Data Model for Lexicography (DMLex), Version 1.0

Working Draft 01

12 March 2022

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. 12 March 2022. 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

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark for above guidance.

Table of Contents

1 Introduction 7
1.1 Glossary 7
1.1.1 Definitions 7
1.1.2 Key concepts 8
2 Conformance 9
3 The Core Specification 10
3.1 Data types 10
3.1.1 `LexicographicResource` 10
3.1.2 `Entry` 11
3.1.3 `Sense` 11
3.1.4 `PartOfSpeech` 12
3.1.5 `Label` 13
3.1.6 `Pronunciation` 13
3.1.7 `InflectedForm` 14
3.1.8 `Example` 14
3.2 Metadata types 15
3.2.1 `InflectedFormCaption` 15
3.2.2 `PartOfSpeechValue` 16
3.2.3 `LabelValue` 17
3.2.4 `ExampleSource` 17
3.2.5 `Mapping` 17
4 Bilingual Module 19
4.1 Extensions to core data types 19
4.1.1 `LexicographicResource` 19
4.1.2 `Sense` 19
4.1.3 `Example` 19
4.2 Data types 19
4.2.1 `HeadwordTranslation` 19
4.2.2 `TranslationPartOfSpeech` 19
4.2.3 `TranslationLabel` 20
4.2.4 `TranslationPronunciation` 20
4.2.5 `TranslationInflectedForm` 20
4.2.6 `HeadwordTranslation` 21
4.2.7 `ExampleTranslation` 21
4.3 Metadata types 21
4.3.1 `TranslationInflectedFormCaption` 21
4.3.2 `TranslationLabelValue` 22
4.3.3 `TranslationPartOfSpeechValue` 22
5 Internal Linking Module 23
5.1 Extensions to core data types 23
5.1.1 `LexicographicResource` 23
5.1.2 `Entry` 23
5.1.3 `Sense` 23
5.2 Relation types 23
5.2.1 `EntrySet` 23
5.2.2 `SenseSet` 24
5.2.3 `SensePair` 25
5.2.4 `SenseTuple` 26
5.2.5 `SenseEntryTuple` 27
5.3 Metadata types 28
5.3.1 `EntrySetCaption` 28
5.3.2 `SenseSetCaption` 28
5.3.3 `SensePairCaption` 28
5.3.4 `SenseTupleCaption` 28
5.3.5 `SenseEntryTupleCaption` 28

6 External Linking Module 30
6.1 Relation types 30
6.1.1 Same 30
6.1.2 Subsumption 30
6.1.3 Overlap 30
7 Inline Markup Module 31
7.1 Marker types 31
7.1.1 PlaceholderMarker 31
7.1.2 HeadwordMarker 31
7.1.3 CollocateMarker 31
8 Etymology Module 32
8.1 Extensions to Entry object type 32
8.2 Etymology object type 32
8.3 EtymDescription 32
8.4 Etymon 32
8.5 EtymTranslation 33
8.6 EtymDate 33
8.7 EtymRelation relation type 33
8.8 Extensions to Mapping object type 34
8.9 EtymonGramLabel object type 34
8.10 EtymonLanguage object type 34
8.11 EtymonType object type 34
8.12 EtymRelationsCaption object type 34
9 DMLex XML serialization 36
9.1 Introduction 36
9.2 DMLex namespaces and validation artifacts for its XML serialization 36
9.3 Conformance to the XML serialization of DMLex Version 1.0 36
9.4 Other XML serialization provisions 37
9.5 XML serialization elements 37
9.5.1 Diagram 37
9.6 XML serialization attributes 37
9.7 Example file 38
10 DMLex JSON serialization 39
10.1 Introduction 39
10.2 DMLex namespaces and validation artifacts for its JSON serialization 39
10.3 Conformance to the JSON serialization of DMLex Version 1.0 39
10.4 Other JSON serialization provisions 40
10.5 JSON serialization objects 40
10.5.1 Diagram 40
10.6 JSON serialization properties 40
10.7 Example file 41
11 DMLex RDF serialization 42
11.1 Introduction 42
11.2 DMLex namespaces and validation artifacts for its RDF serialization 42
11.3 Conformance to the RDF serialization of DMLex Version 1.0 42
11.4 Other RDF serialization provisions 43
11.5 RDF serialization objects 43
11.5.1 Diagram 43
11.6 RDF serialization attributes 43
11.7 Example file 44

