..*
«rdfProperty»
skos:narrower
0..*
«rdfProperty»
skos:broader
0..*
«rdfProperty»
skos:related
0..*
«rdfProperty»
dct:isReplacedBy
0..*
«rdfProperty»
dct:replaces
euvoc:xlScopeNote
«rdfProperty»
0..*
euvoc:xlExample
«rdfProperty»
0..*
euvoc:xlEditorialNote
«rdfProperty»
0..*
euvoc:xlHistoryNote
«rdfProperty»
0..*
euvoc:xlChangeNote
«rdfProperty»
0..*
euvoc:xlDefinition
«rdfProperty»
0..*
«rdfsClass,recommended»
euvoc:XlNote
«rdfProperty»
+ dct:created: xsd:date [0..1]
+ dct:modified: xsd:date [0..1]
+ dct:source: rdfs:Resource [0..1]
+ rdf:value: rdfs:Literal
skosxl:prefLabel
«rdfProperty»
1..*
«rdfsClass,mandatory»
skosxl:Label
«rdfProperty»
+ dct:created: xsd:date [0..1]
+ dct:modified: xsd:date [0..1]
+ dct:source: rdfs:Literal [0..1]
+ euvoc:endDate: xsd:date [0..1]
+ euvoc:startDate: xsd:date [0..1]
+ euvoc:status: rdfs:Resource
+ lexinfo:animacy: rdfs:Resource [0..1]
+ lexinfo:formNumberVariant: rdfs:Resource [0..1]
+ lexinfo:gender: rdfs:Resource [0..1]
+ lexinfo:number: rdfs:Resource [0..1]
+ owl:deprecated: xsd:boolean [0..1]
+ owl:versionInfo: rdfs:Literal [0..1]
+ skosxl:literalForm: rdfs:Literal
dct:type
«rdfProperty» {LabelType}
0..1
skosxl:altLabel
«rdfProperty»
0..*
skosxl:hiddenLabel
«rdfProperty»
0..*
skosxl:prefLabel
«rdfProperty»
1..*
0..*
skos:member
«rdfProperty»
1..1
dct:type
«rdfProperty»
{NotationType}
skos:notation
«rdfProperty»
0..*
1..*
skosxl:prefLabel
«rdfProperty»
«rdfsClass,recommended»
skos:Collection
«rdfProperty»
+ dct:created: xsd:date [0..1]
+ dct:modified: xsd:date [0..1]
0..*
«rdfProperty»
skos:member
«rdfsClass,recommended»
euvoc:XlNotation
«rdfProperty»
+ dct:created: xsd:date [0..1]
+ dct:modified: xsd:date [0..1]
+ euvoc:endDate: xsd:date [0..1]
+ euvoc:startDate: xsd:date [0..1]
+ owl:deprecated: xsd:boolean [0..1]
+ rdf:value: rdfs:Literal

6. skos-ap-eu.xmi

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.

Table 1. Properties

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.

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.

Table 2. Properties

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.
rdf:value	rdfs:Literal	1..1	The literal form of the note.

rdfs:Resource

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

skos:Collection

Table 3. Properties

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.
skos:member	skos:Collection	0..*	Collection has member.
skos:member	skos:Concept	0..*	Collection has member.
skosxl:prefLabel	skosxl:Label	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..*	Collection has member.

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.

Table 4. Properties

Name	Type	Cardinality	Definition
dct:created	xsd:date	0..1	Date of creation of the resource.
dct:dateAccepted	xsd:date	0..1	Date when the resource creation or modification is accepted by the governance committee.
dct:dateSubmitted	xsd:date	0..1	Date when the resource creation or modification is sent for review to the governance committee.
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.

