INTERNATIONAL STANDARD

ISO 24622-2

First edition
2019-07

Language resource management — Component metadata infrasctructure (CMDI) —

Part 2: Component metadata specification language

Gestion des ressources linguistiques — Composante infrastructure de métadonnées (CMDI) —

Partie 2: Composante linguistique spécifique aux métadonnées

Reference number
ISO 24622-2:2019(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 37, *Language and terminology*, Subcommittee SC 4, *Language resource management*.

A list of all parts in the ISO 24622 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

Many researchers, from the humanities and other domains, have a strong need to study resources in close detail. Nowadays more and more of these resources are available online. To be able to find these resources, they are described with metadata. These component metadata (CMD) instances are collected and made available via central catalogues. Often, resource providers want to include specific properties of a resource in their metadata to provide all relevant descriptions for a specific type of resource. The purpose of catalogues tends to be more generic and addresses a broader target audience. It is hard to strike the balance between these two ends of the spectrum with one metadata schema, and mismatches can negatively impact the quality of metadata provided. The goal of the component metadata infrastructure (CMDI) is to provide a flexible mechanism to build resource specific metadata schemas out of shared components and semantics[14][15].

In CMDI the metadata lifecycle starts with the need of a metadata modeller to create a dedicated metadata profile for a specific type of resource. Modellers can browse and search a registry for components and profiles that are suitable or come close to meeting their requirements. A component groups together metadata elements that belong together and can potentially be reused in a different context. Components can also group other components. Existing component registries, e.g., the CLARIN (common language resources and technology infrastructure) Component Registry[16], might already contain any number of components. These can be reused as they are, or be adapted by modifying, adding or removing some metadata elements and/or components. Also completely new components can be created to model the unique aspects of the resources under consideration. All the needed components are combined into one profile specific for the type of resources. Any component, element and value in such a profile may be linked to a semantic description — a *concept* — to make their meaning explicit[21]. These semantic descriptions can be stored in a semantic registry, e.g., the CLARIN Concept Registry[17]. In the end metadata creators can create records for specific resources that comply with the profile relevant for the resource type, and these records can be provided to local and global catalogues[22].

CMDI has originally been developed in the context of the European CLARIN infrastructure initiative with input from other initiatives and experts. Already in its preparatory phase, which started in 2007, the infrastructure needed flexibility in the metadata domain as it was confronted with many types of resources that had to be accurately described. For Version 1.0 a toolkit[20] was created, consisting of the XML schemas and XSLT stylesheets to validate and transform components, profiles and records. Version 1.1 included some small changes and has seen small incremental backward compatible advances since 2011. This version has been in use, new developments and the development of this document resulted in Version 1.2[18]. Also CMDI has seen a growing number of tools and infrastructure systems that deal with its records and components and rely on its shared syntax and semantics.

In ISO 24622-1, the component metadata model has been standardized. This document is compliant with ISO 24622-1, and also extends and constrains it at various places (see also the red parts in the UML class diagram in Figure 1):

— support for attributes on both components and elements is added,

— a profile is limited to one root component, and

— an element always belongs to a specific component.

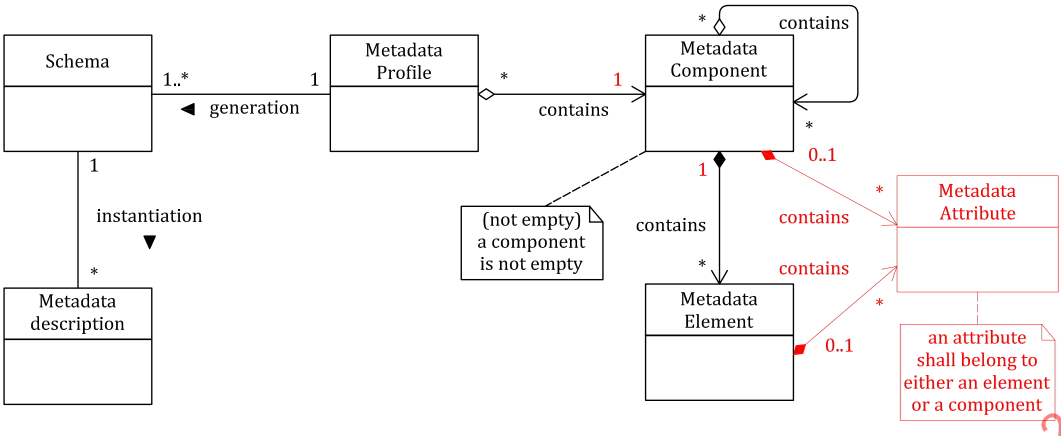

Figure 1 — Component metadata model and its extensions

Language resource management — Component metadata infrasctructure (CMDI) —

Part 2: Component metadata specification language

IMPORTANT — The electronic file of this document contains colours which are considered to be useful for the correct understanding of the document. Users should therefore consider printing this document using a colour printer.

1 Scope

The component metadata lifecycle needs a comprehensive infrastructure with systems that cooperate well together. To enable this level of cooperation this document provides in depth descriptions and definitions of what CMDI records, components and their representations in XML look like.

This document describes these XML representations, which enable the flexible construction of interoperable metadata schemas suitable for, but not limited to, describing language resources. The metadata schemas based on these representations can be used to describe resources at different levels of granularity (e.g. descriptions on the collection level or on the level of individual resources).

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at http://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1 General terms

3.1.1
concept
unit of knowledge created by a unique combination of characteristics

[SOURCE: ISO 1087:—[1]), 3.2.3, modified — Note 1 to entry and Note 2 to entry have been deleted.]

3.1.2
concept link
reference from a *CMD profile* (3.2.11), *CMD component* (3.2.3), *CMD element* (3.2.5), *CMD attribute* (3.2.2) or a value in a *controlled_vocabulary* (3.1.4) to an entry in a *semantic registry* (3.1.11) via a *Uniform Resource Identifier* (3.1.13)

Note 1 to entry: Typically a concept link is provided as a *persistent identifier* (3.1.9).

1) Revision of ISO 1087:2000 under preparation. Stage at the time of publication: ISO/FDIS 1087:2019.

3.1.3
concept registry
semantic registry (3.1.11) maintaining *concepts* (3.1.1)

EXAMPLE The CLARIN Concept Registry[17] as used in the CLARIN infrastructure.

3.1.4
controlled vocabulary
closed/open vocabulary
set of values that can be used either to constrain the set of permissible values or to provide suggestions for applicable values in a given context

3.1.5
data category
class of data items that are closely related from a formal or semantic point of view

EXAMPLE /part of speech/, /subject field/, /definition/.

Note 1 to entry: A data category can be viewed as a generalization of the notion of a field in a database.

[SOURCE: ISO 30042:2019, 3.8, modified — Note 2 to entry has been deleted.]