Appendixes

A References 45
A.1 Normative references 45
A.2 Informative references (Informative) 46
B Machine Readable Validation Artifacts (Informative) 47
C Security and privacy considerations 48

D Specification Change Tracking (Informative) 49
E Acknowledgements (Informative) 50

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 Data types

3.1.1 `LexicographicResource`

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.

Warning

The correct name of this data type in DMLex is lexicographic resource, not lexical.

Properties:

- `language` (optional, IETF language code)
  - The language of headwords, definitions, examples.

Note

DMLex is based on the assumption that all headwords in a single lexicographic resource are in the same language, and that definitions and examples, if any occur in the resource, are in that language too. The `language` property informs potential users of the resource which language that is.

- `transcriptionScheme` (optional, IETF language code, eg. `en-fonipa` for English IPA)
  - The scheme (e.g. IPA) in which the `transcription` property of `Pronunciation` objects is given.

Note

DMLex is based on the assumption that, if you do use any pronunciation transcriptions in your (monolingual) lexicographic resource, they all follow the same scheme. The `transcriptionScheme` tells potential users of the lexicographic resource which scheme that is.

- `metadataLanguage` (IETF language code)
  - The language of the display captions of metadata items.

Children:

- `Entry` (one or more)
- `PartOfSpeechValue` (zero or more)
- `LabelValue` (zero or more)
- `InflectedFormCaption` (zero or more)
- `Mapping` (zero or more)

Example 1. Example of a lexicographic resource

```
LexicographicResource: language="en"
    Entry: headword="aardvark"
        Sense: ...
        Sense: ...
    Entry: headword="abacus"
        Sense: ...
        Sense: ...
    Entry: headword="abandon"
        Sense: ...
        Sense: ...
```

3.1.2 `Entry`

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 an 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)
- `Label` (zero or more)
- `Pronunciation` (zero or more)
- `InflectedForm` (zero or more)
- `Sense` (zero or more)

Example 2. Example of a lexicographic entry

```
Entry: headword="bank", homographNumber=1
    PartOfSpeech: ...
    Pronunciation: ...
    Sense: ...
    Sense: ...
```

3.1.3 `Sense`

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.

Note

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

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.

    Note

    Indicators are sometimes used in dictionaries as "mini-definitions" instead of or in addition to regular definitions. They are short, one-word or two-word paraphrases of the sense. Their purpose is to allow the (human) user to find the desired sense quickly.
- `definition` (optional, non-empty string)
  - A long statement that describes and/or explains the meaning of a sense.

    Note

    In DMLex, the term definition encompasses not only formal sciency definitions, but also less formal explanations.

Children:

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

Example 3. Example of an entry with two senses, each having both an indicator and a definition

```
Entry: headword="bank"
    Sense: indicator="financial institution", definition = "an institution which..."
        Example: ...
    Sense: indicator="riverside", definition = "an earthen mound which..."
        Example: ...
```

3.1.4 `PartOfSpeech`

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)
  - Must be one of the values defined by instances of `PartOfSpeechValue`.
- `listingOrder`
  - Can be implicit from the serialization.

Note

If you want to model other grammatical properties of the headword besides part of speech, such as gender (of nouns) or aspect (of verbs), the recommended way to do that in DMLex is to conflate them to the part-of-speech label, for example "masculine noun" and "feminine noun", or "perfective verb" and "imperfective verb".

Example 4. Example of a part of speech

```
LexicographicResource
  PartOfSpeechValue: value="n", displayValue="noun"
  Entry: headword="aardvark"
    PartOfSpeech: value="n"
```

3.1.5 `Label`

An indication of some restriction on the use of the lexical item such time (old-fashioned, neologism), region (dialect), register (formal, colloquial), domain (medicine, politics) or grammar (singular-only).

