Data Model for Lexicography (DMLex), Version 1.0

Working Draft 01

21 December 2020

Specification URIs

This version:
http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/dmlex-v1.0-wd01.html (Authoritative)
http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/dmlex-v1.0-wd01.pdf
http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/dmlex-v1.0-wd01.xml

Previous version:
http://docs.oasis-open.org/lexidma/dmlex/v1.0/N/A/dmlex-v1.0-N/A.html (Authoritative)
http://docs.oasis-open.org/lexidma/dmlex/v1.0/N/A/dmlex-v1.0-N/A.pdf
http://docs.oasis-open.org/lexidma/dmlex/v1.0/N/A/dmlex-v1.0-N/A.xml

Latest version:
http://docs.oasis-open.org/lexidma/dmlex/v1.0/dmlex-v1.0.html (Authoritative)
http://docs.oasis-open.org/lexidma/dmlex/v1.0/dmlex-v1.0.pdf
http://docs.oasis-open.org/lexidma/dmlex/v1.0/dmlex-v1.0.xml

Technical Committee:
OASIS Lexicographic Infrastructure Data Model and API (LEXIDMA) TC

Chair:
Tomaž Erjavec (tomaz.erjavec@ijs.si), Jozef Stefan Institute

Editors:
David Filip (david.filip@adaptcentre.ie), Trinity College Dublin (ADAPT)
Simon Krek (simon.krek@ijs.si), Jozef Stefan Institute

Additional artifacts:

NONE AT THE MOMENT

Related Work:

This specification is related to:

- *No related specifications.*

Declared namespaces:

This specification declares one or more namespaces. Namespace isn't considered an XML specific feature in this serialization independent specification.

The core namespace

- http://docs.oasis-open.org/lexidma/ns/dmlex-1.0

Module namespaces:

- TBD
- TBD

Key words:

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in BCP 14 [RFC2119] and [RFC8174] if, and only if, they appear in all capitals, as shown here.

Abstract:

This document defines the 1st version of a data model in support of the high priority technical goals described in the LEXIDMA TC's charter, including:

- Serialization independent Data Model for Lexicography (DMLex)
- XML serialization of DMLex
- JSON serialization of DMLex
- RDF serialization of DMLex
- Informative Ontolex-Lemon mapping
- Informative TEI Lex-0 mapping

Status:

This document was last revised or approved by the LEXIDMA TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=lexidma#technical.

TC members should send comments on this document to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/lexidma/.

This specification is provided under the Non-Assertion Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/lexidma/ipr.php).

Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification the following citation format should be used:

[DMLex-1.0]

Data Model for Lexicography Version 1.0. Edited by David Filip and Simon Krek. 21 December 2020. OASIS Working Draft 01. http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/dmlex-v1.0-wd01.html. Latest version: http://docs.oasis-open.org/lexidma/dmlex/v1.0/dmlex-v1.0.html.

Notices

Table of Contents

1 Introduction

Data Model for Lexicography (DMLex) is a *serialization independent Object Model* designed by a group of lexicographers, lexicographic software providers, and researchers. It is intended to provide a simple, modular, and easy to adopt data model for use by all lexicographic industry actors across companies and academia as well as geographic locations. Adoption of this data model will facilitate exchange of lexicographic and linguistic corpus data globally and enable effective interchange with adjacent industries such as language services, terminology management, and technical writing. Semantic interoperability of lexicographic data will help lexicographers to surpass linguistically and geographically demarcated approaches and create a truly global market for lexicographic data across and among languages and locales.

All text is normative unless otherwise labeled. The following common methods are used for labeling portions of this specification as informative and hence non-normative:

Appendices and sections marked as "(Informative)" or "Non-Normative" in title,
Notes (sections with the "Note" title),
Warnings (sections with the "Warning" title),
Examples (mainly example code listings, tree diagrams, but also any inline examples or illustrative exemplary lists in otherwise normative text),
Schema and other validation artifacts listings (the corresponding artifacts are normative, not their listings).

1.1 Glossary

1.1.1 Definitions

Agent

any application or tool that generates (creates), reads, edits, writes, processes, stores, renders or otherwise handles documents, messages, or any other instantiations of the *DMLex*.

Agent is the most general application conformance target that subsumes all other specialized user agents disregarding whether they are defined in this specification or not.

Enrich, Enriching

the process of associating metadata and resources with *DMLex* based lexicographic data.

Enricher, Enricher Agent

any *Agent* that performs the *Enriching* process.

Modify, Modification

the process of changing core and module *DMLex* based structural and inline objects that were previously created by other *Writers*

Processing Requirements

- LIOM objects MAY be *Modified* and *Enriched* at the same time.

Modifier, Modifier Agent

an *Agent* that performs the *Modification* process.