Name	Type	Cardinality	Definition
lemon:context	rdfs:Resource	0..*	Denotes the pragmatic, discursive or technical context of a concept or a constraint on the concept properties.
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.
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: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..*	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:hiddenLabel	rdfs:Literal	0..*	A lexical label for a resource that should be hidden when generating visual displays of the resource, but should still be accessible to free text search operations. Mis-spelled terms are normally included as hidden labels.
skos:historyNote	rdfs:Literal	0..*	A note about the past state/use/meaning 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..*	A note that helps to clarify the meaning of a concept.
skos:inScheme	skos:ConceptScheme	1..*	A concept scheme in which the concept is included. A concept may be a member of more than one concept scheme.
euvoc:xlDefinition	euvoc:XlNote	0..*	A statement or formal explanation of the meaning of a concept.
skos:broadMatch	skos:Concept	0..*	The properties skos:broadMatch and skos:narrowMatch are used to state a hierarchical mapping link between two concepts.
euvoc:xlHistoryNote	euvoc:XlNote	0..*	A note about the past state/use/meaning of a concept.
skos:narrowMatch	skos:Concept	0..*	The properties skos:broadMatch and skos:narrowMatch are used to state a hierarchical mapping link between two concepts.
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).
euvoc:xlExample	euvoc:XlNote	0..*	An example of the use of a concept.
dct:isReplacedBy	skos:Concept	0..*	A related resource that supplants, displaces, or supersedes the described resource.
skos:exactMatch	skos:Concept	0..*	The property skos:exactMatch is used to link two concepts, indicating a high degree of confidence that the concepts can be used interchangeably across a wide range of information retrieval applications.
skos:closeMatch	skos:Concept	0..*	The property skos:closeMatch is used to link two concepts that are sufficiently similar that they can be used interchangeably in some information retrieval applications.
skos:relatedMatch	skos:Concept	0..*	The property skos:relatedMatch is used to state an associative mapping link between two concepts.
skos:narrower	skos:Concept	0..*	A concept that is less general in meaning. Narrower concepts are typically rendered as children in a concept hierarchy (tree).
skosxl:altLabel	skosxl:Label	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.
skosxl:hiddenLabel	skosxl:Label	0..*	A lexical label for a resource that should be hidden when generating visual displays of the resource, but should still be accessible to free text search operations. Mis-spelled terms are normally included as hidden labels.
skos:topConceptOf	skos:ConceptScheme	0..*	The property skos:hasTopConcept is, by convention, used to link a concept scheme to the SKOS concept(s) which are topmost in the hierarchical relations for that scheme.

Name	Type	Cardinality	Definition
skosxl:prefLabel	skosxl:Label	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:notation	euvoc:XlNotation	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.
dct:replaces	skos:Concept	0..*	A related resource that is supplanted, displaced, or superseded by the described resource.
skos:related	skos:Concept	0..*	A concept with which there is an associative semantic relationship.
euvoc:xlChangeNote	euvoc:XlNote	0..*	A note about a modification to a concept.
euvoc:xlEditorialNote	euvoc:XlNote	0..*	A note for an editor, translator or maintainer of the vocabulary.
euvoc:xlScopeNote	euvoc:XlNote	0..*	A note that helps to clarify the meaning of a concept.
skos:broadMatch	skos:Concept	0..*	The properties skos:broadMatch and skos:narrowMatch are used to state a hierarchical mapping link between two concepts.
skos:narrowMatch	skos:Concept	0..*	The properties skos:broadMatch and skos:narrowMatch are used to state a hierarchical mapping link between two concepts.
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:exactMatch	skos:Concept	0..*	The property skos:exactMatch is used to link two concepts, indicating a high degree of confidence that the concepts can be used interchangeably across a wide range of information retrieval applications.
skos:closeMatch	skos:Concept	0..*	The property skos:closeMatch is used to link two concepts that are sufficiently similar that they can be used interchangeably in some information retrieval applications.
skos:relatedMatch	skos:Concept	0..*	The property skos:relatedMatch is used to state an associative mapping link between two concepts.
skos:narrower	skos:Concept	0..*	A concept that is less general in meaning. Narrower concepts are typically rendered as children in a concept hierarchy (tree).
dct:replaces	skos:Concept	0..*	A related resource that is supplanted, displaced, or superseded by the described resource.
skos:related	skos:Concept	0..*	A concept with which there is an associative semantic relationship.

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.

Table 5. Properties

Name	Type	Cardinality	Definition
dct:created	xsd:date	0..1	Date of creation of the resource.
dct:identifier	rdfs:Literal	0..1	An unambiguous reference to the resource within a given context. Recommended best practice is to identify the resource by means of a string conforming to a formal identification system.
dct:modified	xsd:date	0..1	Date of modification of the resource.
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: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.