Child of:

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

Properties:

- `value` (non-empty string)
  - Must be one of the values defined by instances of `LabelValue`.
- `listingOrder`
  - Can be implicit from the serialization.

Example 5. Example of a label

```
LexicographicResource
  LabelValue: value="sl", displayValue="slang"
  Entry: headword="bitch"
    Sense: definition="a female dog"
    Sense: definition="a woman who is difficult to get along with"
      Label: value="sl"
```

3.1.6 `Pronunciation`

Information about the pronunciation of its parent.

Child of:

- `Entry`
- `InflectedForm`

Properties:

- At least one of:
  - `transcription` (non-empty string)

    Note

    The scheme of the transcription (e.g. IPA) can be communicated to users of the lexicographic resource through the `transcriptionScheme` property of the `LexicographicResource` object.

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

Children:

- `Label` (zero or more)

Example 6. Example of pronunciation given as transcription

```
Entry: headword="aardvark"
    Pronunciation: transcription="a:rdva:rk"
```

Example 7. Example of pronunciation given as a sound file

```
Entry: headword="aardvark"
    Pronunciation: recording="aardvark.mp3"
```

Example 8. Example of pronunciation given both ways

```
Entry: headword="aardvark"
    Pronunciation: transcription="a:rdva:rk", recording="aardvark.mp3"
```

3.1.7 `InflectedForm`

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

Child of:

- `Entry`

Properties:

- `caption` (non-empty string) e.g. 'plural'
  - Must be one of the values defined by instances of `InlectedFormCaption`.
- `value` (non-empty string)
- `listingOrder`
  - Can be implicit from the serialization.

Children:

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

Example 9. Example of an entry with two inflected forms

```
LexicographicResource: language="ga", displayLanguage="en"
    PartOfSpeechCaption: value="nf", displayValue="feminine noun"
    InflectedFormCaption: value="gs", displayValue="genitive singular"
    InflectedFormCaption: value="pl", displayValue="plural"
    Entry headword="deirfiúr"
        PartOfSpeech: value="nf"
        InflectedForm: caption="gs", value="deirféar"
        InflectedForm: caption="pl", value="deirfiúracha"
```

3.1.8 `Example`

A sentence or other text extract which illustrates the headword being used in the sense.

Child of:

- `Sense`

Properties:

- `text` (non-empty string)
- `listingOrder`
  - Can be implicit from the serialization.
- `sourceIdentity` (optional)
  - Must be one of the values defined by instances of `ExampleSource`.
- `sourceDescription` (optional, non-emty text)
  - A free-form statement about the source of the example.

Example 10. A simple example

```
LexicographicResource: language="en" displayLanguage="en"

  Entry: headword="pack"
    Sense:
      Example: text="I need to pack my bags."
```

Example 11. A sourced example

```
LexicographicResource: language="en" displayLanguage="en"
  ExampleSource value="wodehouse1967" displayValue="P.G. Wodehose (1967) The World of J

  Entry: headword="pack"
    Sense:
      Example: text="Have you started packing yet, Jeeves?" sourceIdentity="wodehouse19
```

Example 12. A sourced example with additional details about where in the source it comes from

```
LexicographicResource: language="en" displayLanguage="en"
  ExampleSource value="wodehouse1967" displayValue="P.G. Wodehose (1967) The World of J

  Entry: headword="pack"
    Sense:
      Example: text="Have you started packing yet, Jeeves?" sourceIdentity="wodehouse19
```

Example 13. An example with ad-hoc attribution

```
LexicographicResource: language="en" displayLanguage="en"

  Entry: headword="pack"
    Sense:
      Example: text="Pack up your knives and go!" sourceDescription="overheard in Dubli
```

3.2 Metadata types

3.2.1 `InflectedFormCaption`

An `InflectedFormCaption` represents one of several allowed values for the `caption` property of `InflectedForm` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

Example 14. Example: restricting the captions of inflected forms

```
LexicographicResource: language="ga", displayLanguage="en"
    PartOfSpeechCaption: value="nf", displayValue="feminine noun"
    InflectedFormCaption: value="gs", displayValue="genitive singular"
    InflectedFormCaption: value="pl", displayValue="plural"
    Entry headword="deirfiúr"
        PartOfSpeech: value="nf"
        InflectedForm: caption="gs", value="deirféar"
        InflectedForm: caption="pl", value="deirfiúracha"
```