Writer, Writer Agent

an *Agent* that creates, generates, or otherwise writes a *DMLex* based document for whatever purpose, including but not limited to *Creator*, *Modifier*, and *Enricher Agents*.

> **Note**
>
> *DMLex* intends to define lossless interchange between and among different pipelines that can be based on arbitrary *DMLex* conformant specifications (public) or even private informally *DMLex* based formats (that could evolve into *DMLex* conformant public specifications over time). Details of interchange using a given serialization are to be found in this specifications normative and informative appendices on various serializations and mappings.

1.1.2 Key concepts

DMLex Core

The core of *DMLex* 1.0 consists of the minimum set of data objects required to (a) create a *DMLex* Instance that contains interoperable lexicographic data ...[ELABORATE]

The namespace that corresponds to the core subset of *DMLex* is `http://docs.oasis-open.org/lexidma/ns/dmlex-1.0`.

DMLex-defined (elements and attributes)

The following is the list of allowed schema URI prefixes for *DMLex-defined* elements and attributes:

http://docs.oasis-open.org/lexidma/ns/

Elements and attributes from other namespaces are not *DMLex-defined*.

DMLex Conformant Specification

A publicly available specification that normatively defines a serialization (document, fragment or message payload format) that conforms to the DMLex defined in this specification.

Currently known conformant serializations and mappings are defined in appendices to this standard: [olinks to appendices, including their normative status]

DMLex Instance

A document, fragment or message payload that is a valid instance of any *DMLex Conformant Specification*.

> **Note**
>
> Documents, fragments or message payloads cannot conform directly to DMLex, as DMLex itself doesn't define serializations and an *ad hoc* format cannot be reliably checked against the complex requirements of the DMLex. Currently known LIOM instances include: [Could define and register media tyoes based on some of the appendices].

DMLex Module

A module is an OPTIONAL set of data and metadata categories that were designed to store information about a process applied to a *DMLex Instance* and the data incorporated into a *DMLex Instance* as result of that process.

Each official module defined for DMLex Version 1.0 has its data categories and structure defined within this specification and corresponds to at least one dedicated namespace, see Normatively Used Namespaces.

2 Conformance

1. *DMLex Instances Conformance*
    a. Conformant *DMLex Instances* MUST be well formed and valid instances according to one of DMLex Serialization Specifications.
    b. Another Instance conformance clause.
    c. ...
    d. *DMLex Instances* MAY contain custom extensions, as defined in the Extension Mechanisms section. Extensions MUST be serialized in a way conformant with the pertaining DMLex Serialization Specifications.
2. *Application Conformance*

   a. DMLex *Writers* MUST create conformant *DMLex Instances* to be considered DMLex compliant.

   b. *Agents* processing conformant *DMLex Instances* that contain custom extensions are not REQUIRED to understand and process non-DMLex objects or attributes. However, conformant applications SHOULD preserve existing custom extensions when processing conformant *DMLex Instances*, provided that the objects that contain custom extensions are not removed according to DMLex Processing Requirements or the extension's own processing requirements.

   c. All *Agents* MUST comply with Processing Requirements for otherwise unspecified *Agents* or without a specifically set target *Agent*.

   d. Specialized *Agents* defined in this specification - this is *Writer*, *Modifier*, and *Enricher Agents* - MUST comply with the Processing Requirements targeting their specifically defined type of *Agent* on top of Processing Requirements targeting all *Agents* as per point c. above.

   e. DMLex is an object model explicitly designed for exchanging data among various *Agents*. Thus, a conformant DMLex application MUST be able to accept *DMLex Instances Created*, *Modified*, or *Enriched* by a different application, provided that:

      i. The processed files are conformant *DMLex Instances* according to the same DMLex Serialization Specification,

      ii. in a state compliant with all relevant Processing Requirements.

3. *Backwards Compatibility*

   a. N/A.

Note

DMLex Instances cannot be conformant to this specification w/o being conformant to a specific serialization.

3 The Core Specification

The DMLex Core is for monolingual lexical resources, where headwords, definitions, examples etc. are all in one and the same language.

3.1 `LexicographicResource` object type

A data set which can be viewed and used by humans as a dictionary and - simultaneously - ingested, processed and understood by software agents as a machine-readable database. Terminological note: *lexicographic* resource, not *lexical*.

Properties:

- `language` (optional, IETF language code)
  - The language of headwords, definitions, examples.
- `transcriptionScheme` (optional, reference to some external authority - which?)
  - The scheme (e.g. IPA) in which the `transcription` property of `Pronunciation` objects is given.

Children:

- `Entry` (one or more)

3.2 `Entry` object type

A part of a lexicographic resource which contains information related to exactly one headword.

