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 XML Localisation Interchange File Format (XLIFF) 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

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 6
1.1 Glossary 6
1.1.1 Definitions 6
1.1.2 Key concepts 7
2 Conformance 8
3 The Core Specification 9
3.1 General Processing Requirements 9
3.2 Objects 9
3.2.1 Tree Structure 9
3.2.2 Structural Objects 9
3.2.3 Inline Elements 11
3.3 Attributes 18
3.3.1 DMLex Attributes 18
3.3.2 XML namespace 21
3.4 Inline Content 22
3.4.1 Text 23
3.4.2 Inline Codes 23
3.4.3 Annotations 23
3.4.4 White Spaces 24
3.4.5 Bidirectional Text 24
3.4.6 Content Comparison 25
3.5 Extension Mechanisms 25
3.5.1 Extension Points 25
3.5.2 Constraints 26
3.5.3 Processing Requirements 26
4 The DMLex Modules Specifications 27
4.1 Module 01 content model 27
4.1.1 Introduction 27
4.1.2 Module namespace and validation artifacts 27
4.1.3 Module fragment identification prefix 27
4.1.4 Conformance to the Module 01 specification 28
4.1.5 Module 01 functionality 01 28
4.1.6 Module 01 data categories defined in the Module 01 Module 29
4.1.7 Module objects 29
4.1.8 Module Attributes 31
4.1.9 Example file 32
5 DMLex XML serialization 33
5.1 Introduction 33
5.2 DMLex namespaces and validation artifacts for its XML serialization 33
5.3 Conformance to the XML serialization of DMLex Version 1.0 33
5.4 Other XML serialization provisions 34
5.5 XML serialization elements 34
5.5.1 Diagram 34
5.6 XML serialization attributes 34
5.7 Example file 35
6 DMLex JSON serialization 36
6.1 Introduction 36
6.2 DMLex namespaces and validation artifacts for its JSON serialization 36
6.3 Conformance to the JSON serialization of DMLex Version 1.0 36
6.4 Other JSON serialization provisions 37
6.5 JSON serialization objects 37
6.5.1 Diagram 37
6.6 JSON serialization properties 37
6.7 Example file 38
7 DMLex RDF serialization 39

7.1 Introduction 39
7.2 DMLex namespaces and validation artifacts for its RDF serialization 39
7.3 Conformance to the RDF serialization of DMLex Version 1.0 39
7.4 Other RDF serialization provisions 40
7.5 RDF serialization objects 40
7.5.1 Diagram 40
7.6 RDF serialization attributes 40
7.7 Example file 41

Appendixes

A References 42
A.1 Normative references 42
A.2 Informative references (Informative) 43
B Machine Readable Validation Artifacts (Informative) 44
B.1 Tree diagram of validation artifacts 44
C Security and privacy considerations 45
D Specification Change Tracking (Informative) 46
E Acknowledgements (Informative) 47

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

[ELABORATE]

3.1 General Processing Requirements

- An Agent processing a valid DMLex Instance that contains only objects and attributes from namespaces that correspond to DMLex-defined elements and attributes that it cannot handle MUST preserve those objects and properties.
- An Agent processing a valid DMLex Instance that contains custom objects (extension objects from namespaces that are not DMLex-defined) that it cannot handle SHOULD preserve those objects and attributes.

3.2 Objects

This section contains a description of all objects used in DMLex Core.

3.2.1 Tree Structure

Legend:

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

```
dmlex
|
+---entry +
  |
  [structural diagram]
```

3.2.2 Structural Objects

The structural objects used in DMLex Core are: `dmlex`, `entry`, `<notes>`, `<note>`, `<data>`, [other object olinks]

3.2.2.1 dmelx

Highest possible level root object for DMLex Instances.

Contains:

- One or more `entry` elements

Attributes:

- `version`, REQUIRED
- `xml:lang`, OPTIONAL
- `srcLang`, OPTIONAL
- `trgLang`, OPTIONAL
- `xml:space`, OPTIONAL
- attributes from other namespaces, OPTIONAL

Constraints

- The attributes `srcLang` and `trgLang` MUST be both either set or not set.
- The attribute `xml:lang` MUST be used if and only if the pair of `trgLang` and `srcLang` is not used.

3.2.2.2 entry

Static container for a dynamic structure of lexicographic objects and attributes encoding lexicographic payload and metadata.

Contains:

- elements from other namespaces, OPTIONAL
- Zero or one `<notes>` elements followed by
- object structure ELABORATE in this & subsequent simple list members.

Attributes:

- `id`, REQUIRED
- `name`, OPTIONAL
- `srcDir`, OPTIONAL
- `trgDir`, OPTIONAL
- `xml:space`, OPTIONAL
- `type`, OPTIONAL
- attributes from other namespaces, OPTIONAL

Constraints

- An `entry` MUST contain ELABORATE.
- The following DMLex Module objects are explicitly allowed by the wildcard `other`:

  - Zero or one `<md01:object_wrapper_01>` objects
  - Zero or one `<md01:object_wrapper_02>` objects

- Module and Extension elements MAY be used in any order.
- The following DMLex Module attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `urn:oasis:names:tc:xliff:fs:2.0`, OPTIONAL, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace `http://www.w3.org/2005/11/its`, OPTIONAL, provided that the Constraints specified in the ITS Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:itsm:2.1`, OPTIONAL, provided that the Constraints specified in the ITS Module are met.

3.2.2.3 notes

Collection of comments.