3.1.6
language tag
textual code used to assist in identifying languages in every mode of communication

Note 1 to entry: This includes constructed and artificial languages but excludes languages not intended primarily for human communication, for example in spoken, written, signed, or otherwise signaled, communication (see IETF BCP 47[6]).

Note 2 to entry: Language tags may be used to assist in the identification of a language in every mode of communication, for example in spoken, written, signed, or otherwise signaled, communication.

3.1.7
media type
MIME type
media type specification used originally for textual, non-textual, multi-part message bodies of emails and which provides technical format information on data

Note 1 to entry: For the purposes of this document, it is as described in IETF RFC 6838[8].

3.1.8
metadata
resource (3.1.10) that is a description of another resource, usually given as a set of properties in the form of attribute-value pairs

Note 1 to entry: This description can contain information about the resource, aspects or parts of the resource and/or artefacts and actors connected to the resource.

3.1.9
persistent identifier
PID
unique identifier that ensures permanent access for a digital object by providing access to it independently of its physical location or current ownership

Note 1 to entry: Unique in this context means that the PID will not be issued again for other *resources* (3.1.10). However, the same PID can reference different representations or incarnations of the resource at the discretion of the resource provider.

[SOURCE: ISO 24619:2011, 3.2.4]

3.1.10
resource
entity, possibly digitally accessible, that can be described in terms of its content and technical properties, referenced by a *Uniform Resource Identifier* (3.1.13)

3.1.11
semantic registry
directory of (authoritative) definitions of *term* (3.1.12), *concept* (3.1.1) or *data category* (3.1.5), or the system maintaining it

Note 1 to entry: These registries generally also provide *persistent identifiers* (3.1.9) for their entries.

3.1.12
term
designation that represents a general *concept* (3.1.1) in a specific domain or subject

EXAMPLE "planet", "tower", "pen", "numeral", "number", "square root", "logarithm", "unit of measurement", "base of a logarithm", "chemical element", "chemical compound", "HP Laserjet 1100", "Nobel Prize in Physics".

Note 1 to entry: Terms may be partly or wholly verbal.

Note 2 to entry: Terms can include letters and letter symbols, numerals, mathematical symbols, typographical signs and syntactic signs (e.g. punctuation marks, such as hyphens, parentheses, square brackets and other connectors or delimiters), sometimes in character styles (i.e. fonts and bold, italic, bold italic, or other style conventions) governed by domain-, subject-, or language-specific conventions.

[SOURCE: ISO 1087:—, 3.4.2]

3.1.13
Uniform Resource Identifier
URI
sequence of characters that identifies a *resource* (3.1.10)

Note 1 to entry: IETC RFC 3986[7] defines the generic URI syntax and a process for resolving URI references that might be in relative form, along with guidelines and security considerations for the use of URIs on the Internet.

3.2 CMDI

3.2.1
CCSL
CMDI component specification language
XML (3.3.4) based language for describing a *CMD component* (3.2.3) and a *CMD profile* (3.2.11) according to the *CMD model* (3.2.10)

3.2.2
CMD attribute
unit within a *CMD element* (3.2.5) that describes the level at which properties of a CMD element can be provided by means of *value scheme* (3.2.20) constrained atomic values

3.2.3
CMD component
component
reusable, structured template for the description of (an aspect of) a *resource* (3.1.10), defined by means of a *CMD specification* (3.2.14) document with the potential of including other CMD components, either through reference or inline definition

3.2.4
CMD component registry
component registry
service where a *CMD specification* (3.2.14) can be registered and accessed

3.2.5
CMD element
element definition
unit within a *CMD component* (3.2.3) that describes the level of the *CMD instance* (3.2.6) that can carry atomic values governed by a *value scheme* (3.2.20), and does not contain further levels except for that of the *CMD attribute* (3.2.2)

3.2.6
CMD instance
metadata instance
CMDI file
CMDI instance
metadata record
CMD record
file that conforms to the general CMD instance structure as described in this document and, at the *CMD instance payload* (3.2.9) level, follows the specific structure defined by the *CMD profile* (3.2.11) it relates to

3.2.7
CMD instance envelope
section of a *CMD instance* (3.2.6) which is structured uniformly for all instances and contains the *CMD instance header* (3.2.8) and the list of *resource proxies* (3.2.18) which may be referenced from the *CMD instance payload* (3.2.9) section

3.2.8
CMD instance header
section of a *CMD instance* (3.2.6) marked as 'header', providing information on that *CMD instance* as such, not the *resource* (3.1.10) that is described by the metadata file

3.2.9
CMD instance payload
section of a *CMD instance* (3.2.6) that follows the structure defined by the *CMD profile* (3.2.11) it references and contains the description of the *resource* (3.1.10) to which that CMD instance relates

3.2.10
CMD model
component metadata model
metadata model that is based on *CMD components* (3.2.3)

Note 1 to entry: For the purposes of this document, it is as specified in ISO 24622-1.

3.2.11
CMD profile
profile
structured template for the description of a class of *resources* (3.1.10) providing the complete structure for a *CMD instance payload* (3.2.9) by means of a hierarchy of *CMD components* (3.2.3)

3.2.12
CMD profile schema
schema definition by which the correctness of a *CMD instance* (3.2.6) with respect to the *CMD profile* (3.2.11) it pertains to can be evaluated

Note 1 to entry: The CMD profile schema may be expressed as *XML Schema* (3.3.11) but also in other XML schema languages.

3.2.13
CMD root component
CMD component (3.2.3) that is defined at the highest level within a *CMD profile* (3.2.11) that may have one or more child *CMD components* (3.2.3) but no siblings

Note 1 to entry: In the *CMD instance payload* (3.2.9), it is instantiated exactly once.

3.2.14
CMD specification
component specification
component definition
profile specification
profile definition
representation of a *CMD component* (3.2.3) or *CMD profile* (3.2.11), expressed using the constructs of the *CCSL* (3.2.1)

3.2.15
CMD specification header
component header
profile header
section of a *CMD specification* (3.2.14) marked as 'header', providing information on that CMD specification as such that is not part of the defined structure

3.2.16
CMDI
component metadata infrastructure
metadata description framework consisting of the *CMD model* (3.2.10) and infrastructure to process instances of parts of the model

3.2.17
inline CMD component
CMD component (3.2.3) that is created and stored within another CMD component and cannot be addressed from other CMD components

3.2.18
resource proxy
CMD resource reference
representation of a *resource* (3.1.10) within a *CMD instance* (3.2.6) containing a *Uniform Resource Identifier* (3.1.13) as a reference to the resource itself and an indication of its nature

3.2.19
resource proxy reference
reference from any point within the *CMD instance payload* (3.2.9) to any of the *resource proxy* (3.2.18) elements

3.2.20
value scheme
set of constraints governing the range of values allowed for a specific *CMD element* (3.2.5) or *CMD attribute* (3.2.2) in a *CMD instance* (3.2.6), expressed in terms of an *XML Schema datatype* (3.3.12), *controlled vocabulary* (3.1.4), or *regular expression* (3.3.3)

3.3 XML

3.3.1
foreign attribute
XML attribute (3.3.5) defined in a *namespace* (3.3.2) other than those declared in *CMDI* (3.2.16), to be included in a *CMD instance* (3.2.6) as additional information targeted to specific receivers or applications

3.3.2
XML namespace
namespace
method for qualifying element and attribute names used in XML

Note 1 to entry: For the purposes of this document, it is as described in W3C XML Namespaces[10].