Child of:

- `LexicographicResource`

Properties:

- `headword` (non-empty string)
  - The headword can be a single word, a multi-word expression, or any expression in the source language which is being described by the entry in the lexicographic resource.
- `homographNumber` (number, optional)

Note: entries do not have any explicit listing order. An application can imply a listing order from a combination of the headword and the homograph number.

Children:

- `PartOfSpeech` (zero or more)
- `Pronunciation` (zero or more)
- `InflectedForm` (zero or more)
- `Sense` (zero or more)

3.3 `Sense` object type

A part of an entry which groups together information relating to one of the (possibly multiple) meanings (or meaning potentials) of the entry's headword.

Child of:

- `Entry`

Properties:

- `listingOrder`
  - Can be implicit from the serialization.
- `indicator` (optional, non-empty string)
  - A short statement that indicates the meaning of a sense and permits its differentiation from other senses in the entry.
- `definition` (optional, non-empty string)
  - A long statement that describes and or explains the meaning of a sense.

Children:

- `Usage` (zero or more)
- `Example` (zero or more)

3.4 The difference between entries and senses

An **entry** is a container for "formal" properties of the headword such as orthography, morphology, syntax and pronunciation. A **sense** is a container for those of the headword's properties which are statements about semantics and pragmatics.

3.5 `PartOfSpeech` object type

Any of the word classes to which a lexical item may be assigned, e.g. noun, verb, adjective, etc.

Child of:

- `Entry`

Properties:

- `value` (non-empty string)
  - Can be constrained by the DMLex Controlled Vocabularies Module.
- `listingOrder`
  - Can be implicit from the serialization.

3.6 `Usage` object type

An indication of some restriction on the use of the lexical item. The restriction can be pragmatic (time, region, register), semantic (domain, semantic type) or formal ('no plural').

Child of:

- `Sense`
- `Pronunciation`
- `InflectedForm`

Properties:

- `value` (non-empty string)
  - Can be constrained by the DMLex Controlled Vocabularies Module.
  - Its type (eg. whether register, temporal, geographic etc) can be specified by the by the DMLex Controlled

    - Vocabularies Module.
- `listingOrder`
    - Can be implicit from the serialization.

3.7 `Pronunciation` object type

Information about the pronunciation of its parent.

Child of:

- `Entry`
- `InflectedForm`

Properties (at least one):

- `transcription` (non-empty string)
- `recording` (string: name or URL of a sound file)
- `listingOrder`
    - Can be implicit from the serialization.

Children:

- `Usage` (zero or more)

3.8 `InflectedForm` object type

An inflected headword is a form of the inflectional paradigm of its parent.

Child of:

- `Entry`

Properties:

- `label` (non-empty string) e.g. 'plural'
    - Can be constrained by the DMLex Controlled Vocabularies Module.
- `value` (non-empty string)
- `listingOrder`
    - Can be implicit from the serialization.

Children:

- `Usage` (zero or more)
- `Pronunciation` (zero or more)

3.9 `Example` object type

An instance of a lexical item's usage in a specific sense.

Child of:

- `Sense`

Properties:

- `text` (non-empty string)
- `listingOrder`
    - Can be implicit from the serialization.

4 Bilingual Module

Extends DMLex Core to support the encoding of bilingual lexicographic resources.

4.1 Extensions to `LexicographicResource` object type

Additional properties:

- `translationLanguage` (optional, IETF language code)
- `translationTranscriptionScheme` (optional, reference to some external authority - which?)
  - The scheme (e.g. IPA) in which the `transcription` property of `TranslationPronunciation` objects is given.

4.2 Extensions to `Sense` object type

Additional children:

- `HeadwordTranslation` (zero or more)

4.3 `HeadwordTranslation` object type

The translation equivalent of the headword in one of its senses.

Child of:

- `Sense`

Properties:

- `text` (non-empty string)
  - Can be a single word, a multi-word expression, or indeed any expression in the target language.
- `listingOrder`
  - Can be implicit from the serialization.

Children:

- `TranslationPartOfSpeech` (zero or more)
- `TranslationUsage` (zero or more)
- `TranslationPronunciation` (zero or more)
- `TranslationInflectedForm` (zero or more)

4.4 `TranslationPartOfSpeech` object type

Any of the word classes to which the translation may be assigned, e.g. noun, verb, adjective, etc.

Child of:

- `HeadwordTranslation`

Properties:

- `value` (non-empty string)
  - Can be constrained by the DMLex Controlled Vocabularies Module.
- `listingOrder`
  - Can be implicit from the serialization.

4.5 `TranslationUsage` object type

An indication of some restriction on the use of its parent. The restriction can be pragmatic (time, region, register), semantic (domain, semantic type) or formal ('no plural').