Name	Type	Cardinality	Definition
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.
dct:hasPart	skos:ConceptScheme	0..*	A related resource that is included either physically or logically in the described resource.
euvoc:domain	skos:Concept	0..1	Indicates the subject of the controlled vocabulary. This property has a similar function as the dct:subject and dcat:theme.
dct:isPartOf	skos:ConceptScheme	0..*	A related resource in which the described resource is physically or logically included.
skosxl:prefLabel	skosxl:Label	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.
dct:hasPart	skos:ConceptScheme	0..*	A related resource that is included either physically or logically in the described resource.
dct:isPartOf	skos:ConceptScheme	0..*	A related resource in which the described resource is physically or logically included.

skosxl:Label

The class skosxl:Label is a special class of lexical entities. An instance of the class skosxl:Label is a resource and may be named with a URI.

An instance of the class skosxl:Label has a single literal form. This literal form is an RDF plain literal (which is a string of UNICODE characters and an optional language tag [https://www.w3.org/TR/rdf-concepts/]). The property skosxl:literalForm is used to give the literal form of an skosxl:Label. If two instances of the class skosxl:Label have the same literal form, they are not necessarily the same resource.

Table 6. Properties

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:Literal	0..1	A related resource from which the described resource is derived 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.
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.
euvoc:status	rdfs:Resource	1..1	Indicates the status of a resource in the process of asset management.
lexinfo:animacy	rdfs:Resource	0..1	The characteristic of a word indicating that in a given discourse community, its referent is considered to be alive or to possess a quality of volition or consciousness.
lexinfo:formNumberVariant	rdfs:Resource	0..1	A grammatical category that indicates grammatical variation in numerals. This feature is used in Maltese language.
lexinfo:gender	rdfs:Resource	0..1	A grammatical category that indicates grammatical relationships between words in sentences.
lexinfo:number	rdfs:Resource	0..1	Grammatical category for the variation in form of nouns, pronouns, and any words agreeing with them, depending on how many persons or things are referred to. In many languages, the grammatical distinction that indicates the number of objects referred to by the term or word.
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.
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.
skosxl:literalForm	rdfs:Literal	1..1	The literal form of an skosxl:Label. An instance of the class skosxl:Label has one and only one literal form.
dct:type	skos:Concept	0..1	Specify the context where a specified notation is considered unique.

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.

xsd:boolean

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

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!

7. On concept evolution and versioning

SKOS is a common data model for sharing and linking knowledge organization systems via the Web. It describes means to define concepts and organise them in concept schemes but omits to specify anything about how such organisation should evolve over time.

There has been discussions and work done to address this issue. A list of references on this topic is available on W3C wiki [(6)]

. This section reflects the result of discussions in the MDR team of the Publications Office and addresses a few issues related to evolution of digital assets based on SKOS model. The solutions described below have informative purpose and should in no way be considered normative.

Preliminary questions

It is difficult to talk about temporal evolution of SKOS assets as a whole and therefore of concepts and concept schemes in particular without quickly sliding into the classic problems in metaphysics[(7)] related to time[(8)], change[(9)] and identity[(10)]. An alternative and perhaps a useful way of escaping the metaphysical distinctions is taking a semiotic[(11)] standpoint and think about the evolution of signs, "external reality" things and "mental" concepts. Hence it is useful to make a few distinctions between kinds of described things/entities as the outcome of such clarification will naturally lead to a versioning and evolution strategy for the asset at hand.

[(6)] SKOS/Issues/ConceptEvolution. https://www.w3.org/2001/sw/wiki/SKOS/Issues/ConceptEvolution

[(7)] van Inwagen, Peter and Sullivan, Meghan, "Metaphysics", The Stanford Encyclopedia of Philosophy (Spring 2018 Edition), Edward N. Zalta (ed.), https://plato.stanford.edu/archives/spr2018/entries/metaphysics/

[(8)] Markosian, Ned, "Time", The Stanford Encyclopedia of Philosophy (Fall 2016 Edition), Edward N. Zalta (ed.), https://plato.stanford.edu/archives/fall2016/entries/time/.

[(9)] Gallois, Andre, "Identity Over Time", The Stanford Encyclopedia of Philosophy (Winter 2016 Edition), Edward N. Zalta (ed.), https://plato.stanford.edu/archives/win2016/entries/identity-time/.