3.3.3
regular expression
sequence of characters that denote a set of strings

Note 1 to entry: When used to constrain a lexical space, a regular expression asserts that only strings in the defined set of strings are valid literals for values of that type.

Note 2 to entry: See also W3C XSchema Part 2[12], Appendix F.

3.3.4
XML
markup language for describing hierarchical structures within a text file

Note 1 to entry: For the purposes of this document, it is as defined by W3C recommendation for the Extensible Markup Language XML[9].

3.3.5
XML attribute
property of an *XML element* (3.3.9)

Note 1 to entry: For the purposes of this document, it is as defined by W3C recommendation for the extensible Markup Language XML[9].

3.3.6
XML attribute declaration
constituent of an *XML Schema* (3.3.11) that constrains the structure and content of a specific *XML attribute* (3.3.5)

Note 1 to entry: For the purposes of this document, it is as defined by W3C recommendation on XSD[13], Section 3.2.

3.3.7
XML container element
XML element (3.3.9) that has one or more XML elements as its descendants

3.3.8
XML document
document represented in XML

Note 1 to entry: For the purposes of this document, it is as defined by W3C recommendation for the extensible Markup Language XML[9].

3.3.9
XML element
constituent of an *XML document* (3.3.8)

Note 1 to entry: For the purposes of this document, it is as defined by W3C recommendation for the extensible Markup Language XML[9].

3.3.10
XML element declaration
constituent of an *XML Schema* (3.3.11) that constrains the structure and content of a specific *XML element* (3.3.9)

Note 1 to entry: For the purposes of this document, it is as defined by W3C recommendation on XSD[13], Section 3.3.

3.3.11
XML Schema
document that complies with the XML Schema recommendation

Note 1 to entry: For the purposes of this document, it refers to the W3C XSchema Part 1 recommendation[11].

3.3.12
XML Schema datatype
predefined set of permissible content within an *XML element* (3.3.9) or an *XML attribute* (3.3.5) of an *XML document* (3.3.8) used in an *XML Schema* (3.3.11)

Note 1 to entry: For the purposes of this document, it is as described in W3C XSchema Part 2[12].

4 Notational and XML namespace conventions

The following notational conventions for XML fragments are used throughout this document:

— <Element>
an XML element with the generic identifier Element that is bound to a default XML namespace;

— <prefix:Element>
an XML element with the generic identifier Element that is bound to an XML namespace denoted by the prefix prefix;

— <prefix:{Element}>
an XML element with a contextually specified identifier that is bound to an XML namespace denoted by the prefix prefix;

— <prefix:{Element}>*
any number of XML elements with contextually specified identifiers that are bound to an XML namespace denoted by the prefix prefix;

— @attr
an XML attribute with the name attr;

— @{attr}
an XML attribute with a contextually specified name;

— @{attr}*
any number of XML attributes with contextually specified names;

— @prefix:attr
an XML attribute with the name attr that is bound to an XML namespace denoted by the prefix prefix;

— string
the literal string shall be used either as element content or attribute value;

— xs:type
the XML schema type with name type.

The XML namespace names and prefixes given in Table 1 are used throughout this document as existing suitable examples. The column "Recommended Syntax" indicates which syntax variant should be used by the toolkit and other creators of CMDI related documents.

Table 1 — XML namespaces and prefixes used in this document as existing suitable examples

Prefix	Namespace name	Comment	Recommended syntax
cmd	http://www.clarin.eu/cmd/1	CMD instance (general/envelope)	prefixed
cmdp	http://www.clarin.eu/cmd/1/profiles/profileId	CMDI payload (CMD profile specific)	prefixed

cue	http://www.clarin.eu/cmd/cues/1	Cues for tools	prefixed
xs	http://www.w3.org/2001/XMLSchema	XML Schema	prefixed

NOTE The inclusion of the major version number (i.e. 1) in the clarin.eu namespaces, but not the minor version number reflects the approach that across minor versions within a major version of the CMDI specification, the namespace is kept constant for compatibility reasons.