Child of:

- `HeadwordTranslation`
- `TranslationPronunciation`
- `TranslationInflectedForm`

Properties:

- `value` (non-empty string)
  - Can be constrained by the DMLex Controlled Vocabularies Module.
  - Its type (eg. whether register, temporal, geographic etc) can be specified by the by the DMLex Controlled Vocabularies Module.
- `listingOrder`

    - Can be implicit from the serialization.

4.6 `TranslationPronunciation` object type

Information about the pronunciation of its parent.

Child of:

- `HeadwordTranslation`
- `TranslationInflectedForm`

Properties (at least one):

- `transcription` (non-empty string)
- `recording` (string: name or URL of a sound file)
- `listingOrder`
    - Can be implicit from the serialization.

Children:

- `TranslationUsage` (zero or more)

4.7 `TranslationInflectedForm` object type

A form of the inflectional paradigm of its parent.

Child of:

- `HeadwordTranslation`

Properties:

- `label` (non-empty string) e.g. 'plural'
    - Can be constrained by the DMLex Controlled Vocabularies Module.
- `value` (non-empty string)
- `listingOrder`
    - Can be implicit from the serialization.

Children:

- `TranslationUsage` (zero or more)
- `TranslationPronunciation` (zero or more)

4.8 `HeadwordTranslation` object type

The translation equivalent of the headword in one of its senses.

Child of:

- `Sense`

Properties:

- `text` (non-empty string)
    - Can be a single word, a multi-word expression, or indeed any expression in the target language.
- `listingOrder`
    - Can be implicit from the serialization.

Children:

- `TranslationPartOfSpeech` (zero or more)
- `TranslationUsage` (zero or more)
- `TranslationPronunciation` (zero or more)
- `TranslationInflectedForm` (zero or more)

4.9 Extensions to `Example` object type

Additional children:

- `ExampleTranslation` (zero or more)

4.10 `ExampleTranslation` object type

The translation of an example.

Child of:

- `Example`

Properties:

- `text` (non-empty string)
- `listingOrder`
    - Can be implicit from the serialization.

5 Entry Structuring Module

5.1 `SenseGroup` relation type

Represents the fact that a group of senses (all belonging to the same entry) should be grouped when presented to a human user. Typically, when an entry has a large number of senses, it is a convenience to the human user to group them into a smaler number of groups by some broad criterion, such as by part of speech or by semantic similarity.

Participants:

- `Sense` (two or more)

Properties:

- `indicator` (optional, non-empty string)
    - A short statement that indicates the broad meaning that unites the senses in this group and permits their differentiation from other senses in the entry.

5.2 `Subsense` relation type

Represents the fact that one sense (the subordinate sense) should be treated as a subsense of another sense (the subordinate). Both senses belong to the same entry.

Participants:

- the superordinate `Sense` (exactly one)
- the subordinate `Sense` (exactly one)

5.3 `Subentry` relation type

Represents the fact that one entry (= the subordinate entry) should be treated as a subentry inside the sense (= the superordinate sense) of another entry.

Participants:

- the superordinate `Sense` (exactly one)
- the subordinate `Entry` (exactly one)

Properties:

- `listingOrder`
    - Can be implicit from the serialization.

6 Crossreferencing Module

6.1 `Variant` relation type

Represents the fact that two entries are understood by the lexicographer as variants (for example masculine and feminine counterparts, spelling variants).

Participants:

- `Entry` (two or more)

6.2 `Opposition` relation type

Represents the fact that two senses (typically - but not necessarily - belonging to two different entries) have opposite meanings. This includes antonyms, converses and so on.

Participants:

- `Sense` (exactly two)

6.3 `Similarity` relation type

Represents the fact that two or more senses (typically - but not necessarily - belonging to two different entries) have the same or similar meanings. This includes synonyms, near synonyms, immediate hypernyms/hyponyms and so on.

Participants:

- `Sense` (two or more)

6.4 `Pertainment` relation type

Represents the fact that two or more senses (typically - but not necessarily - belonging to two different entries) are related to each other, in ways other than opposition and similarity.

Participants:

- `Sense` (two or more)

7 Inline Markup Module

7.1 `Placeholder` markup type

Marks up a substring inside a headword (or inside a headword translation) which is not part of the expression itself but stands for things that can take its place, or constitutes some kind of meta-notation. Examples:

- `beat [sb.] up`
- `continue [your] studies`

Markup of:

- `headword` property of `Entry`
- `text` property of `HeadwordTranslation`

7.2 `Headword` markup type

Marks up a substring inside an example (or inside an example translation) which corresponds to the headword (or to a translation of the headword).

Markup of:

- `text` property of `Example`
- `text` property of `ExampleTranslation`

8 Controlled Vocabularies Module

This module makes it possible to describe constraints on the values of certain plain-text properties of objects defined in DMLex Core and in DMLex Bilingual Module.