Contains:

- One or more `<note>` elements

3.2.2.4 note

This is a DMLex specific way how to present end user readable comments and annotations. A note can contain information about list olinks of objects to which notes can pertain.

Contains:

- Text

Attributes:

- id, OPTIONAL
- appliesTo, OPTIONAL
- category, OPTIONAL
- priority, OPTIONAL
- attributes from other namespaces, OPTIONAL

Constraints

- The following DMLex Module attributes are explicitly allowed by the wildcard `other`:

  - fs:fs, OPTIONAL
  - fs:subFs, OPTIONAL

3.2.2.5 data

Storage for the original data of an inline code.

Contains:

- Non-translatable text
- Zero, one or more <cp> elements.

Non-translatable text and <cp> elements MAY appear in any order.

Attributes:

- id, REQUIRED
- dir, OPTIONAL
- xml:space, OPTIONAL, the value is restricted to `preserve` on this element

3.2.3 Inline Elements

The XLIFF Core inline elements at the <source> or <target> level are: <cp>, <ph>, <pc>, <sc>, <ec>, <mrk>, <sm> and .

The elements at the <unit> level directly related to inline elements are: <originalData> and <data>.

3.2.3.1 cp

Represents a Unicode character that is invalid in XML.

Contains:

This element is always empty.

Parents:

<data>, <mrk>, <source>, <target> and <pc>

Attributes:

- hex, REQUIRED

Example:

```
<unit id="1">
  <segment>
```

```
    <source>Ctrl+C=<cp hex="0003"/></source>
  </segment>
</unit>
```

The example above shows a character U+0003 (Control C) as it has to be represented in XLIFF.

Processing Requirements

- Writers MUST encode all invalid XML characters of the content using `<cp>`.
- Writers MUST NOT encode valid XML characters of the content using `<cp>`.

3.2.3.2 ph

Represents a standalone code of the original format.

Contains:

This element is always empty.

Parents:

`<source>`, `<target>`, `<pc>` and `<mrk>`

Attributes:

- `canCopy`, OPTIONAL
- `canDelete`, OPTIONAL
- `canReorder`, OPTIONAL
- `copyOf`, OPTIONAL
- `disp`, OPTIONAL
- `equiv`, OPTIONAL
- `id`, REQUIRED.
- `dataRef`, OPTIONAL
- `subFlows`, OPTIONAL
- `subType`, OPTIONAL
- `type`, OPTIONAL
- attributes from other namespaces, OPTIONAL

Example:

```
<unit id="1">
  <originalData>
    <data id="d1">%d</data>
    <data id="d2">&lt;br/></data>
  </originalData>
  <segment>
    <source>Number of entries: <ph id="1" dataRef="d1" /><ph id="2"
        dataRef="d2"/>(These entries are only the ones matching the
        current filter settings)</source>
  </segment>
</unit>
```

Constraints

- The following XLIFF Module attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `urn:oasis:names:tc:xliff:fs:2.0`, OPTIONAL, provided that the Constraints specified in the Format Style Module are met.

- attributes from the namespace `urn:oasis:names:tc:xliff:sizerestriction:2.0`, OPTIONAL, provided that the Constraints specified in the Size and Length Restriction Module are met.

- No other attributes MUST be used.

Processing Requirements

- Extractors MUST NOT use the <ph> element to represent spanning codes.

  Rationale: Using a standalone placeholder code for a spanning code does not allow for controlling the span (for instance tag order and data integrity) when Modifying inline content and is in direct contradiction to the business logic described in Representation of the codes and normative statements included in Usage of <pc> and <sc>/<ec>

  ## Note

  It is possible although not advised to use <ph> to mask non translatable inline content. The preferred way of protecting portions of inline content from translation is the Core Translate Annotation. See also discussion in the ITS Module section on representing translatability inline..

3.2.3.3 pc

Represents a well-formed spanning original code.

Contains:

- Text
- Zero, one or more `<cp>` elements
- Zero, one or more `<ph>` elements
- Zero, one or more `<pc>` elements
- Zero, one or more `<sc>` elements
- Zero, one or more `<ec>` elements
- Zero, one or more `<mrk>` elements
- Zero, one or more `<sm>` elements
- Zero, one or more `<em>` elements

Text and inline elements may appear in any order.

Parents:

- `<source>`
- `<target>`
- `<pc>`
-`<mrk>`

Attributes:

- `canCopy`, OPTIONAL
- `canDelete`, OPTIONAL
- `canOverlap`, OPTIONAL
- `canReorder`, OPTIONAL
- `copyOf`, OPTIONAL
- `dispEnd`, OPTIONAL
- `dispStart`, OPTIONAL
- `equivEnd`, OPTIONAL
- `equivStart`, OPTIONAL
- `id`, REQUIRED
- `dataRefEnd`, OPTIONAL
- `dataRefStart`, OPTIONAL
- `subFlowsEnd`, OPTIONAL

- subFlowsStart, OPTIONAL
- subType, OPTIONAL
- type, OPTIONAL
- dir, OPTIONAL
- attributes from other namespaces, OPTIONAL

Example:

```
<unit id="1">
  <originalData>
    <data id="1">&lt;B&gt;</data>
    <data id="2">&lt;/B&gt;</data>
  </originalData>
  <segment><pc id="1" dataRefStart="1" dataRefEnd="2">
      Important</pc> text</source></segment>
</unit>
```

Constraints

- The following XLIFF Module attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `urn:oasis:names:tc:xliff:fs:2.0`, OPTIONAL, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:sizerestriction:2.0`, OPTIONAL, provided that the Constraints specified in the Size and Length Restriction Module are met.