5 Structure of CMDI instances

5.1 General structure

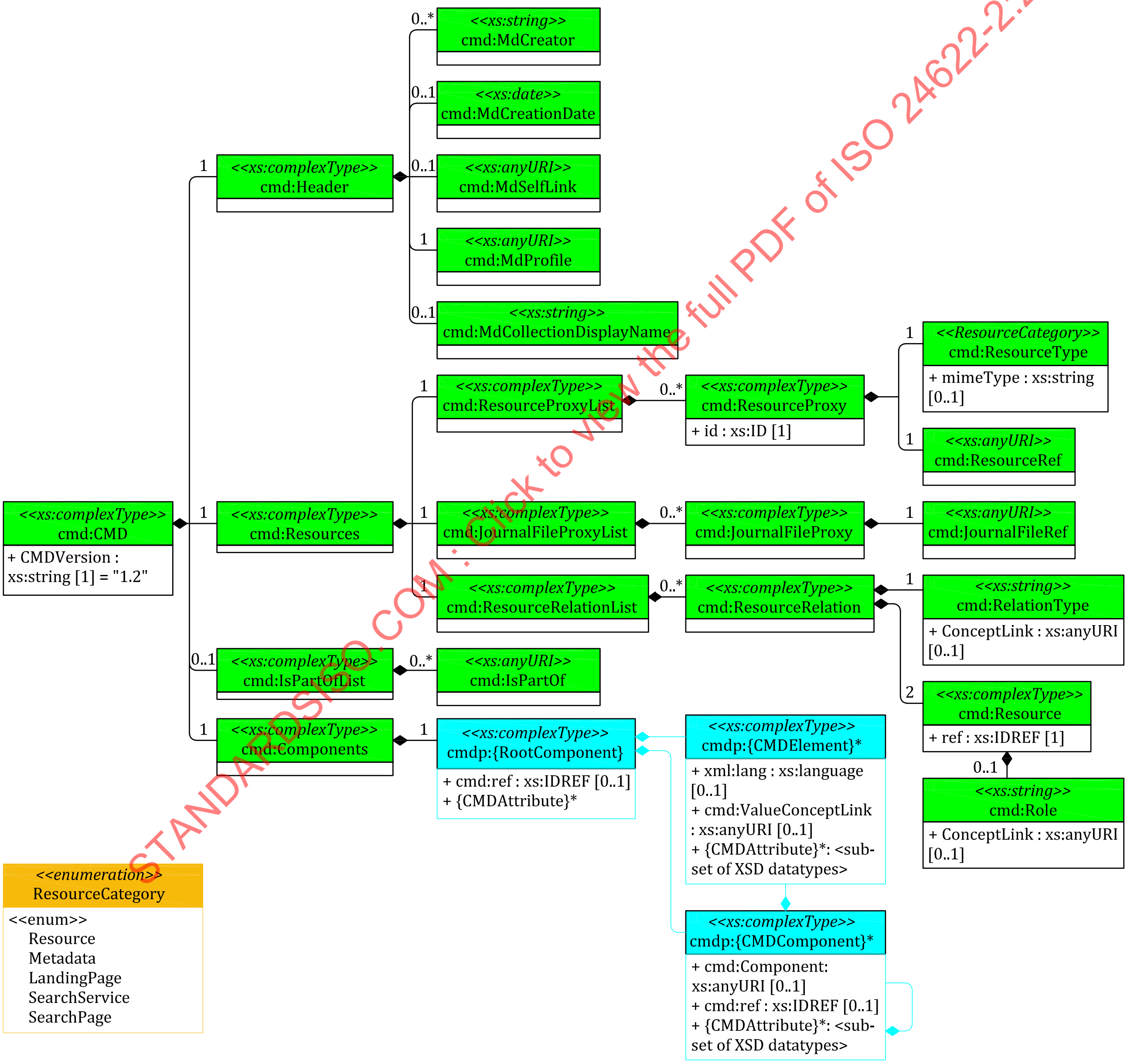

Figure 2 — The structure of a CMD instance

See Figure 2, which uses the following colour scheme: Green boxes represent elements that are potentially present in all CMD instances (the CMD instance envelope). Blue boxes and associations represent elements defined by the CMD profile (the CMD instance payload). The diagram is meant for overview and illustration; full details are found in Tables 2 to 15.

A CMD instance contains the actual metadata of one specific resource (hereafter referred to as the *described resource*) and might also be referred to as a *CMD record* or *CMD instance*. All CMD instances have the same structure at the top level (the *CMD instance envelope*). At a lower level, parts of its structure are defined by the CMD profile upon which it is based (the *CMD instance payload*).

5.2 The main structure

A CMD instance has the (XML) root element <cmd:CMD> with one attribute and 4 sub-elements that appear in the mandatory order described in Table 2.

Table 2 — Root element: order of child elements

Name	Value type	Occurrences	Description
<cmd:CMD>	xs:complex type		The (XML) root element of the CMD instance.
@CMDVersion	xs:string("1.2")	1	Denotes the CMDI version on which this CMDI file is based.
<cmd:Header>	xs:complexType	1	Encapsulates core administrative data about the CMDI file.
<cmd:Resources>	xs:complexType	1	Includes 3 lists containing information about resource proxies and their interrelations.
<cmd:IsPartOfList>	xs:complexType	0 or 1	A list of <cmd:IsPartOf> elements, each referencing a larger external resource of which the described resource (as a whole) forms a part.
<cmd:Components>	xs:complexType	1	This element contains the CMD profile specific section of the CMD instance. Here the descriptive metadata of the resource are found.

The first three elements (<cmd:Header>, <cmd:Resources> and <cmd:IsPartOfList>) constitute the CMD instance envelope and reside in the cmd namespace. The CMD instance payload is contained in the <cmd:Components> element, the elements of the instance payload (which is CMD profile specific) exists in the CMD profile specific namespace (prefix cmdp), possibly adorned with attributes in the cmd namespace.

In addition to this, foreign attributes (XML attributes of other namespaces than those defined in Clause 4) may occur anywhere in <cmd:Header>, <cmd:Resources> and <cmd:IsPartOfList> elements and on the <cmd:Components> element (but not on any of its children). These foreign namespaces should be ignored by tools unrelated to the party associated with the namespace and therefore may be removed during processing. The foreign namespace shall be representative of the party that introduces the extension. For example, the namespace should not start with http://www.clarin.eu, http://clarin.eu, etc. unless the foreign namespace is introduced by the owner of the domain clarin.eu.

A detailed specification of the above-mentioned parts of a CMD instance is given in the next four clauses.

EXAMPLE CMD instance envelope.

This example shows the main structure of a CMD instance.

```
<cmd:CMD CMDVersion="1.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:cmd="http://www.clarin.eu/cmd/1"
```

```
    xmlns:cmdp="http://www.clarin.eu/cmd/1/profiles/clarin.eu:cr1:p_1311927752306"
    xsi:schemaLocation="http://www.clarin.eu/cmd/1
         http://www.clarin.eu/cmd/1/xsd/cmd-envelop.xsd
         http://www.clarin.eu/cmd/1/profiles/clarin.eu:cr1:p_1311927752306
         https://catalog.clarin.eu/ds/ComponentRegistry/rest/registry/profiles/cla
         rin.eu:cr1:p_1311927752306/1.2/xsd">
      <cmd:Header>
          …
      </cmd:Header>
      <cmd:Resources>
          <cmd:ResourceProxyList>
              …
          </cmd:ResourceProxyList>
          <cmd:JournalFileProxyList>
              …
          </cmd:JournalFileProxyList>
          <cmd:ResourceRelationList>
              …
          </cmd:ResourceRelationList>
      </cmd:Resources>
      <cmd:IsPartOfList>
          …
      </cmd:IsPartOfList>
      <cmd:Components>
          …
      </cmd:Components>
  </cmd:CMD>
```

5.3 The <Header> element

The header of a CMD instance mainly contains administrative information about the metadata, that is metadata about the CMD instance itself. The included elements shall follow the structure and order described in Table 3.

Table 3 — <Header> element: order of child elements

Name	Value type	Occurrences	Description
<cmd:Header>	xs:complexType		Encapsulates core administrative data about the CMD instance.
<cmd:MdCreator>	xs:string	0 to unbounded	Denotes the creator of this metadata file.
<cmd:MdCreationDate>	xs:date	0 or 1	The date this metadata file was created.
<cmd:MdSelfLink>	xs:anyURI	0 or 1	A reference to this metadata file in its home repository, in the form of a persistent identifier (RECOMMENDED) or a Uniform Resource Identifier.
<cmd:MdProfile>	xs:anyURI	1	The CMD profile upon which this metadata file is based, given by its identifier in a component registry.
<cmd:MdCollectionDisplayName>	xs:string	0 or 1	The collection to which the described resource belongs, given as a human-readable name. Exploitation tools can use this name to present metadata collections.

EXAMPLE Header with foreign attribute.