8.1 Extensions to `LexicographicResource` object type

Additional properties:

- `labelLanguage` (IETF language code)
  - The language of the display values of labels.

Additional children:

- `PartOfSpeechLabel` (zero or more)
- `TranslationPartOfSpeechLabel` (zero or more)
- `UsageLabel` (zero or more)

- `TranslationUsageLabel` (zero or more)
- `InflectedFormLabel` (zero or more)
- `TranslationInflectedFormLabel` (zero or more)

8.2 `PartOfSpeechLabel` object type

A `PartOfSpeechLabel` represents one of several allowed values for the `value` property of `PartOfSpeech` objects.

Properties:

- `value` (non-empty string)
- `displayValue` (optional)

Children:

- `LabelMapping` (zero or more)

8.3 `TranslationPartOfSpeechLabel` object type

A `TranslationPartOfSpeechLabel` represents one of several allowed values for the `value` property of `TranslationPartOfSpeech` objects.

Properties:

- `value` (non-empty string)
- `displayValue` (optional)

Children:

- `LabelMapping` (zero or more)

8.4 `UsageLabel` object type

A `UsageLabel` represents one of several allowed values for the `value` property of `Label` objects.

Properties:

- `type` (one of: `normative`, `register`, `temporal`, `geographic`, `sociocultural`, `domain`, `frequency`, `attitude`)
- `value` (non-empty string)
- `displayValue` (optional)

Children:

- `LabelMapping` (zero or more)

8.5 `TranslationUsageLabel` object type

A `TranslationUsageLabel` represents one of several allowed values for the `value` property of `TranslationLabel` objects.

Properties:

- `type` (one of: `normative`, `register`, `temporal`, `geographic`, `sociocultural`, `domain`, `frequency`, `attitude`)
- `value` (non-empty string)
- `displayValue` (optional)

Children:

- `LabelMapping` (zero or more)

8.6 `InflectedFormLabel` object type

An `InflectedFormLabel` represents one of several allowed values for the `label` property of `InflectedForm` objects.

Properties:

- `value` (non-empty string)
- `displayValue` (optional)

Children:

- `LabelMapping` (zero or more)

8.7 `TranslationInflectedFormLabel` object type

A `TranslationInflectedFormLabel` represents one of several allowed values for the `label` property of `TranslationInflectedForm` objects.

Properties:

- `value` (non-empty string)
- `displayValue` (optional)

Children:

- `LabelMapping` (zero or more)

8.8 `LabelMapping` object type

Represents the fact that an item in the controlled vocabulary is equivalent to item provided by en external authority.

Parents:

- `PartOfSpeechLabel`
- `TranslationPartOfSpeechLabel`
- `UsageLabel`
- `TranslationUsageLabel`
- `InflectedFormLabel`
- `TranslationInflectedFormLabel`

Properties:

- `sameAs` (URI)

9 DMLex XML serialization

9.1 Introduction

This serialization defines ... ELABORATE

Note

This serialization specification chiefly describes how... ELABORATE

Warning

There is an important difference or similar ELABORATE

However, … ELABORATE.

Implementers who wish to better access... ELABORATE.

9.2 DMLex namespaces and validation artifacts for its XML serialization

This normative/informative XML serialization of DMLex Version 1.0 makes use of all DMLex namespaces (both core and modules) namespaces: `http://docs.oasis-open.org/lexidma/ns/dmlex-1.0`, and `urn:oasis:names:tc:lexidma:module_01:1.0`, and `other namespace identifiers as necessary`. NAMESPACE SUPPORT IN XML WILL NEED

Validation artifacts [specify type of artifacts if any available at all] for this RDF serialization are available at http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/filename1.filetype1, http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/filename2.filetype1, and http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/filename1.filetype2.

Note

[Potential note content]

.

9.3 Conformance to the XML serialization of DMLex Version 1.0

Note

> Some data categories.... Other data categories …. The below conformance statement is relevant for all data categories expressible in the XML serialization.. There is no interrelation between data categories/ data categories are related as explained in ....

Processing Requirements

- Conformant Processors MUST be able to use and compute at least all the *DMLex Core* data categories encoded in *DMlex XML Instances* as per certain conformance clauses ELABORATE
    - Subrequirement, ELABORATE.
- Conformant *Agents* MUST be DMLex Conformant in the sense of DMLex Application Conformance and also ELABORATE.

  In particular:

    - Conformant *Creators* MUST be capable of *Creating DMlex XML Instances* ELABORATE.
    - Conformant *Enrichers* MUST be capable of *Enriching DMlex XML Instances* with at least ELABORATE.. .
    - Conformant *Modifiers* MUST be capable of updating at least one data category according to its own Constraints and Processing Requirements as specified in ELABORATE.
    - Etc. ELABORATE.