[(10)] Deutsch, Harry and Garbacz, Pawel, "Relative Identity", The Stanford Encyclopedia of Philosophy (Fall 2018 Edition), Edward N. Zalta (ed.), https://plato.stanford.edu/archives/fall2018/entries/identity-relative/

[(11)] Atkin, Albert, "Peirce's Theory of Signs", The Stanford Encyclopedia of Philosophy (Summer 2013 Edition), Edward N. Zalta (ed.), https://plato.stanford.edu/archives/sum2013/entries/peirce-semiotics/.

Next are provided a few competency questions aimed at clarifying the nature of the asset at hand and thus getting one step closer in making a choice in the versioning strategy that will be outlined later.

- What is the level of abstraction of managed concepts? Do the concepts represent entities/individuals or more abstract classes of things?
- What is the nature of the described things and what keeps them together in one asset?
- How to distinguish one concept from another one? What are the criteria of judgement?
- For properties beyond SKOS model (employed in the Application Profile) decide which of them are *transient* (time dependent) and *intransigent* (time invariant).

Versioning strategies

The problem of versioning can be kept outside the SKOS content by simply publishing successive evolved versions of a plain asset marking accordingly each dataset with a new version. Otherwise, if the historicity needs to permeate the SKOS content then there are few approaches to do so.

There is an approach which proposes altering the core SKOS model described by Tennis and Sutton in their paper "Extending the Simple Knowledge Organization System for Concept Management in Vocabulary Development Applications"[(12)]. This approach, which suggests describing concepts at two levels (abstract and concrete) is clean and straight forward way of distinguishing transient from intransigent things. However, an additional level of complexity is added, making it difficult to use for lay users. Also the extension is hardly backwards compatible with the original SKOS model. This means that a wider adoption is needed for this new model to be deemed practical. Finally there is not software support for maintaining such descriptions.

Without altering the conceptual structure of the SKOS model, there are three versioning strategies that are considered practical:

- *Concept evolution* - the concept description comprises the entire or partial historical account. This means that the concept URI is maintained stable over time and the changes are incrementally recorded on the concept distinguishing between the latest valid value and the historical trace of values for each property. The strength of this approach is that the URI remains persistent over a long time, but the weakness is that it is more complex to derive the latest correct and valid description of the concept.
- *Concept succession* - the concept description has no historical account but the concept has a validity interval. When the concept description is no longer valid a new successor concept replaces the current one, the latter becoming deprecated. This means that the concept URI is maintained stable over time, but when a new concept is created to replace the current one, then it is marked deprecated thus encouraging the users to switch and start using the new URI.
- *Versioning by concept scheme* - the concept schemes function as version containers. This approach resembles slightly the "No versioning" approach, only that each dataset is conceptualised as a concept scheme. A concept belongs to as many concept schemes for as long it is current. The concept scheme provides validity interval or publication date.
- *No versioning* - keep the versioning issue outside the content. Versions are maintained as separate editions of the evolving dataset.

Necessary extensions

In order to deal with versioning, regardless of the approach, a few properties are deemed useful in order to mark the concept status, mark validity interval of a concept and eventually specify the successor/predecesor of a concept.

[(12)] Tennis, J. T. and Sutton, S. A. (2008), Extending the simple knowledge organization system for concept management in vocabulary development applications. J. Am. Soc. Inf. Sci., 59: 25-37. doi:10.1002/asi.20702

In order to accommodate the **concept versioning**, the properties for which the historicity is important or necessary, have to be reified is some way. The recommendations are provided in the "RDF Primer"[13] and in the "Defining N-ary Relations on the Semantic Web"[14]. A popular reification example for SKOS properties allowing the qualification of concept labels is provided in the SKOS-XL extension[15]. This extension alone, however is not sufficient to describe the label temporal evolution, and two more properties are necessary: label status and/or label validity interval (start and end date). This approach allows instatiations like the one in the listing below where the concept MARE from CorporateBody authority table has several pref labels out of which only the last one is valid and the first two are deprecated each having a validity period.

```
corporate-body:MARE a skos:Concept ;
  skosxl:prefLabel [
    euvoc:startDate "1977-01-01";
    euvoc:endDate "1999-07-30";
    owl:deprecated "true";
    skoxl:literalValue "DG XIV Fisheries".
  ];
  skoxl:prefLabel [
    euvoc:startDate "1999-10-01";
    euvoc:endDate "2005-01-12";
    owl:deprecated "true";
    skosxl:literalValue "Directorate General for Fisheries".
  ];
  skoxl:prefLabel [
    euvoc:startDate "2005-01-13";
    skosxl:literalValue "Directorate-General for Maritime Affairs and Fisheries".
  ].
```