This example shows the header of a CMD instance, including the use of a foreign attribute, i.e., containing the ORCID id of the creator.

```
<cmd:Header>
     <cmd:MdCreator orcid:id="http://orcid.org/0000-0001-5727-2427"
        xmlns:orcid="http://www.orcid.org/ns/orcid">John Doe</cmd:MdCreator>
     <cmd:MdCreationDate>2012-04-17</cmd:MdCreationDate>
     <cmd:MdSelfLink>hdl:1234/567890</cmd:MdSelfLink>
     <cmd:MdProfile>clarin.eu:cr1:p_1311927752306</cmd:MdProfile>
     <cmd:MdCollectionDisplayName>CLARIN-NL web
        services</cmd:MdCollectionDisplayName>
</cmd:Header>
```

5.4 The <Resources> element

5.4.1 General structure of the <Resources> element

This section of the CMDI file provides the sequence of

— files which are parts of or closely related to the described resource (<cmd:ResourceProxyList> and <cmd:JournalFileProxyList>),

— possible relations between pairs of these files (<cmd:ResourceRelationList>),

and shall follow the structure and order described in Table 4.

Table 4 — <Resources> element: order of child elements

Name	Value type	Occurrences	Description
<cmd:Resources>	xs:complexType		Includes 3 lists containing information about resource proxies and their interrelations.
<cmd:ResourceProxyList>	xs:complexType	1	A list of <cmd:ResourceProxy> elements, each referencing a file contained in or closely related to the described resource.
<cmd:JournalFileProxyList>	xs:complexType	1	A list of <cmd:JournalFileProxy> elements, each referencing a file (“journal file”) containing provenance information about the described resource.
<cmd:ResourceRelationList>	xs:complexType	1	A list of <cmd:ResourceRelation> elements, each representing a relationship between 2 resource files (as listed in the <cmd:ResourceProxyList>)

5.4.2 The list of resource proxies

<cmd:ResourceProxyList> contains a sequence of zero or more occurrences of <cmd:ResourceProxy>, each representing a file/part of the described resource, and shall follow the structure and order described in Table 5.

Table 5 — <ResourceProxyList> element: order of child elements

Name	Value type	Occurrences	Description
<cmd:ResourceProxyList>	xs:complexType		Contains a list of resource proxies (see next row).
<cmd:ResourceProxy>	xs:complexType	0 to unbounded	Represents a file which is a part of or closely related to the described resource.
@id	xs:ID	1	Local identifier for the parent <cmd:ResourceProxy>, unique within this CMD instance.
<cmd:ResourceType>	xs:string (“Resource”, “Metadata”, “LandingPage”, “SearchService”, “SearchPage”; see the description for each of the possible values)	1	The type of the file represented by this <cmd:ResourceProxy>.
@mimetype	xs:string	0 or 1	The media type of the file.
<cmd:ResourceRef>	xs:anyURI	1	A reference to the file represented by this <cmd:ResourceProxy>, in the form of a persistent identifier (**RECOMMENDED**) or a Uniform Resource Identifier.

Resource types

— **Resource**
A resource that is described in the present CMD instance, e.g., a text document, media file or tool.

— **Metadata**
A metadata resource, i.e., another CMD instance, that is subordinate to the present CMD instance. The media type of this metadata resource should be `application/x-cmdi+xml`.

— **SearchPage**
Resource that is a web page that allows the described resource to be queried by an end-user.

— **SearchService**
A resource that is a web service that allows the described resource to be queried by means of dedicated software.

— **LandingPage**
A resource that is a web page that provides the original context of the described resource, e.g., a “deep link” into a repository system.

The most general value of ResourceType is that of a resource. Other values are more specific and should be selected only if they are applicable. This way, the value resource is consistent with the use of *resource* in this document, usually not enclosing metadata, landing pages, etc. that are not part of the resource to be described.

5.4.3 The list of journal files

<cmd:JournalFileProxyList> contains a sequence of zero or more occurrences of <cmd:JournalFileProxy>, each representing a file containing provenance information about the described resource, and shall follow the structure and order described in Table 6.

Table 6 — `<JournalFileProxyList>` element: order of child elements

Name	Value type	Occurrences	Description
<cmd:JournalFileProxyList>	xs:complexType		Contains a list of journal file proxies (see next row).
<cmd:JournalFileProxy>	xs:complexType	0 to unbounded	Represents a file containing provenance information about the described resource.
<cmd:JournalFileRef>	xs:anyURI	1	A reference to the file represented by this <cmd:JournalFileProxy>, in the form of a persistent identifier (**RECOMMENDED**) or a Uniform Resource Identifier.

NOTE The actual content and layout of the journal file is outside the scope of this document.

5.4.4 The list of relations between resource files

`<cmd:ResourceRelationList>` contains a sequence of zero or more occurrences of `<cmd:ResourceRelation>`, each representing a relation between any pair of `<cmd:ResourceProxies>`, and shall follow the structure and order described in Table 7.

If these parts are present they shall appear in the order given in Table 7.

Table 7 — `<ResourceRelationList>` element: order of child elements

Name	Value type	Occurrences	Description
<cmd:ResourceRelationList>	xs:complexType		Contains a list of resource relations (see next row).
<cmd:ResourceRelation>	xs:complexType	0 to unbounded	A representation of a relation between 2 resource proxies listed in <cmd:ResourceProxyList>.
<cmd:RelationType>	xs:string	1	The type of the relation represented by its parent <cmd:ResourceRelation>.
@ConceptLink	xs:anyURI	0 or 1	A reference to some concept registry (e.g. CLARIN Concept Registry[17]), indicating the semantics of <cmd:RelationType>.
<cmd:Resource>	xs:complexType	2	References one of the resource proxies participating in the relationship.
@ref	xs:IDREF	1	A reference to the <cmd:ResourceProxy> with id=ref (the <cmd:ResourceProxy> represented by its parent <cmd:Resource> element).
<cmd:Role>	xs:string	0 or 1	Indicates the role its parent resource plays in the relationship.
@ConceptLink	xs:anyURI	0 or 1	A reference to some concept registry (e.g. CLARIN Concept Registry[17]), indicating the semantics of <cmd:Role>.

EXAMPLE 1 Resources.

This example shows a list of resources of various types.

```
<cmd:Resources>
     <cmd:ResourceProxyList>
         <cmd:ResourceProxy id="lp_0000000001">
             <cmd:ResourceType
                 mimetype="application/x-httpd-php"
             >LandingPage</cmd:ResourceType>
             <cmd:ResourceRef>
                http://hdl.handle.net/11858/00-1779-0000-0007-D919-0
             </cmd:ResourceRef>
         </cmd:ResourceProxy>
         <cmd:ResourceProxy id="sru_0000000001">
             <cmd:ResourceType
                 mimetype="application/sru+xml"
             >SearchService</cmd:ResourceType>
             <cmd:ResourceRef
             >https://clarin.phonetik.uni-muenchen.de/BASSRU/</cmd:ResourceRef>
         </cmd:ResourceProxy>
         <cmd:ResourceProxy id="c_0000000001">
             <cmd:ResourceType
                 mimetype="application/x-cmdi+xml"
             >Metadata</cmd:ResourceType>
             <cmd:ResourceRef>
                https://clarin.phonetik.uni-
                muenchen.de/BASRepository/Public/Corpora/ZIPTEL/0001.1.cmdi.xml
             </cmd:ResourceRef>
         </cmd:ResourceProxy>
         <cmd:ResourceProxy id="h0">
             <cmd:ResourceType>Resource</cmd:ResourceType>
             <cmd:ResourceRef>hdl:1839/00-SERV-0000-0000-0009-D</cmd:ResourceRef>
         </cmd:ResourceProxy>
         <cmd:ResourceProxy id="h1">
             <cmd:ResourceType
                 mimetype="application/vnd.sun.wadl+xml"
             >Resource</cmd:ResourceType>
             <cmd:ResourceRef>
                http://catalog.clarin.eu/adelheidws/wadl/main.wadl</cmd:ResourceRef
         </cmd:ResourceProxy>
     </cmd:ResourceProxyList>
     <cmd:JournalFileProxyList/>
     <cmd:ResourceRelationList/>
</cmd:Resources>
```