9.4 Other XML serialization provisions

Other provisions. ELABORATE

> **Warning**
>
> ELABORATE.
>
> **Note**
>
> ELABORATE
>
> Another informative pointer.

Processing Requirements

- *Writers* MUST ELABORATE in *DMLex XML Instances*.

9.5 XML serialization elements

DMLex XML serialization uses the following elements:

[comma separated list of element olinks]

9.5.1 Diagram

Legend:

1 = one
+ = one or more
? = zero or one
* = zero, one or more

```
programlisting code to display the diagram
```

9.6 XML serialization attributes

The XML serialization uses the following attributes:

[comma separated list of attribute olinks]

9.7 Example file

Example 1. Example of a couple of DMLex entries RDF serialized

The following example file includes markup related to several *DMLex Core and Module* data categories.

```
<!-- example file metadata -->

programlisting code
```

10 DMLex JSON serialization

10.1 Introduction

This serialization defines ... ELABORATE

Note

This serialization specification chiefly describes how... ELABORATE

Warning

There is an important difference or similar ELABORATE

However, … ELABORATE.

Implementers who wish to better access... ELABORATE.

10.2 DMLex namespaces and validation artifacts for its JSON serialization

This normative/informative JSON serialization of DMLex Version 1.0 makes use of all DMLex namespaces (both core and modules) namespaces: `http://docs.oasis-open.org/lexidma/ns/dmlex-1.0`, and `urn:oasis:names:tc:lexidma:module_01:1.0`, and `other namespace identifiers as necessary`. NAMESPACE SUPPORT IN JSON WILL NEED

Validation artifacts [specify type of artifacts if any available at all] for this RDF serialization are available at http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/filename1.filetype1, http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/filename2.filetype1, and http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/filename1.filetype2.

Note

[Potential note content]

.

10.3 Conformance to the JSON serialization of DMLex Version 1.0

Note

Some data categories.... Other data categories …. The below conformance statement is relevant for all data categories expressible in the JSON serialization.. There is no interrelation between data categories/ data categories are related as explained in ....

Processing Requirements

- Conformant Processors MUST be able to use and compute at least all the *DMLex Core* data categories encoded in *DMlex JSON Instances* as per certain conformance clauses ELABORATE
    - Subrequirement, ELABORATE.
- Conformant *Agents* MUST be DMLex Conformant in the sense of DMLex Application Conformance and also ELABORATE.

  In particular:

    - Conformant *Creators* MUST be capable of *Creating DMlex JSON Instances* ELABORATE.
    - Conformant *Enrichers* MUST be capable of *Enriching DMlex JSON Instances* with at least ELABORATE.. .
    - Conformant *Modifiers* MUST be capable of updating at least one data category according to its own Constraints and Processing Requirements as specified in ELABORATE.
    - Etc. ELABORATE.

10.4 Other JSON serialization provisions

Other provisions. ELABORATE

Warning

ELABORATE.

Note

ELABORATE

Another informative pointer.

Processing Requirements

- *Writers* MUST ELABORATE in *DMLex JSON Instances*.

10.5 JSON serialization objects

DMLex JSON serialization uses the following objects:

[comma separated list of object olinks]

10.5.1 Diagram

Legend:

1 = one
+ = one or more
? = zero or one
* = zero, one or more

```
programlisting code to display the diagram
```

10.6 JSON serialization properties

The JSON serialization uses the following properties:

[comma separated list of properties olinks]

10.7 Example file

Example 2. Example of a couple of DMLex entries RDF serialized

The following example file includes markup related to several *DMLex Core and Module* data categories.

```
<!-- example file metadata -->

programlisting code
```

11 DMLex RDF serialization

11.1 Introduction

This serialization defines ... ELABORATE

Note

This serialization specification chiefly describes how... ELABORATE

Warning

There is an important difference or similar ELABORATE

However, … ELABORATE.

Implementers who wish to better access... ELABORATE.

11.2 DMLex namespaces and validation artifacts for its RDF serialization

This normative/informative RDF serialization of DMLex Version 1.0 makes use of all DMLex namespaces (both core and modules) namespaces: `http://docs.oasis-open.org/lexidma/ns/dmlex-1.0`, and `urn:oasis:names:tc:lexidma:module_01:1.0`, and `other namespace identifiers as necessary.`

Validation artifacts [specify type of artifacts if any available at all] for this JSON serialization are available at http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/filename1.filetype1, http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/filename2.filetype1, and http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/filename1.filetype2.

Note

[Potential note content]

.