- No other attributes MUST be used.

Processing Requirements

- Extractors MUST NOT use the <pc> element to represent standalone codes.

  Rationale: Using a spanning code for a standalone code can easily result in having text inside a span where the original format does not allow it.

3.2.3.4 sc

Start of a spanning original code.

Contains:

This element is always empty.

Parents:

<source>, <target>, <pc> and <mrk>

Attributes:

- canCopy, OPTIONAL
- canDelete, OPTIONAL
- canOverlap, OPTIONAL
- canReorder, OPTIONAL
- copyOf, OPTIONAL
- dataRef, OPTIONAL
- dir, OPTIONAL
- disp, OPTIONAL
- equiv, OPTIONAL

- id, REQUIRED
- isolated, OPTIONAL
- subFlows, OPTIONAL
- subType, OPTIONAL
- type, OPTIONAL
- attributes from other namespaces, OPTIONAL

Example:

```
<unit id="1">
  <segment>
    <source><sc id="1" type="fmt" subType="xlf:b"/>
        First sentence. </source>
  </segment>
  <segment>
    <source>Second sentence.<ec startRef="1" type="fmt"
        subType="xlf:b"/></source>
  </segment>
</unit>
```

Constraints

- The following XLIFF Module attributes are explicitly allowed by the wildcard other:

  - attributes from the namespace urn:oasis:names:tc:xliff:fs:2.0, OPTIONAL, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace urn:oasis:names:tc:xliff:sizerestriction:2.0, OPTIONAL, provided that the Constraints specified in the Size and Length Restriction Module are met.

- No other attributes MUST be used.
- The values of the attributes canCopy, canDelete, canReorder and canOverlap MUST be the same as the values the ones in the <ec> element corresponding to this start code.
- The attribute isolated MUST be set to yes if and only if the <ec> element corresponding to this start marker is not in the same <unit>, and set to no otherwise.

Processing Requirements

- Extractors MUST NOT use the <sc> / <ec> pair to represent standalone codes.

  Rationale: Using a spanning code for a standalone code can easily result in having text inside a span where the original format does not allow it.

3.2.3.5 ec

End of a spanning original code.

Contains:

This element is always empty.

Parents:

<source>, <target>, <pc> and <mrk>

Attributes:

- canCopy, OPTIONAL

- canDelete, OPTIONAL
- canOverlap, OPTIONAL
- canReorder, OPTIONAL
- copyOf, OPTIONAL
- dataRef, OPTIONAL
- dir, OPTIONAL
- disp, OPTIONAL
- equiv, OPTIONAL
- id, OPTIONAL
- isolated, OPTIONAL
- startRef, OPTIONAL
- subFlows, OPTIONAL
- subType, OPTIONAL
- type, OPTIONAL
- attributes from other namespaces, OPTIONAL

Example:

```
<unit id="1">
  <originalData>
    <data id="d1">\b </data>
    <data id="d2">\i </data>
    <data id="d3">\b0 </data>
    <data id="d4">\i0 </data>
  </originalData>
  <segment>
    <source>Text in <sc id="1" dataRef="d1"/>bold <sc id="2"
        dataRef="d2"/> and<ec startRef="1" dataRef="d3"/>
         italics<ec startRef="2" dataRef="d4"/>. </source>
  </segment>
</unit>
```

Constraints

- The following XLIFF Module attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `urn:oasis:names:tc:xliff:fs:2.0`, OPTIONAL, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:sizerestriction:2.0`, OPTIONAL, provided that the Constraints specified in the Size and Length Restriction Module are met.

- No other attributes MUST be used.

- The values of the attributes `canCopy`, `canDelete` and `canOverlap` MUST be the same as the values the ones in the `<sc>` element corresponding to this end code.

- The value of the attribute `canReorder` MUST be `no` if the value of `canReorder` is `firstNo` in the `<sc>` element corresponding to this end code.

- The attribute `isolated` MUST be set to `yes` if and only if the `<sc>` element corresponding to this end code is not in the same `<unit>` and set to `no` otherwise.

- If and only if the attribute `isolated` is set to `yes`, the attribute `id` MUST be used instead of the attribute `startRef` that MUST be used otherwise.

- If and only if the attribute `isolated` is set to `yes`, the attribute `dir` MAY be used, otherwise the attribute `dir` MUST NOT be used on the `<ec>` element.

Processing Requirements

- Extractors MUST NOT use the <sc> / <ec> pair to represent standalone codes.

  Rationale: Using a spanning code for a standalone code can easily result in having text inside a span where the original format does not allow it.

3.2.3.6 mrk

Represents an annotation pertaining to the marked span.

Contains:

- Text
- Zero, one or more <cp> elements
- Zero, one or more <ph> elements
- Zero, one or more <pc> elements
- Zero, one or more <sc> elements
- Zero, one or more <ec> elements
- Zero, one or more <mrk> elements
- Zero, one or more <sm> elements
- Zero, one or more elements

Text and inline elements may appear in any order.

Parents:

<source>, <target>, <pc> and <mrk>

Attributes:

- id, REQUIRED
- translate, OPTIONAL
- type, OPTIONAL
- ref, OPTIONAL
- value, OPTIONAL
- attributes from other namespaces, OPTIONAL

Constraints

- The [XML namespace] MUST NOT be used at this extension point.
- The following XLIFF Module attributes are explicitly allowed by the wildcard `other`:

  - attributes from the namespace `urn:oasis:names:tc:xliff:fs:2.0`, OPTIONAL, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:sizerestriction:2.0`, OPTIONAL, provided that the Constraints specified in the Size and Length Restriction Module are met.
  - attributes from the namespace `http://www.w3.org/2005/11/its`, OPTIONAL, provided that the Constraints specified in the ITS Module are met.
  - attributes from the namespace `urn:oasis:names:tc:xliff:itsm:2.1`, OPTIONAL, provided that the Constraints specified in the ITS Module are met.