Example 15. Example: mapping the captions of inflected forms to external inventories

```
InflectedFormCaption: value="gs", displayValue="genitive singular"
    Mapping: sameAs="http://www.lexinfo.net/ontology/3.0/lexinfo#singular"
    Mapping: sameAs= "http://www.lexinfo.net/ontology/3.0/lexinfo#genitiveCase"
InflectedFormCaption: value="pl", displayValue="plural"
    Mapping: sameAs="http://www.lexinfo.net/ontology/3.0/lexinfo#plural"
```

3.2.2 `PartOfSpeechValue`

Represents one of several allowed values for the `value` property of `PartOfSpeech` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

Example 16. Example: restricting part-of-speech values

```
LexicographicResource: displayLanguage="en"
    PartOfSpeechValue: value="n", displayValue="noun"
    PartOfSpeechValue: value="v", displayValue="verb"
    PartOfSpeechValue: value="adj", displayValue= "adjective"
    Entry: headword="aardvark"
        PartOfSpeech: value="n"
    Entry: headword="speak"
        PartOfSpeech: value="v"
    Entry: headword="small"
        PartOfSpeech: value="adj"
```

Example 17. Example: mapping part-of-speech values to external inventories

```
PartOfSpeechValue: value="n", displayValue="noun"
    Mapping: sameAs="http://www.lexinfo.net/ontology/3.0/lexinfo#noun"
```

Example 18. Example: mapping complex part-of-speech values to external inventories

```
PartOfSpeechValue: value="Nma", displayValue="masculine animate noun"
    Mapping: sameAs="http://www.lexinfo.net/ontology/3.0/lexinfo#noun"
    Mapping: sameAs="http://www.lexinfo.net/ontology/3.0/lexinfo#masculine"
    Mapping: sameAs="http://www.lexinfo.net/ontology/3.0/lexinfo#animate"
```

3.2.3 `LabelValue`

Represents one of several allowed values for the `value` property of `Label` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

Example 19. Example: restricting the values of Label objects

```
LexicographicResource: displayLanguage="en"
    LabelValue: value="sl", displayValue="slang"
    LabelValue: value="vul", displayValue="vulgar"
```

Example 20. Example: mapping the values of Label objects to external ontologies

```
LexicographicResource: displayLanguage="en"
    LabelValue: value="sl", displayValue="slang"
        Mapping: sameAs="http://www.lexinfo.net/ontology/3.0/lexinfo#slangRegister"
    LabelValue: value="vul", displayValue="vulgar"
        Mapping: sameAs="http://www.lexinfo.net/ontology/3.0/lexinfo#vulgarRegister"
```

3.2.4 `ExampleSource`

Represents one of several allowed sources for the `sourceIdentity` property of `Example` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

3.2.5 `Mapping`

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

Parents:

- `PartOfSpeechCaption`
- `TranslationPartOfSpeechCaption`

- `LabelCaption`
- `EntrySetCaption`
- `SenseSetCaption`
- `SensePairCaption`
- `SenseTupleCaption`
- `TranslationLabelCaption`
- `InflectedFormCaption`
- `TranslationInflectedFormCaption`

Properties:

- `sameAs` (URI)

4 Bilingual Module

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

4.1 Extensions to core data types

4.1.1 `LexicographicResource`

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.

4.1.2 `Sense`

Additional children:

- `HeadwordTranslation` (zero or more)

4.1.3 `Example`

Additional children:

- `ExampleTranslation` (zero or more)

4.2 Data types

4.2.1 `HeadwordTranslation`

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)
- `TranslationLabel` (zero or more)
- `TranslationPronunciation` (zero or more)
- `TranslationInflectedForm` (zero or more)

4.2.2 `TranslationPartOfSpeech`

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)
  - Must be one of the values defined by instances of `TranslationPartOfSpeechValue`.
- `listingOrder`
  - Can be implicit from the serialization.