11.3 Conformance to the RDF serialization of DMLex Version 1.0

Note

Some Module 01 data categories.... Other data categories …. The below conformance statement is only relevant for data categories .. There is no interrelation between data categories/ data categories are related as explained in ....

Processing Requirements

- Conformant Processors MUST be able to use and compute at least all the *DMLex Core* data categories encoded in *DMlex RDF Instances* as per certain conformance clauses ELABORATE
    - Subrequirement, ELABORATE.
- Conformant *Agents* MUST be DMLex Conformant in the sense of DMLex Application Conformance and also ELABORATE.

  In particular:
    - Conformant *Creators* MUST be capable of *Creating DMlex RDF Instances* ELABORATE.
    - Conformant *Enrichers* MUST be capable of *Enriching DMlex Instances* with at least one of the above specified Module 01 data categories.
    - Conformant *Modifiers* MUST be capable of updating at least one of the above specified Module 01 data categories according to its own Constraints and Processing Requirements as specified in the Module 01 Module.
    - Etc. ELABORATE.

11.4 Other RDF serialization provisions

Other provisions. ELABORATE

Warning

ELABORATE.

Note

ELABORATE

Another informative pointer.

Processing Requirements

- *Writers* MUST ELABORATE in *DMLex RDF Instances*.

11.5 RDF serialization objects

DMLex RDF serialization uses the following objects:

[comma separated list of object olinks]

11.5.1 Diagram

Legend:

1 = one
+ = one or more
? = zero or one
* = zero, one or more

```
programlisting code to display the diagram
```

11.6 RDF serialization attributes

The RDF serialization uses the following attributes:

[comma separated list of object olinks]

11.7 Example file

Example 3. Example of a couple of DMLex entries RDF serialized

The following example file includes markup related to several *DMLex Core and Module* data categories.

```
<!-- example file metadata -->

programlisting code
```

Appendix A References

This appendix contains the normative and informative references that are used in this document. Normative references are specific (identified by date of publication and/or edition number or Version number) and Informative references are either specific or non-specific.

While any hyperlinks included in this appendix were valid at the time of publication, OASIS cannot guarantee their long-term validity.

A.1 Normative references

[**BCP 14**] is a concatenation of [RFC 2119] and [RFC 8174]

[**RFC 2119**] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, *https://www.ietf.org/rfc/rfc2119.txt* IETF (Internet Engineering Task Force) RFC 2119, March 1997.

[**RFC 8174**] B. Leiba, *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*, *https://www.ietf.org/rfc/rfc8174.txt* IETF (Internet Engineering Task Force) RFC 8174, May 2017.

[**BCP 47**] M. Davis, *Tags for Identifying Languages*, *http://tools.ietf.org/html/bcp47* IETF (Internet Engineering Task Force).

[**RFC 3552**] R. Escrola, B. Korver, *Guidelines for Writing RFC Text on Security Considerations*, *https://www.tools.ietf.org/rfc/rfc3552.txt* IETF (Internet Engineering Task Force) RFC 3552, July 2003.

[**EXAMPLE_ABBREV**] N. Surname, A. Surname, *Exampe Title*, *example.org/citetitleExample Citetitle*, Month dd, yyyy.

[**ITS**] David Filip, Shaun McCance, Dave Lewis, Christian Lieske, Arle Lommel, Jirka Kosek, Felix Sasaki, Yves Savourel *Internationalization Tag Set (ITS) Version 2.0*, *http://www.w3.org/TR/its20/* W3C Recommendation 29 October 2013.

[**JSON**] *The JavaScript Object Notation (JSON) Data Interchange Format*, *https://tools.ietf.org/html/rfc8259* IETF RFC 8259 December 2017.

[**NOTE-datetime**] M. Wolf, C. Wicksteed, *Date and Time Formats*, *http://www.w3.org/TR/NOTE-datetime* W3C Note, 15th September 1997.

[**NVDL**] International Standards Organization, *ISO/IEC 19757-4, Information Technology - Document Schema Definition Languages (DSDL) - Part 4: Namespace-based Validation Dispatching Language (NVDL)*, *http://standards.iso.org/ittf/PubliclyAvailableStandards/c038615_ISO_IEC_19757-4_2006(E).zip* ISO, June 1, 2006.

[**RFC 3987**] M. Duerst and M. Suignard, *Internationalized Resource Identifiers (IRIs)*, *https://www.ietf.org/rfc/rfc3987.txt* IETF (Internet Engineering Task Force) RFC 3987, January 2005.

[**RFC 7303**] H. Thompson and C. Lilley, *XML Media Types*, *https://www.tools.ietf.org/html/rfc7303* IETF (Internet Engineering Task Force) RFC 7303, July 2014.