EXAMPLE 2 A minimally specified relation between resource files.

```
<cmd:ResourceRelation>
    <cmd:RelationType>duplicates</cmd:RelationType>
    <cmd:Resource ref="_395">
    <cmd:Resource ref="_394"/>
</cmd:ResourceRelation>
```

EXAMPLE 3 A maximally specified relation between resource files.

This example shows a semantically rich specification of a relationship between two resources, i.e., relation type and roles are annotated with concept references from various semantic registries.

```
<cmd:ResourceRelation>
     <cmd:RelationType
         ConceptLink="http://www.w3.org/ns/oa#describing"
     >describing</RelationType>
     <cmd:Resource ref="rp1">
        <cmd:Role
         ConceptLink="http://hdl.handle.net/11459/CCR_C-6024_88c0ac12-c24b-dd5a-
         d183-07e6dae25c52"
        >source</cmd:Role>
    </cmd:Resource>
```



```
    <cmd:Resource ref="rp2">
        <cmd:Role
         ConceptLink="http://hdl.handle.net/11459/CCR_C_c4e689ff-3724-10f7-8eb5-
         aee00b313f5f"
        >target</cmd:Role>
    </cmd:Resource>
 </cmd:ResourceRelation>
```

5.5 The `<IsPartOfList>` element

`<cmd:IsPartOfList>` contains a sequence of zero or more occurrences of `<cmd:IsPartOf>`, each representing an external resource of which the described resource constitutes a part, and shall follow the structure and order described in Table 8.

Table 8 — `<IsPartOfList>` element: order of child elements

Name	Value type	Occurrences	Description
`<cmd:IsPartOfList>`	`xs:complexType`		Contains a list of `<cmd:IsPartOf>` (see next row).
`<cmd:IsPartOf>`	`xs:anyURI`	0 to unbounded	A reference to an external resource of which the described resource is a part, in the form of a persistent identifier (**RECOMMENDED**) or a Uniform Resource Identifier.

NOTE The inverse of the IsPartOf can be indicated by a resource proxy with resource type Metadata in the instance that describes the composite.

EXAMPLE The IsPartOf List.

This example shows an IsPartOf referring to another CMD instance, which describes the collection this instance is a part of.

```
<cmd:IsPartOfList>
     <cmd:IsPartOf>hdl:11858/00-1779-0000-0006-BF00-E@format=cmdi</cmd:IsPartOf>
</cmd:IsPartOfList>
```

5.6 The CMD components

This section of the CMD instance forms what may be referred to as descriptive metadata about the described resource.

The CMD profile referenced by the XML element `<cmd:MdProfile>` in `<cmd:Header>` defines what XML elements and XML attributes are mandatory or optional in this section. Some attributes may appear universally in XML elements contained in any CMD instance payload section regardless of the CMD profile, but rather depending on the corresponding level in the matching CMD profile, i.e., whether the XML element is reflecting a CMD component or CMD element. Table 9 describes the mandatory structure and order of this section as a function of the definition of a specific CMD profile.

Table 9 — The `<Components>` element of a CMD instance: basic structure of content

Name	Value type	Occurrences	Description
`<cmd:Components>`	`xs:complexType`		Container for the CMD instance payload.
`<cmdp:{RootComponent}>`	`xs:complexType`	1	The XML element housing all the metadata about the described resource, complying with the CMD profile schema identified in the `<cmd:MdProfile>` element in the CMD instance header.

Table 9 *(continued)*

Name	Value type	Occurrences	Description
@cmd:ref	xs:IDREF	0 or 1	Reference to a <cmd:ResourceProxy> with @id equal to the value of this attribute, to which this substructure specifically applies.
@{CMDAttribute}*	As specified in the CMD profile	As specified in the CMD profile	Custom attribute, defined as an allowed or mandatory child in a CMD specification.
<cmdp:{CMDElement}>*	As specified in the CMD profile	As specified in the CMD profile	Atomic piece of information about the described resource.
@xml:lang	xs:language	0 or 1	Indicates the language of the <cmdp:{CMDElement}> content by a language tag.
@cmd:ValueConceptLink	xs:anyURI	0 or 1	Reference to a concept in an external vocabulary. Used in case the value <cmdp:{CMDElement}> is selected from a controlled vocabulary.
@{CMDAttribute}*	As specified in the CMD profile	As specified in the CMD profile	Custom attribute, defined as an allowed or mandatory child in the specification of a CMD element.
<cmdp:{CMDComponent}>*	xs:complexType	As specified in the CMD profile	A chunk of information about the described resource, composed of CMD elements and other CMD components.
@cmd:ComponentId	xs:anyURI	0 or 1	Identifier of the CMD specification of <cmdp:{CMDComponent}> in a CMD component registry.
@cmd:ref	xs:IDREF	0 or 1	Reference to a <cmd:ResourceProxy> with @id equal to the value of this attribute, to which this substructure specifically applies.
@{CMDAttribute}*	As specified in the CMD profile	As specified in the CMD profile	Custom attribute, defined as an allowed or mandatory child in a CMD specification.
<cmdp:{CMDElement}>*	As specified in the CMD profile	As specified in the CMD profile	Atomic piece of information related to the described resource and forming a part of its parent CMD component.
<cmdp:{CMDComponent}>*	xs:complexType	As specified in the CMD profile	A chunk of information related to the described resource, forming a part of its parent CMD component and further composed of CMD elements and other CMD components.

EXAMPLE CMD instance payload.