4.2.3 `TranslationLabel`

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)
  - Must be one of the values defined by instances of `TranslaionLabelValue`.
- `listingOrder`
  - Can be implicit from the serialization.

4.2.4 `TranslationPronunciation`

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:

- `TranslationLabel` (zero or more)

4.2.5 `TranslationInflectedForm`

A form of the inflectional paradigm of its parent.

Child of:

- `HeadwordTranslation`

Properties:

- `label` (non-empty string) e.g. 'plural'
  - Must be one of the values defined by instances of `TranslationInlectedFormCaption`.
- `value` (non-empty string)

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

Children:

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

4.2.6 `HeadwordTranslation`

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)
- `TranslationLabel` (zero or more)
- `TranslationPronunciation` (zero or more)
- `TranslationInflectedForm` (zero or more)

4.2.7 `ExampleTranslation`

The translation of an example.

Child of:

- `Example`

Properties:

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

4.3 Metadata types

4.3.1 `TranslationInflectedFormCaption`

Represents one of several allowed values for the `caption` property of `TranslationInflectedForm` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

4.3.2 `TranslationLabelValue`

Represents one of several allowed values for the `value` property of `TranslationLabel` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

4.3.3 `TranslationPartOfSpeechValue`

Represents one of several allowed values for the `value` property of `TranslationPartOfSpeech` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

5 Internal Linking Module

Used to construct clickable (or otherwise navigable) hyperlinks between objects within the same `LexicographicResource`.

5.1 Extensions to core data types

5.1.1 `LexicographicResource`

Additional children:

- `EntrySetCaption` (zero or more)
- `SenseSetCaption` (zero or more)
- `SensePairCaption` (zero or more)
- `SenseTupleCaption` (zero or more)
- `SenseEntryTupleCaption` (zero or more)

5.1.2 `Entry`

Additional properties:

- `id`

5.1.3 `Sense`

Additional properties:

- `id`

5.2 Relation types

5.2.1 `EntrySet`

Represents an unordered set of two or more entries. Recommended for encoding variants such for example masculine and feminine counterparts or spelling variants.

Properties:

- `caption` (non-empty string) eg. "variants"
  - Must be one of the values defined by instances of `EntrySetCaption`.

Participants (all in the same `LexicographicResource`):

- `entry` (two or more)
  - Reference to an `Entry` object.

Example 21. Encoding variants

```
LexicographicResource: language="en", displayLanguage="en"
    LabelValue: value="eu", displayValue="European English"
    LabelValue: value="am", displayValue="American English"
    EntrySetCaption value="vars" displayValue="variants"

    Entry id="#colour" headword="colour"
        Label: value="eu"

    Entry id="#color" headword="color"
        Label: value="am"

    EntrySet caption="vars"
 entry: #colour
 entry: #color
```

Example 22. Encoding gender pairs

```
LexicographicResource: language="en", displayLanguage="en"
    EntrySetCaption value="gpair" displayValue="gender pair"

    Entry id="#actor" headword="actor"

    Entry id="#actress" headword="actress"

    EntrySet caption="gpair"
 entry: #actor
 entry: #actress
```

Example 23. Encoding homonyms

```
LexicographicResource: language="de" displayLanguage="en" translationLanguage="en"
    EntrySetCaption value="homo" displayValue="homonyms"
    PartOfSpeechValue value="n-masc" displayValue="masculine noun"
    PartOfSpeechValue value="n-fem" displayValue="feminine noun"

    Entry id="#der-see" headword="See"
 PartOfSpeech: value="n-masc"
 Sense: id="#der-see-1"
  Translation: lake

    Entry id="#die-see" headword="See"
 PartOfSpeech: value="n-fem"
 Sense: id="#die-see-1"
  Translation: see

    EntrySet caption="homo"
 entry: #der-see
 entry: #die-see
```

5.2.2 SenseSet

Represents an unordered set of two or more senses (typically - but not necessarily - belonging to two or more different entries). Recommended for encoding synonyms, near synonyms etc.

Properties:

- `caption` (non-empty string) eg. "synonyms"
  - Must be one of the values defined by instances of `SenseSetCaption`.