See the Annotations section for more details and examples on how to use the <mrk> element.

3.2.3.7 sm

Start marker of an annotation where the spanning marker cannot be used for wellformedness reasons.

Contains:

This element is always empty.

Parents:

<source>, <target>, <pc> and <mrk>

Attributes:

- id, REQUIRED
- translate, OPTIONAL
- type, OPTIONAL
- ref, OPTIONAL
- value, OPTIONAL
- attributes from other namespaces, OPTIONAL

Constraints

- The [XML namespace] MUST NOT be used at this extension point.
- The following XLIFF Module attributes are explicitly allowed by the wildcard other:

  - attributes from the namespace urn:oasis:names:tc:xliff:fs:2.0, OPTIONAL, provided that the Constraints specified in the Format Style Module are met.
  - attributes from the namespace urn:oasis:names:tc:xliff:sizerestriction:2.0, OPTIONAL, provided that the Constraints specified in the Size and Length Restriction Module are met.
  - attributes from the namespace http://www.w3.org/2005/11/its, OPTIONAL, provided that the Constraints specified in the ITS Module are met.
  - attributes from the namespace urn:oasis:names:tc:xliff:itsm:2.1, OPTIONAL, provided that the Constraints specified in the ITS Module are met.

See the Annotations section for more details and examples on how to use the <sm> element.

3.2.3.8 em

End marker of an annotation where the spanning marker cannot be used for wellformedness reasons.

Contains:

This element is always empty.

Parents:

<source>, <target>, <pc> and <mrk>

Attributes:

- startRef, REQUIRED

See the Annotations section for more details and examples on how to use the element.

3.3 Attributes

This section lists all the various attributes used on DMLex core objects.

3.3.1 DMLex Attributes

The attributes defined in DMLex 1.0 are:

[olinks to attribute anchors in their files, left in some XLIFF attributes that should come handy] dir,hex, id, ref, srcDir, srcLang, trgLang, trgDir, and version.

3.3.1.1 dir

Directionality - indicates the directionality of content.

Value description: `ltr` (Left-To-Right), `rtl` (Right-To-Left), or `auto` (determined heuristically, based on the first strong directional character in scope, see [UAX #9]).

Default value: default values for this attribute depend on the element in which it is used:

- When used in a `<pc>`, `<sc>`, or `<ec>` element that has a `<source>` element as its parent:

  The value of the `srcDir` attribute of the `<unit>` element, in which the elements are located.

- When used in a `<pc>`, `<sc>`, or `<ec>` element that has a `<target>` element as its parent:

  The value of the `trgDir` attribute of the `<unit>` element, in which the elements are located.

- When used in a `<pc>`, `<sc>`, or `<ec>` element that has a `<pc>` element as its parent:

  The value of the `dir` attribute of the parent `<pc>` element.

- When used in `<data>`:

  The value `auto`.

Used in: `<data>`, `<pc>`, `<sc>`, and `<ec>`.

3.3.1.2 hex

Hexadecimal code point - holds the value of a Unicode code point that is invalid in XML.

Value description: A canonical representation of the hexBinary [XML Schema Datatypes] data type: Two hexadecimal digits to represent each octet of the Unicode code point. The allowed values are any of the values representing code points invalid in XML, between hexadecimal 0000 and 10FFFF (both included).

Default value: undefined

Used in: `<cp>`.

Example:

```
<cp hex="001A"/><cp hex="0003"/>
```

The example above shows a character U+001A and a character U+0003 as they have to be represented in XLIFF.

3.3.1.3 id

Identifier - a character string used to identify an element.

Value description: NCNAME. The scope of the values for this attribute depends on the element, in which it is used.

> **Note**
>
> All of the above defined uniqueness scopes ignore Module and Extension data. It would be impossible to impose those uniqueness requirements onto Module or Extension data. As Core only Modifiers could inadvertently cause conflicts with Modules or Extensions based data they cannot access. Modules and Extensions reusing Core need to specify their own uniqueness

scopes for the dmlex:id. In general, Modules and Extensions are advised to mimic the Core uniqueness requirement within their specific wrapper elements enclosing the reused Core elements or attributes, yet Module or Extensions are free to set wider uniqueness scopes if it makes business sense.

Default value: undefined

Used in:

[olinks to objects that use id] `entry`, `note`, `data`, `sc`, `ec`, `ph`, `pc`, `mrk` and `sm`.

3.3.1.4 ref

Reference - holds a reference for the associated annotation.

Value description: A value of the [XML Schema Datatypes] type anyURI. The semantics of the value depends on the type of annotation:

- When used in a term annotation, the URI value is referring to a resource providing information about the term.
- When used in a translation candidates annotation, the URI value is referring to an external resource providing information about the translation candidate.
- When used in a comment annotation, the value is referring to a `<note>` element within the same enclosing `<unit>`.
- When used in a custom annotation, the value is defined by each custom annotation.

Default value: undefined

Used in: `<mrk>` or `<sm>`.