This example shows various (optional) aspects of the payload of a CMD instance: the use of language tags (@xml:lang) for multilingual elements (cmdp:description), identifiers of the specification of the instantiated components (@cmd:ComponentId), references to an item from a vocabulary (@cmd:ValueConceptLink), and CMD components, CMD elements and CMD attributes defined in a CMD profile (cmdp:* and @*).

```
<cmd:Components>
     <cmdp:ToolService>
         …
         <cmdp:GeneralInfo>
             <cmdp:Name>Adelheid</cmdp:Name>
                 <cmdp:Description xml:lang="en">A web-application with which an end user
can have historical Dutch text tokenized, lemmatized and part-of-speech tagged, using the
most appropriate resources (such as lexica) for the text in question.</cmdp:Description>
             </cmdp:Description>
```



```
             <cmdp:Description xml:lang="nl">Een webapplicatie waarmee een eindgebruiker
teksten in oud nederlands kan laten tokeniseren, lemmatiseren en ontleden, gebruikmakend
van de resources (zoals lexica) die het beste bij die tekst passen.</cmdp:Description>
         </cmdp:GeneralInfo>
         …
         <cmdp:Contact
              cmd:ComponentId="clarin.eu:cr1:c_1271859438113">
             <cmdp:Person>Drs. D. Broeder</cmdp:Person>
             <cmdp:Address>Wundtlaan 1, 6525 XD Nijmegen, The
                Netherlands</cmdp:Address>
             <cmdp:Email>Daan.Broeder@mpi.nl</cmdp:Email>
         <cmdp:Organisation
           cmd:ValueConceptLink="
                  http://openskos.meertens.knaw.nl/Organisations/8c778a30-
                  f607-45fd-838d-1ea00cea9150">
                  Max Planck Institute for Psycholinguistics (MPI)
      </cmdp:Organisation>
             <cmdp:Telephone Type="work"
             >+31 - 00 - 1234567</cmdp:Telephone>
         </cmdp:Contact>
         …
         <cmdp:Tool CoreVersion="1.0" cmd:ref="h0">
             <cmdp:toolInput>
                 <cmdp:data>
                       <cmdp:MimeType>text/xml</cmdp:MimeType>
                       <cmdp:characterEncoding>UTF-8</cmdp:characterEncoding>
                       <cmdp:datatype>ATL XML</cmdp:datatype>
                   </cmdp:data>
                </cmdp:toolInput>
                …
         </cmdp:Tool>
     </cmdp:ToolService>
</cmd:Components>
```

6 CCSL (CMDI Component Specification Language)

6.1 General structure of the CCSL

CCSL, the CMDI Component Specification Language, is used to describe a CMD component or CMD profile. Hence, a CCSL document provides the structure for describing an aspect of a resource or (in the case of a CMD profile) the complete CMD instance payload. It is also the basis for the generation of the XML schema file that is used to validate a CMD instance. See Figure 3 and 7.1 for details.

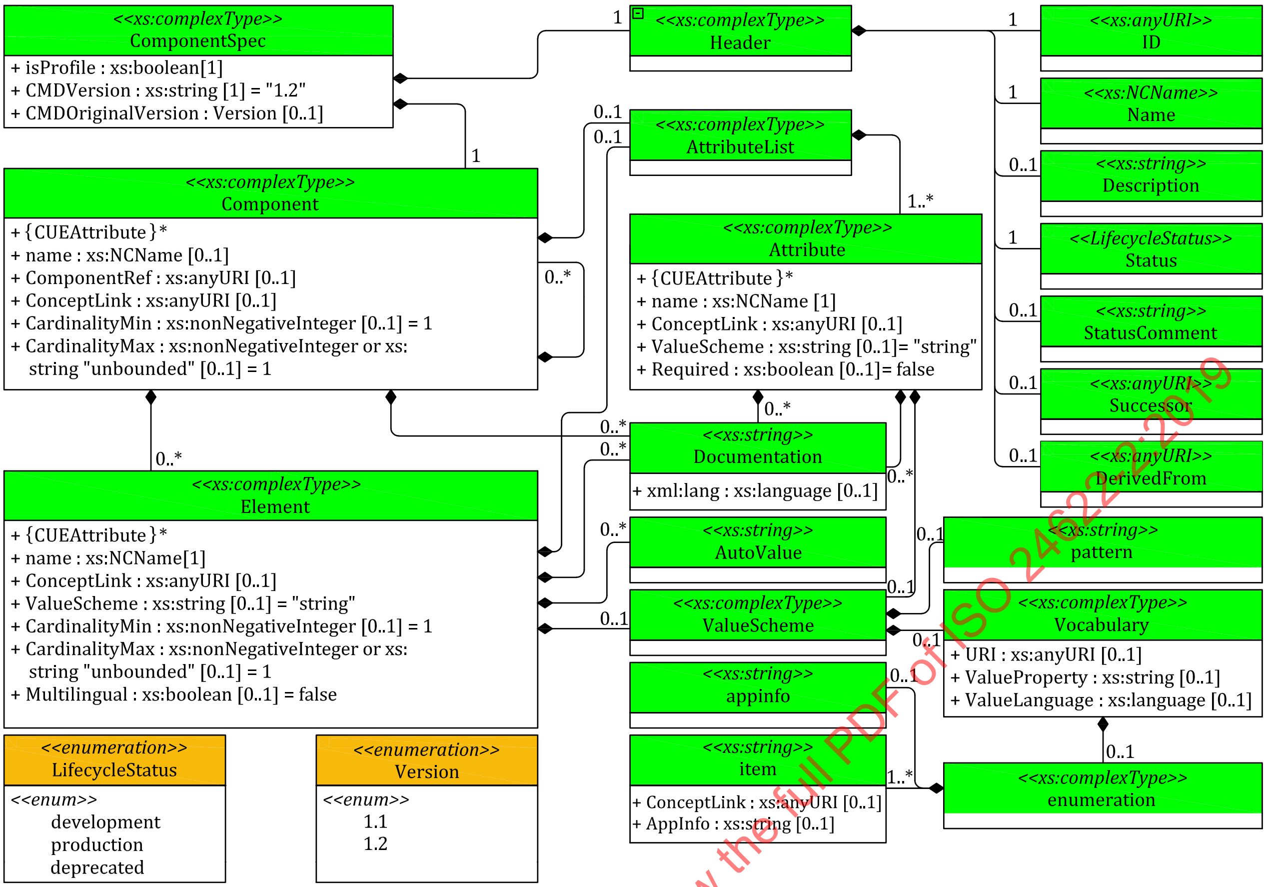

Figure 3 — Structure of a CCSL document

A CCSL document shall contain a CCSL header and the actual CMD component description. Its root element shall contain an XML attribute `@isProfile` to indicate if the document specifies a CMD profile or a CMD component and it shall contain an XML attribute `@CMDVersion` specifying the CMDI version ("1.2"). The root element may also contain an XML attribute `@CMDOriginalVersion` specifying the CMDI version that was originally used to create the component.

Table 10 describes the root element and its direct descendants. The described structure and order shall be followed.

Table 10 — Root element of CCSL documents: structure

Name	Value type	Occurrences	Description
<ComponentSpec>	xs:complexType		Root element of the CCSL document.
@isProfile	xs:boolean	1	Indication about the specification's status as a CMD profile.
@CMDVersion	xs:string ("1.2")	1	CMDI version of this CMD specification.
@CMDOriginalVersion	xs:string ("1.1", "1.2")	0 or 1	CMDI version in which the CMD specification was created (default: 1.2).
<Header>	xs:complexType	1	Header of the CMD specification.
<Component>	xs:complexType	1	Definition of a component's structure (the CMD root component in case of a CMD profile).