Participants (all in the same `LexicographicResource`):

- `sense` (two or more)
  - Reference to a `Sense` object.

Example 24. Encoding near synonyms

```
LexicographicResource: language="de" displayLanguage="en" translationLanguage="en"
    EntrySetCaption value="nsyn" displayValue="near synonyms"
    PartOfSpeechValue value="n-masc" displayValue="masculine noun"
    PartOfSpeechValue value="n-fem" displayValue="feminine noun"
    PartOfSpeechValue value="n-neut" displayValue="neuter noun"

    Entry id="#die-see" headword="See"
       PartOfSpeech: value="n-fem"
 Sense: id="#die-see-1"
            Translation: "see"

    Entry id="#das-meer" headword="Meer"
       PartOfSpeech: value="n-neut"
 Sense: id="#das-meer-1"
            Translation: "see"

    Entry id="#der-ozean" headword="Ozean"
       PartOfSpeech: value="n-masc"
 Sense: id="#der-ozean-1"
            Translation: "ocean"

    SenseSet caption="nsyn"
 sense: #die-see-1
 sense: #das-meer-1
 sense: #der-ozean-1
```

5.2.3 `SensePair`

Represents an undirected pair of two senses (typically - but not necessarily - belonging to two different entries). Recommended for representing pairs of antonyms, opposites, converses and so on.

Properties:

- `caption` (non-empty string) eg. "antonyms"
  - Must be one of the values defined by instances of `SensePairCaption`.

Participants (all in the same `LexicographicResource`):

- `sense` (exacly two)
  - Reference to a `Sense` object.

Example 25. Encoding antonyms

```
LexicographicResource: language="en" displayLanguage="en"
    EntrySetCaption value="anto" displayValue="antonyms"

    Entry id="#buy" headword="buy"
 Sense: id="#buy-1"

    Entry id="#sell" headword="sell"
 Sense: id="#see-1"

    SensePair caption="anto"
 entry: #buy-1
 entry: #sell-1
```

5.2.4 `SenseTuple`

Represents a directed pair of two senses. When the two senses belong to two different entries, then this type is recommended for representing pairs of hypernym/hyponym, broader/narrower, holonym/meronym. When the two senses belong to the same entry, then this type is recommended for representing the relationship between a sense and a subsense.

Properties:

- `caption` (non-empty string) eg. "hypernymy"
  - Must be one of the values defined by instances of `SenseTupleCaption`.

Participants (all in the same `LexicographicResource`):

- `superordinate` (exacly one)
  - Reference to a `Sense` object.
- `subordinate` (exacly one)
  - Reference to a `Sense` object.

Example 26. Encoding meronymy

```
LexicographicResource: language="en" displayLanguage="en"
  EntrySetCaption value="mero" displayValue="meronymy"

  Entry id="#glasses" headword="glasses"
    Sense: id="#glasses-1"

  Entry id="#microscope" headword="microscope"
    Sense: id="#microscope-1"

  Entry id="#lense" headword="lense"
    Sense: id="#lense-1"

  SenseTuple caption="mero"
    superordinate: #glasses-1
    subordinate: #lense-1

  SenseTuple caption="mero"
    superordinate: #microscope-1
    subordinate: #lense-1
```

5.2.5 SenseEntryTuple

Represents a directed pair of a sense and an entry, such that the entry does not contain the sense. This type is recommended to be used for situations when the entry is expected to be show to a human user as a subentry inside the sense of another entry.

Properties:

- `caption` (non-empty string) eg. "subentry"
  - Must be one of the values defined by instances of `SenseEntryTupleCaption`.

Participants (all in the same `LexicographicResource`):

- `superordinate` (exacly one)
  - Reference to a `Sense` object.
- `subordinate` (exacly one)
  - Reference to an `Entry` object.