Example:

```
<unit id="1">
  <segment>
    <source>The <pc id="1">ref</pc> attribute of a term
        annotation holds a <mrk id="m1" type="term"
        ref="http://dbpedia.org/page/Uniform_Resource_Identifier">
        URI</mrk> pointing to more information about the given
        term.</source>
  </segment>
</unit>
```

3.3.1.5 srcDir

Source directionality - indicates the directionality of the source content.

Value description: `ltr` (Left-To-Right), `rtl` (Right-To-Left), , or `auto` (determined heuristically, based on the first strong directional character in scope, see [UAX #9]).

Default value: default values for this attribute depend on the element in which it is used:

- When used in `<file>`:

  The value `auto`.

- When used in any other element:

The value of the `srcDir` attribute of its parent element.

Used in: `<file>`, `<group>`, and `<unit>`.

3.3.1.6 srcLang

Source language - the code of the language, in which the text to be Translated is expressed.

Value description: A language code as described in [BCP 47].

Default value: undefined

Used in: `<xliff>`.

3.3.1.7 trgLang

Target language - the code of the language, in which the Translated text is expressed.

Value description: A language code as described in [BCP 47].

Default value: undefined

Used in: `<xliff>`.

3.3.1.8 trgDir

Target directionality - indicates the directionality of the target content.

Value description: `ltr` (Left-To-Right), `rtl` (Right-To-Left), or `auto` (determined heuristically, based on the first strong directional character in scope, see [UAX #9]).

Default value: default values for this attribute depend on the element in which it is used:

- When used in `<file>`:

  The value `auto`.

- When used in any other element:

  The value of the `trgDir` attribute of its parent element.

Used in: `<file>`, `<group>`, and `<unit>`.

3.3.1.9 version

XLIFF Version - is used to specify the Version of the XLIFF Document. This corresponds to the Version number of the XLIFF specification that the XLIFF Document adheres to.

Value description: Text.

Default value: undefined

Used in: `<xliff>`.

3.3.2 XML namespace

The attributes from XML namespace used in DMLex are: xml:lang and xml:space.

3.3.2.1 xml:lang

Language - the xml:lang attribute specifies the language variant of the text of a given element. For example: `xml:lang="fr-FR"` indicates the French language as spoken in France.

Value description: A language code as described in [BCP 47].

Default value: default values for this attribute depend on the element in which it is used:

- Describe xml:lang inheritance rules for DMLex objects and relationship to srcLang, trgLang, as well as itmsLang
- When used in any other object:

  The value of the `xml:lang` attribute of its parent element.

Used in:

[olinks to objects using xml:lang] and where extension attributes are allowed.

Warning

Elaborate on dangers of the inhertance, interaction and inline usage

3.3.2.2 xml:space

White spaces - the xml:space attribute specifies how white spaces (ASCII spaces, tabs and line-breaks) are to be treated.

Value description: `default` or `preserve`. The value `default` signals that an application's default white-space processing modes are acceptable for this element; the value `preserve` indicates the intent that applications preserve all the white space. This declared intent is considered to apply to all elements within the content of the element where it is specified, unless overridden with another instance of the xml:space attribute. For more information see the section on xml:space [http://www.w3.org/TR/REC-xml/#sec-white-space] in the [XML] specification.

Default value: default values for this attribute depend on the object in which it is used:

- When used in `data`:

  The value `preserve`.

- When used in `dmlex`:

  The value `default`.

- When used in any other object:

  The value of the `xml:space` attribute of its parent object.

Used in: `dmlex>`, other olinks.

3.4 Inline Content

Provisions pertaining to inline codes and annotaions

- Text -- Textual content.
- Inline codes -- Sequences of content that are not linguistic text, such as formatting codes.
- Annotations -- Markers that delimit a span of the content and carry or point to information about the specified content.

  For example: an object indicating that a given expression in a text span references an entry.

There are ... objects that contain inline markup in DMLex: olinks to those objects.

In some cases, data directly associated with inline objects MAY also be stored at structural levels such as the `entry` object.

3.4.1 Text

The DMLex inline objects provisions do not prescribe how to represent normal text, specific provisions MAY be specified for different serializations.

3.4.1.1 Expressing characters invalid in certain serializations

DMLex needs to be able to represent all Unicode code points [Unicode].

the `cp` object MUST be used for those and only those Uniocode code points that are illegal in a certain serialization.

The syntax and semantics of `cp` in DMLex are similar to the ones of <cp> in the Unicode Locale Data Markup Language [LDML].

3.4.2 Inline Codes

Inline codes provisions ELABORATE..

Standalone
: A standalone code is a code that corresponds to a single position in the content. An example of such code is the
 element in HTML.

Spanning
: A spanning code is a code that encloses a section of the content using a start and an end marker. There are two kinds of spanning codes:

- Codes that can overlap, that is: they can enclose a non-closing or a non-opening spanning code. Such codes do not have an XML-like behavior. For example the RTF code `\b1...\b0` is a spanning code that is allowed to overlap.
- Codes that cannot overlap, that is: they cannot enclose a partial spanning code and have an XML-like behavior at the same time. An example of such code is the <emphasis>...</emphasis> element in DocBook.

Table 1. Inline code use cases

Use Case	Example of Representation
Standalone code	`ph`
Well-formed spanning code	`pc`
Start marker of spanning code	`sc`
End marker of spanning code	`ec`

3.4.3 Annotations

An annotation is an object that associates a section of the content with some metadata information.

Annotations MAY be created by any Writer.

Annotations are represented using either the `mrk` object, or the pair of `sm` and `em` objects.

3.4.3.1 Type of Annotations

There are ?several? pre-defined types of annotation and definition of custom types is also allowed.