[**Schematron**] International Standards Organization, *ISO/IEC 19757-3, Information Technology - Document Schema Definition Languages (DSDL) - Part 3: Rule-Based Validation — Schematron (Second Edition)*, *http://standards.iso.org/ittf/PubliclyAvailableStandards/c055982_ISO_IEC_19757-3_2016.zip* ISO, January 15, 2016.

[**UAX #9**] M. Davis, A. Lanin, A. Glass, *UNICODE BIDIRECTIONAL ALGORITHM*, *http://www.unicode.org/reports/tr9/tr9-35.html* Unicode Bidirectional Algorithm, May 18, 2016.

[**UAX #15**] M. Davis, K. Whistler, *UNICODE NORMALIZATION FORMS*, *http://www.unicode.org/reports/tr15/tr15-44.html* Unicode Normalization Forms, February 24, 2016.

[**Unicode**] The Unicode Consortium, *The Unicode Standard*, *http://www.unicode.org/versions/Unicode9.0.0/* Mountain View, CA: The Unicode Consortium, June 21, 2016.

[**XLIFF 2.1**] David Filip, Tom Comerford, Soroush Saadatfar, Felix Sasaki, and Yves Savourel, eds. *XLIFF Version 2.0*, *http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html* OASIS Standard 13 February 2018

[**XML**] W3C, *Extensible Markup Language (XML) 1.0*, *http://www.w3.org/TR/xml/* (Fifth Edition) W3C Recommendation 26 November 2008.

[**XML namespace**] W3C, *Schema document for namespace http://www.w3.org/XML/1998/namespace http://www.w3.org/2001/xml.xsd* [http://www.w3.org/2009/01/xml.xsd]. at http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/informativeCopiesOf3rdPartySchemas/w3c/xml.xsd in this distribution

[**XML Catalogs**] Norman Walsh, *XML Catalogs*, *https://www.oasis-open.org/committees/download.php/14809/xml-catalogs.html* OASIS Standard V1.1, 07 October 2005.

[**XML Schema**] W3C, *XML Schema*, *refers to the two part standard comprising [XML Schema Structures] and [XML Schema Datatypes]* (Second Editions) W3C Recommendations 28 October 2004.

[**XML Schema Datatypes**] W3C, *XML Schema Part 2: Datatypes*, [*http://www.w3.org/TR/xmlschema-2/*](http://www.w3.org/TR/xmlschema-2/) (Second Edition) W3C Recommendation 28 October 2004.

[**XML Schema Structures**] W3C, *XML Schema Part 1: Structures*, [*https://www.w3.org/TR/xmlschema-1/*](https://www.w3.org/TR/xmlschema-1/) (Second Edition) W3C Recommendation 28 October 2004.

A.2 Informative references (Informative)

[**LDML**] *Unicode Locale Data Markup Language* [*http://unicode.org/reports/tr35/*](http://unicode.org/reports/tr35/)

[**UAX #29**] M. Davis, *UNICODE TEXT SEGMENTATION*, [*http://www.unicode.org/reports/tr29/*](http://www.unicode.org/reports/tr29/) Unicode text Segmentation.

Appendix B Machine Readable Validation Artifacts (Informative)

CURRENTLY NO VALIDATION ARTIFACTS FORESEEN FOR THE OM.. JUST FOR SERIALIZATIONS

MAY LIST CONFORMANT ARTIFACTS FOR SPECIFIC SERILIZATIONS AT A LATER STAGE

Appendix C Security and privacy considerations

Note

OASIS strongly recommends that Technical Committees consider issues that might affect safety, security, privacy, and/or data protection in implementations of their work products and document these for implementers and adopters. For some purposes, you may find it required, e.g. if you apply for IANA registration.

While it may not be immediately obvious how your work product might make systems vulnerable to attack, most work products, because they involve communications between systems, message formats, or system settings, open potential channels for exploit. For example, IETF [RFC 3552] lists “eavesdropping, replay, message insertion, deletion, modification, and man-in-the-middle” as well as potential denial of service attacks as threats that must be considered and, if appropriate, addressed in IETF RFCs.

In addition to considering and describing foreseeable risks, this section should include guidance on how implementers and adopters can protect against these risks.

We encourage editors and TC members concerned with this subject to read Guidelines for Writing RFC Text on Security Considerations, IETF [RFC 3552], for more information.

Appendix D Specification Change Tracking (Informative)

This appendix will contain tracked changes after the csprd01 phase will have been reached.

Appendix E Acknowledgements (Informative)

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

- Erjavec, Tomaž - JSI
- Filip, David - TCD, ADAPT Centre
- Jakubíček, Miloš - Lexical Computing
- Kernerman, Ilan - K Dictionaries
- Kosem, Iztok - JSI
- Krek, Simon - JSI
- McCrae, John - National University of Ireland Galway
- Měchura, Milan - JSI