Concept succession requires a slightly different extension. The need for specifying status and validity interval still remains and in addition there needs to be a way to also specify connection between the old and the new superseding concept. In the listing below is provided an example of how concept succession may be instantiated.

```
corporate-body:EEC a skos:Concept ;
  skos:prefLabel "European Economic Community".
  euvoc:startDate "1958-01-01";
  euvoc:endDate "1993-10-31";
  owl:deprecated "true";
  dct:isReplacedBy corporate-body:EURCOM.

corporate-body:EURCOM a skos:Concept ;
  skos:prefLabel "European Economic Community".
  euvoc:startDate "1993-11-01";
  euvoc:endDate "2009-11-30";
  owl:deprecated "true";
  dct:isReplacedBy corporate-body:EURUN.

corporate-body:EURUN a skos:Concept ;
  skos:prefLabel "European Union".
  euvoc:startDate "2009-12-01".
```

Versioning by concept scheme does not require any SKOS extensions, at least not for concept descriptions. It is recommended however to describe evolution of concept schemes using similar means as above. In the listing below is provided an example how versioning by concept scheme may be instantiated.

```
nuts-scheme:2010 a skos:ConceptScheme;
  skos:prefLabel "Nomenclature of territorial units for statistics";
  owl:versionInfo "version 2010";
  dct:isReplacedBy nuts-scheme:2013.
```

[13] Miller, E., & Manola, F. (2004). RDF Primer. https://www.w3.org/TR/rdf-primer/#deification

[14] Rector, A., & Noy, N. (2006). Defining N-ary Relations on the Semantic Web. https://www.w3.org/TR/swbp-n-aryRelations/

[15] Miles, A. & Isaac A. (2008). SKOS Simple Knowledge Organization System eXtension for Labels (XL). http://www.w3.org/2006/07/SWD/SKOS/xl/

```
nuts-scheme:2013 a skos:ConceptScheme;
  skos:prefLabel "Nomenclature of territorial units for statistics";
  owl:versionInfo "version 2013".

nuts-code:UKK42 a skosConcept;
  skos:prefLabel "Torbay";
  skos:inScheme nuts-scheme2010;
  skos:inScheme nuts-scheme2013.

nuts-code:UKD37 a skosConcept;
  skos:prefLabel "Greater Manchester North East";
  skos:inScheme nuts-scheme2013.
```

8. Requirements for controlled vocabularies

The following is a list of requirements for the controlled vocabularies that are used in this application profile as controlled list of values.

Controlled vocabularies should:

- Be published under an open licence.
- Be operated and/or maintained by an institution of the European Union, by a recognised standards organisation or another trusted organisation.
- Be properly documented.
- Have labels in multiple languages, ideally in all official languages of the European Union.
- Have terms that are identified by dereferenceable[16] URIs with each URI resolving to descriptions about the term.
- Have associated persistence and versioning policies.

These criteria have been applied to select controlled vocabularies used in this application profile.

9. Conformance statement

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.

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.).

[16] Dereferencing HTTP URIs. https://www.w3.org/2001/tag/doc/httpRange-14/2007-05-31/HttpRange-14

10. 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.

11. Publishing Linked Data

This Application Profile is intended for use in a *Linked Data*(17) environment. Therefore data providers should consider publishing the data assets following the recommendations in the W3C Notes "Best Practice Recipes for Publishing RDF Vocabularies"(18) , "Best Practices for Publishing Linked Data"(19) and the ISA report "10 Rules for Persistent URIs"(20) .

(17) W3C. Linked Data. http://www.w3.org/standards/semanticweb/data

(18) W3C. Best Practice Recipes for Publishing RDF Vocabularies. http://www.w3.org/TR/swbp-vocab-pub/

(19) W3C. Best Practices for Publishing Linked Data. https://dvcs.w3.org/hg/gld/raw-file/default/bp/index.html

(20) European Commission. Joinup. 10 Rules for Persistent URIs. https://joinup.ec.europa.eu/community/semic/document/10-rules-persistent-uris