3.4.3.1.1 Custom Annotation

The `mrk` or `sm` and `em` objects can be used to implement custom annotations.

A custom annotation MUST NOT provide the same functionality as a pre-defined annotation.

Usage:

- The `id` attribute is REQUIRED
- The `type` attribute is REQUIRED and set to a unique user-defined value.
- The use and semantics of the `value` and `ref` attributes are user-defined.

For example:

```
One of the earliest surviving works of literature is
<mrk id="m1" type="myCorp:isbn" value="978-0-14-44919-8">The
Epic of Gilgamesh</mrk>.
```

3.4.3.2 Overlapping Annotations

Annotations can overlap spanning inline codes or other annotations. Because of this, a single annotation span can be represented using a pair of `sm` and `em` elements instead of a single `mrk` element.

3.4.4 White Spaces

White space provisions for different serializations and for exchange among serializations.

> **Note**
>
> Note that in some serializations, all whitespace is significant. Therefore it is advisable to maximize usage of `preserve` in markup environments in order to maximize interoperability with other serializations.

Processing Requirements

- For the inline content and all non empty inline objects: The white spaces MUST be preserved if the value for `xml:space` set or inherited at the enclosing `entry` level is `preserve`, and they MAY be preserved if the value is `default`.

3.4.5 Bidirectional Text

Text directionality in DMLex content is defined by inheritance. Source and target content (for instance in bilingual translation dictionaries) can have different directionality.

The initial directionality for both the source and the target content is defined in the top level logical root object element, using the OPTIONAL attributes `srcDir` for the source and `trgDir` for the target. The default value for both attributes is `auto`.

Monolingual and multilingual lexicographic content SHOULD NOT use the attributes `srcDir` and `trgDir`. The `dir` attribute with values `ltr`, `rtl`, or `auto` has standard inheritance throughout the DMlex object model, in case the `srcDir` and `trgDir` are not used on the top level.

In addition, the `data` object has an OPTIONAL attribute `dir` with a value `ltr`, `rtl`, or `auto` that is not inherited. The default value is `auto`.

Directionality text contained in DMLex text nodes is fully governed by [UAX #9], whereas explicit DMlex-defined structural and directionality objects and attributes constitute a higher-level protocol in the sense of [UAX #9]. The DMLex-defined value `auto` determines the directionality based on

the first strong directional character in its scope and DMLex-defined inline directionality behaves exactly as Explicit Directional Isolate Characters, see [UAX #9], http://www.unicode.org/reports/tr9/#Directional_Formatting_Characters.

Note

Please note that this specification does not define explicit markup for inline directional Overrides or Embeddings; in case those are needed. ELABORATE .. [UAX #9] defined Directional Formatting Characters [??].

3.4.6 Content Comparison

This specification defines content equality as follows:

- Two contents are equal if their normalized forms are equal.

A content is normalized when:

- The text nodes are in Unicode Normalized Form C defined in the Unicode Annex #15: Unicode Normalization Forms [UAX #15].
- All annotation markers are removed.
- All pairs of `sc` and `ec` elements that can be converted into a `pc` element, are converted.
- All adjacent text nodes are merged into a single text node.
- For all the text nodes with the white space property set to `default`, all adjacent white spaces are collapsed into a single space.

3.5 Extension Mechanisms

DMLex offers the following mechanisms for storing custom data in DMLex Instances:

1. ??A set of metaobjects to store arbitrary metadata??
2. Using a namespace based mechanism for storing data in custom objects or attributes at predefined extension points.

Both mechanisms can be used simultaneously. [if both defined]

3.5.1 Extension Points

The following DMLex Core objects allow storing custom data in DMLex predefined metaobjects:

- `entry`
- another allowed object olink
- yet another

The following DMLex Core objects accept custom attributes:

- `dmlex>`
- `entry`
- another allowed object olink
- yet another

3.5.1.1 Extensibility of DMLex Modules

For extensibility of DMLex Modules please refer to the relevant Module Sections.

3.5.2 Constraints

- When using identifiers, an extension MUST use either an attribute named `id` or the attribute `xml:id` to specify them.
- Extensions identifiers MUST be unique within ELABORATE.. .
- Identifier values used in extensions MUST be of type `xs:NCNAME`, `xs:NMTOKEN`, or compatible with `xs:NMTOKEN` (e.g. `xs:NAME` and `xs:ID` are compatible).

These constraints are needed for the fragment identification mechanism.

3.5.3 Processing Requirements

- A user extension, whether implemented using predefined metaobjects or using a custom namespace, MUST NOT provide the same functionality as an existing DMLex Core or Module feature, however it MAY complement an extensible DMLex Core or Module feature or provide a new functionality at the provided extension points.
- Another requirement.
- Writers that do not support a given custom namespace based user extension SHOULD preserve that extension without Modification.

4 The DMLex Modules Specifications

This section specifies the OPTIONAL DMLex Modules that MAY be used along with DMLex Core for advanced functionality.

ONE MODULE PLACEHOLDER INCLUDED AS MODULE CONTENT MODEL

4.1 Module 01 content model

4.1.1 Introduction

This module defines ... ELABORATE

Note

This module specification chiefly describes how... ELABORATE

Warning

There is an important difference or similar ELABORATE

However, … ELABORATE.

Implementers who wish to better access... ELABORATE.

4.1.2 Module namespace and validation artifacts

The namespaces for Module 01 are: `https://www.w3.org/2005/11/its/` and `urn:oasis:names:tc:lexidma:module_01:1.0`.