Example 27. Example: encoding a subentry at subsense level

In this example, when the entry for safe is being shown to a human user, the entry for better safe than sorry is supposed to appear as a subentry at at the end of the first sense.

```
LexicographicResource: language="en", displayLanguage="en"
  SenseEntryTupleCaption value="idiosub" displayValue="idiom subentry"

  Entry: id="#safe" headword="safe"
    Sense: id="#safe-1" indicator="protected from harm"
      Example: value="It isn't safe to park here."
    Sense: id="#safe-2" indicator="not likely to cause harm"
      Example: value="Is the ride safe for a small child?"

  Entry: id="#bsts" headword="better safe than sorry"
    Sense: definition="you should be careful even if it seems unnecessary"

  SenseEntryTuple: caption: "idiosub"
    superordinate: #safe-1
    subordinate: #bsts
```

Example 28. Example: encoding a subentry at sense level

In this example, when the entry for bible is being shown to the human user, the entry for the Bible is supposed to be appear there as a subentry, as if it were its first sense.

```
LexicographicResource: language="en", displayLanguage="en"
  SenseEntryTupleCaption value="propnamesub" displayValue="proper name subentry"

  Entry: id="#the-bible" headword="the Bible"
    Sense: id="the-bible-1" definition="the book considered holy by Christians..."

  Entry: id="#bible" headword="bible"
    Sense: id="#bible-1" //empty sense, serves as a container for "the Bible"
    Sense: id="#bible-2" definition="a book considered important..."

  SenseEntryTuple: caption="propnamesub"
    superordinate: #bible-1
    subordinate: #the-bible
```

5.3 Metadata types

5.3.1 `EntrySetCaption`

Represents one of several allowed values for the `caption` property of `EntrySet` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

5.3.2 `SenseSetCaption`

Represents one of several allowed values for the `caption` property of `SenseSet` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

5.3.3 `SensePairCaption`

Represents one of several allowed values for the `caption` property of `SensePair` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

5.3.4 `SenseTupleCaption`

Represents one of several allowed values for the `caption` property of `SenseTuple` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

5.3.5 `SenseEntryTupleCaption`

Represents one of several allowed values for the `caption` property of `SenseEntryTuple` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

6 External Linking Module

Used to construct links between objects from different lexicographic resources.

6.1 Relation types

6.1.1 `Same`

Represents the fact that two or more senses from two or more lexicographic resources are semantically the same.

Participants:

- `Sense` (two or more, no two of which belong to the same `LexicographicResource`)

6.1.2 `Subsumption`

Represents the fact that one sense from one lexicographic resource is semantically wholly contained inside another sense from another lexicographic resource.

Participants:

- the broader `Sense` (exactly one)
- the narrower `Sense` (exactly one)

The two senses must not be within the `LexicographicResource`.

6.1.3 `Overlap`

Represents the fact that two or more senses from two or more lexicographic resources overlap semantically, without either one being semantically wholly contained inside the other.

Participants:

- `Sense` (two or more, no two of which belong to the same `LexicographicResource`)

7 Inline Markup Module

7.1 Marker types

7.1.1 `PlaceholderMarker`

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`

7.1.2 `HeadwordMarker`

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`

7.1.3 `CollocateMarker`

Marks up a substring inside an example (or inside an example translation) where an important collocate of the headword (or of its translation) occurs.

Markup of:

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

Properties:

- `lemma`
  - Optional. The lemmatized form of the collocate.

Example 29. Marking up the headword and a collocate inside an example

```
Example: value="The coroner performed an autopsy."
  HeadwordMarker: value=first occurence of "autopsy"
  CollocateMarker: value=first occurence of "performed" lemma="perform"
```

8 Etymology Module

Extends DMLex Core to support the encoding of etymological information in dictionaries.

8.1 Extensions to `Entry` object type

Additional children:

- `Etymology` (zero or more)

8.2 `Etymology` object type

An `Etymology` represents the historical derivation of a word.

Children:

- `EtymDescription` (zero or more)
- `Etymon` (zero or more)
- `MultiEtymon` (zero or more)

8.3 `EtymDescription`

A plain text form of the etymology, which may contain notes about the etymology. This may be used instead or alongside a structured set of `Etymons`. The goal of this is to capture the textual description of etymologies for retrodigitised dictionaries.