Validation artifacts [specify type of artifacts if any available at all] for this module 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

Although setting and usage of prefixes for namespaces in XML is arbitrary, we are using `md01:` for the `http://docs.oasis-open.org/lexidma/ns/module_01-1.0` namespace and `md01n:` for the `urn:oasis:names:tc:lexidma:module_01:1.0` namespace throughout this specification.

While colon ":" is the mandatory separator character in XML, other separator characters, such as hyphen "-" or undescore "_" MAY be used in other serializations.

4.1.3 Module fragment identification prefix

The fragment identification prefix for the Module 01 module is: `md01`.

Note

Although this module has to use several different XML namespace prefixes it uses only one fragment identification and authority prefix which is `md01`.

4.1.4 Conformance to the Module 01 specification

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 Module 01 Processors MUST be able to use the external global rules included in the module's [validation artifact - adjust according to the type of validation artifact] file http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/its.sch and compute Module 01 data categories encoded in DMlex Instances as per certain conformance clauses ELABORATE
    - Subrequirement, ELABORATE.
- Conformant Agents MUST be DMLex Conformant in the sense of DMLex Application Conformance and also implement at least one Module 01 data category defined in the section Module 01 data categories defined in the Module 01 Module of this Module 01 or provide full support for at least one of the custom annotations etc. ELABORATE.

  In particular:
    - Conformant Creators MUST be capable of Creating at least one of the above specified Module 01 data categories in a resulting conformant DMlex Instance with Module 01 based metadata.
    - 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.

4.1.5 Module 01 functionality 01

Module 01 functionality 01 mechanism provides a way to record metadata..etc. ELABORATE>

Warning

This mechanism is reserved for … ELABORATE.

Note

The Module 01 functionality 01 mechanism has to be always used with the ELABORATE. .

Another informative pointer.

Processing Requirements

- Writers MUST use the attribute … ELABORATE to express the information provided through the Module 01 functionality 01 mechanism in DMLex Instances.

4.1.5.1 Module 01 functionality 01 annotation

This is used to express the Module 01 functionality 01 mechanism on inline markers.

Usage:

- The `id` attribute is REQUIRED.
- Other attributes related provision.
- Etc.

4.1.6 Module 01 data categories defined in the Module 01 Module

The following Module 01 data categories are fully specified within this module:

- data_category_01

4.1.7 Module objects

All Module 01 objects and attributes belong to the http://docs.oasis-open.org/lexidma/ns/module_01-2.1 namespace. The Module 01 defines the following objects:

`md01:object_01>`.

4.1.7.1 Diagram

Legend:

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

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

4.1.7.2 md01:object_01_01

object_01_01 - a standoff element to hold information about ...ELABORATE.

Contains:

This element is always empty.

Parents:

- `md01:object_wrapper_01`

Attributes:

- `attribute_01`, REQUIRED
- olinks to other attributes as needed, OPTIONAL | REQUIRED

Constraints

- At least one of the attributes ELABORATE.. MUST be set.

Processing Requirements

- For all Agents, when any of the attributes ELABORATE.. are declared, ELABORATE REQUIRED.
- Modifiers MUST ELABORATE.. .

4.1.7.3 md01:object wrapper 01

Object wrapper 01 - a standoff wrapper element to group any number of single object 01 objects related to the same node or span of lexicographic content.

Contains:

- One or more `md01:object_01_01` objects

Parents:

- `entry`

Attributes:

- `xml:id` , REQUIRED

Constraints

- Each `md01:object wrapper 01` object SHOULD be referenced ELABORATE.. within the same `entry` object as per Constraints for ELABORATE.. .

Processing Requirements

- Modifiers detecting an orphaned `md01:object wrapper 01` object MAY delete that `md01:object wrapper 01` object.

4.1.7.4 md01:object_02_01

Object 02 01 - a standoff element to hold information of ELABORATE.. .

Contains:

This element is always empty.

Parents:

- `md01_object_wrapper_02`

Attributes:

- each member keeps an olink to an admissible attribute , OPTIONAL | REQUIRED

Constraints

- FOR INSTANCE At least one of the following MUST be set:
  - each list item keeps a para wrapped olink to an admissible attribute,

Processing Requirements

- For all Agents, when any of the attributes ELABORATE are declared on the `md01_object_02_01` object, these apply to the respective structural objects' content or the marker delimited inline spans of Module 01 functionality 01 annotation, from which their enclosing `md01_object_wrapper_02` object is referenced.

4.1.7.5 md 01 object wrapper 02

Md 01 object wrapper 02 - a standoff wrapper element to group any number of single md 01 object 02 01 objects related to the same span of lexicographic content.

Contains:

- One or more `md01_object_02_01` objects

Parents:

- `entry`
- olink to another admissible parent

Attributes:

- `xml:id` , REQUIRED

Constraints

- Each md 01 object wrapper 02 object SHOULD be referenced ELABORATE.. .

Processing Requirements

- Modifiers detecting an orphaned md 01 object wrapper 02 object MAY delete that md 01 object wrapper 02 element.

4.1.8 Module Attributes

The Module 01 uses the following attributes from the http://docs.oasis-open.org/lexidma/ns/module_01-2.1 namespace:

[olinks to module atribute anchors in their files]

The attributes defined in the Module 01 that belong to the `urn:oasis:names:tc:xliff:itsm:2.1` namespace are: `domains` and `lang`.

The Module 01 also uses the `xml:id` attribute.

4.1.8.1 itsm:domains

Domains - the `itsm:domains` attribute expresses the [ITS] Domain [http://www.w3.org/TR/its20/#domain] data category.

Value description: The value is a text string, however commas if present separate distinct domain values within the string.

Default value:: default values for this attribute depend on the element in which it is used:

- When used in `<file>`:

  The value is undefined.

- When used in any other admissible structural element (`<group>` or `<unit>`):

  The value of the `domains` attribute of its parent element (which can be undefined).

- When used in annotations markers `<mrk>` or `<sm>`:

  The value of the `domains` attribute of the innermost `<mrk>`, `<unit>`, or `<mtc:match>` element, element, in which the marker in question is located (which can be undefined).

- When used in the `<mtc:match>` element:

  The value is undefined.

Used in: `<file>` `<group>` `<unit>`, `<mrk>`, `<sm>`, and `<mtc:match>`.

See the ITS Domain Annotation for the normative usage description of this attribute inline.

Warning

This attribute belongs to the `urn:oasis:names:tc:xliff:itsm:2.1` namespace that is being prefixed with `itsm:` throughout this specification, unlike the original W3C ITS namespace `https://www.w3.org/2005/11/its/` that is being prefixed with `its:`.

4.1.8.2 itsm:lang

Inline language information - the `itsm:lang` attribute specifies an inline foreign language span within the source or target content of the otherwise bilingual XLIFF Document. For example: `itsm:lang="fr-FR"` indicates the French language as spoken in France.

> **Note**
>
> This is NEVER used on structural elements that have their Language Information set by the XLIFF Core `xlf:srcLang` and `xlf:trgLang` attributes. It is not advisable to use this attribute on structural elements even outside of XLIFF where the Language Information is typically given by the `xml:lang` attribute.

Value description: A language code as described in [BCP 47].

Default value:

The value of the `xml:lang` or `itsm:lang` attribute set or inherited on the parent element of the `<mrk>` or `<sm>` element in question.

Used in: `<mrk>` and `<sm>`.

See the ITS Language Information Annotation for the normative usage description of this attribute.

> **Note**
>
> `itsm:lang` is an attribute analogical to `xml:lang`. Unlike `xml:lang`, it is allowed on XLIFF inline Annotations. The normative behavior of this attribute results from the XLIFF Core behavior as further specified by the ITS Language Information Annotation.

> **Warning**
>
> This attribute belongs to the `urn:oasis:names:tc:xliff:itsm:2.1` namespace that is being prefixed with `itsm:` throughout this specification, unlike the original W3C ITS namespace `https://www.w3.org/2005/11/its/` that is being prefixed with `its:`.

4.1.8.3 xml:id

Identifier - the `id` attribute from the `http://www.w3.org/XML/1998/` namespace is used to identify module standoff wrapper objects.

Value description: xs:ID

Default value: undefined

Used in:`<locQualityIssues>` and `<provenanceRecords>`.

> **Warning**
>
> Implementers need to be aware that `xs:ID` has to be globally unique.

4.1.9 Example file

Example 1. Module 01 Data Categories Example

The following example file includes markup related to several Module 01 data categories.

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

programlisting code
```

5 DMLex XML serialization

5.1 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.

5.2 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]

.

5.3 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.

5.4 Other XML serialization provisions

Other provisions. ELABORATE

Warning

ELABORATE.

Note

ELABORATE

Another informative pointer.

Processing Requirements

- Writers MUST ELABORATE in DMLex XML Instances.

5.5 XML serialization elements

DMLex XML serialization uses the following elements:

[comma separated list of element olinks]

5.5.1 Diagram

Legend:

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

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

5.6 XML serialization attributes

The XML serialization uses the following attributes:

[comma separated list of attribute olinks]

5.7 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
```

6 DMLex JSON serialization

6.1 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.

6.2 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]

.

6.3 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.

6.4 Other JSON serialization provisions

Other provisions. ELABORATE

Warning

ELABORATE.

Note

ELABORATE

Another informative pointer.

Processing Requirements

- Writers MUST ELABORATE in DMLex JSON Instances.

6.5 JSON serialization objects

DMLex JSON serialization uses the following objects:

[comma separated list of object olinks]

6.5.1 Diagram

Legend:

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

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

6.6 JSON serialization properties

The JSON serialization uses the following properties:

[comma separated list of properties olinks]

6.7 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
```

7 DMLex RDF serialization

7.1 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.

7.2 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]

.

7.3 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.

7.4 Other RDF serialization provisions

Other provisions. ELABORATE

Warning

ELABORATE.

Note

ELABORATE

Another informative pointer.

Processing Requirements

- Writers MUST ELABORATE in DMLex RDF Instances.

7.5 RDF serialization objects

DMLex RDF serialization uses the following objects:

[comma separated list of object olinks]

7.5.1 Diagram

Legend:

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

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

7.6 RDF serialization attributes

The RDF serialization uses the following attributes:

[comma separated list of object olinks]

7.7 Example file

Example 4. 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

B.1 Tree diagram of validation artifacts

```
    [Diagram currently contains tentative example structure with some potential referen

Master NVDL Schema [http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/advanced
   |      |
   |    Core XML Schema [http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/dml
   |      |
   |      +---Module 01 XML Schema [http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/
   |             |
   |             +---Internationalization Tag Set (ITS) Version 1.0 [https://www.w3.org
   |             |
   |             +---XLIFF ITS Module XML Schema [http://docs.oasis-open.org/lexidma/dm
   |
   +---Core constraints [http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/schemas/dml
         |
         +---Module 01 Constraints [http://docs.oasis-open.org/lexidma/dmlex/v1.0/wd01/
```

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