Child of:

- `Etymology`

Properties:

- `text` (non-empty string, required) the text of the etymological description

8.4 `Etymon`

Marks up a form that is the etymological origin of the entry or another etymologically related form

Child of:

- `Etymology`
- `MultiEtymon`

Properties:

- `text` (non-empty string, required) the etymological form
- `type` (non-empty string, optional) the type of the etymology. Typical values include `derivation`, `cognate` or `borrowing`. Values should be specified with a `EtymonType` mapping.
- `language` (string, required) the language of the origin form. Values for languages should be specified with a `EtymonLanguage` mapping.
- `note` (non-empty string, optional) extra information about the language, for example the precise time (e.g., 16th century) or geographical variants
- `reconstructed` (boolean, optional) indicates that the form is reconstructed and not attested in any corpus
- `gramLabel` (string, optional) a label that indicates a particular grammatical form in the same way as for `InflectedForm`. Values should be specified with a `EtymGramLabel` mapping.

- `href` (url, optional) a reference to a source for this etymology in an external resource
- `compoundOrder` (non-negative integer, required for children of `MultiEtymon` only) if this etymon is part of a multietymon this indicates the order of this etymon. This value may be implicit in some serializations.

Children:

- `EtymTranslation` (zero or more)
- `EtymDate` (zero or more)
- `EtymRelation` (zero or more)

8.5 `EtymTranslation`

Gives a translation of a historical or borrowed form in the language of the dictionary

Child of:

- `Etymon`

Properties:

- `text` (non-empty string, required) the translation of the etymological form in the language of the dictionary

8.6 `EtymDate`

Gives a period of time for which an etymology was valid

Child of:

- `Etymon`

Properties:

- `start` (non-empty string, optional) the start date of when this etymon was used. This can be a year in the Gregorian calendar or an approximate value.
- `end` (non-empty string, optional) the end date of when this etymon was used. This can be a year in the Gregorian calendar or an approximate value.

8.7 `EtymRelation` relation type

Indicates a relation between two etymons

Child of:

- `Etymon`

Properties:

- `caption` (non-empty string, required) indicates the relationship between the etymons. The most typical value is `derivative` indicating that one etymon is a later form of another.
  - Can be constrained and mapped to external authorities using a controlled vocabulary.

Participants (all in the same `LexicographicResource`):

- `etymons` (exacly two)
  - Reference to a `Etymon` object.

8.8 Extensions to `Mapping` object type

Additional children:

- `EtymonGramLabel` (zero or more)
- `EtymonLanguage` (zero or more)
- `EtymonType` (zero or more)
- `EtymRelationCaption` (zero or more)

8.9 `EtymonGramLabel` object type

Represents one of several allowed values for the `gramLabel` property of `Etymon` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

8.10 `EtymonLanguage` object type

Represents one of several allowed values for the `language` property of `Etymon` objects. Note that as etymologies frequently refer to languages that are not covered by ISO standards it is necessary to define codes for all languages used in definitions.

Properties:

- `value` (non-empty string)
- `iso639-1` (non-empty string) the code in the ISO 639-1 standard if it exists
- `iso639-3` (non-empty string) the code in the ISO 639-3 standard if it exists
- `displayValue` (optional)

Children:

- `Mapping` (zero or more)

8.11 `EtymonType` object type

Represents one of several allowed values for the `type` property of `Etymon` objects.

Properties:

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

Children:

- `Mapping` (zero or more)

8.12 `EtymRelationsCaption` object type

Represents one of several allowed values for the `caption` property of `EtymRelation` objects.

Properties:

- `value` (non-empty string)

- `displayValue` (optional)

Children:

- `Mapping` (zero or more)

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 30. 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 31. 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 32. 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 [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 [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 [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/* (Second Edition) W3C Recommendation 28 October 2004.

[**XML Schema Structures**] W3C, *XML Schema Part 1: Structures*, *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/*

[**UAX #29**] M. Davis, *UNICODE TEXT SEGMENTATION*, *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
- Jakubí#ek, Miloš - Lexical Computing
- Kernerman, Ilan - K Dictionaries
- Kosem, Iztok - JSI
- Krek, Simon - JSI
- McCrae, John - National University of Ireland Galway
- M#chura, Milan - JSI