EXAMPLE   CCSL document.

This example shows the main structure of a CCSL document.

```
<ComponentSpec isProfile="true" CMDVersion="1.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:cue="http://www.clarin.eu/cmdi/cues/1"
   xsi:noNamespaceSchemaLocation="https://infra.clarin.eu/CMDI/1.x/xsd/cmd-component.xsd">
     <Header>
         …
     </Header>
     <Component CardinalityMax="1" CardinalityMin="1"
        name="ToolService">
         …
     </Component>
</ComponentSpec>
```

6.2 CCSL header

The CCSL header provides information relevant to identify and describe the component. This part includes a persistent identifier, the name, the description of the component and information about the status of the specification. The header shall contain an element indicating the component's status in its lifecycle (using the three lifecycles *development*, *production*, or *deprecated*) and may contain the element `<StatusComment>` to contain information about the reason for the current status. In the case of a deprecated specification that was succeeded by a new specification, the identifier of the direct successor may be stored in the element `<Successor>`.

Table 11 describes the header element and its direct descendants. The described structure and order shall be followed.

Table 11 — CCSL document: header element structure

Name	Value type	Occurrences	Description
<Header>	xs:complexType		Descriptive information about the CMD component.
<ID>	xs:anyURI	1	ID of the CMD specification.
<Name>	xs:NCName	1	Name of the CMD component.
<Description>	xs:string	0 or 1	Description of the CMD component.
<Status>	xs:string ("development", "production", "deprecated"; see the description of each of the possible values)	1	Status in lifecycle.
<StatusComment>	xs:string	0 or 1	Comment about the status.
<Successor>	xs:anyURI	0 or 1	ID of successor CMD component, if available.
<DerivedFrom>	xs:anyURI	0 or 1	ID of CMD component from which this CMD component is derived, if available.

Status values

— **development**
The CMD specification is under construction, i.e., can undergo change at any moment, and therefore only to be used for testing purposes.

— **production**
The CMD specification is stable and will not be changed anymore, i.e., can be used for production-level CMD instances.

— **deprecated**
Usage of this CMD specification is discouraged, and usage of a successor CMD specification, if present, is encouraged.

Additional constraints

— A successor should only be present if the status of the CMD component is deprecated.

EXAMPLE 1 CCSL header.

This example shows the header of a CCSL document.

```
<Header>
     <ID>clarin.eu:cr1:p_1311927752306</ID>
     <Name>ToolService</Name>
     <Description>Description of a tool and/or service(s)</Description>
     <Status>production</Status>
</Header>
```

EXAMPLE 2 CCSL header for deprecated profile with successor.

This example shows the header of a CCSL document for a deprecated profile with also a reference to its successor.

```
<Header>
     <ID>clarin.eu:cr1:p_1311927752306</ID>
     <Name>ToolService</Name>
     <Description>Description of a tool and/or service(s)</Description>
     <Status>deprecated</Status>
     <Successor>clarin.eu:cr1:p_1234567890</Successor>
</Header>
```

6.3 CMD specification

CMD components are defined as a sequence of CMD elements which may be followed by other CMD components. The latter is allowed because CMD components may be included in other CMD components, either by referencing already defined components (i.e. a CMD component with its own identifier) or providing an inline CMD component definition. The former shall be done by assigning the identifier of the referenced CMD component as the value of `@ComponentRef`.

Table 12 describes the element for defining CMD components and its direct descendants. The described structure and order shall be followed.

Table 12 — CCSL document: CMD specification

Name	Value type	Occurrences	Description
<Component>	xs:complexType		Root element of every CMD specification.
@name	xs:NCName	0 or 1	Name of the CMD component.
@ComponentRef	xs:anyURI	0 or 1	Reference to an existing CMD specification with <ID> equal to the value of this attribute.
@ConceptLink	xs:anyURI	0 or 1	Concept link.
@CardinalityMin	xs:nonNegativeInteger	0 or 1	Minimum number of times this CMD component has to occur (default: 1).
@CardinalityMax	xs:nonNegativeInteger or “unbounded”	0 or 1	Maximum number of times this CMD component may occur (default: 1).
<Documentation>	xs:string	0 to unbounded	Documentation about the purpose of the CMD component.
@xml:lang	xs:language	0 or 1	The language tag of the language used by the documentation.
<AttributeList>	xs:complexType	0 or 1	Additional CMD attributes specified by the creator of a CMD component.

Table 12 *(continued)*

Name	Value type	Occurrences	Description
<Attribute>	xs:complexType	1 to unbounded	An additional CMD attribute.
<Element>	xs:complexType	0 to unbounded	The CMD elements of this CMD component.
<Component>	xs:complexType	0 to unbounded	The CMD components nested in this CMD component.

Additional constraints

— A CMD component shall have either a name or a reference to an existing CMD component.

— An inline CMD component should contain at least one CMD element or CMD component.

— For the CMD component that is the direct descendant of <ComponentSpec>, the minimum and maximum cardinalities shall both be 1.

— The value of the minimum cardinality shall be lower or equal to the value of the maximum cardinality.

— For this CMD component, each documentation shall have a unique @xml:lang value; and there shall not be more than one documentation with an empty or missing @xml:lang.

— Within the attribute list each CMD attribute shall have a unique name.

— The CMD elements and CMD components, which are direct descendants of this component, shall all have different names.

— A CMD component shall not be a descendant of itself.

EXAMPLE CMD specification.

This example shows a definition for a CMD component including documentation in two languages.

```
<Component
    ComponentId="clarin.eu:cr1:c_1320657629631"
    name="Service"
    ConceptLink="http://hdl.handle.net/11459/CCR_C-4159_ca0e6cba-cab5-b51a-f430-
        fdcb0756c9ac" CardinalityMin="0" CardinalityMax="unbounded"
      <Documentation xml:lang="en">A web service which is described in enough
      detail to enable automatic invocation for machine
      interaction.</Documentation>
      <Documentation xml:lang="nl">Een webservice, gedetailleerd genoeg
      beschreven om het mogelijk te maken de service automatisch aan te laten
      roepen voor machine-
      interactie.</Documentation>
     <AttributeList>
        …
     </AttributeList>
     …
 </Component>
```

6.4 Definition of CMD elements

CMD elements are a template for storing atomic values governed by a value scheme in a CMD instance. The CCSL specification of a CMD element shall contain the name of the element and may contain a concept link, the value schema, and information about the allowed cardinality of the CMD element. Furthermore, it may be indicated if the CMD element allows for values in more than one language, in which case an unlimited upper cardinality bound is implied. A CMD element shall either have one of the standard XML schema datatypes assigned to it or be constrained by using regular expressions or controlled vocabularies. The latter can be specified by giving the complete list of allowed values or by stating the Uniform Resource Identifier of an external controlled vocabulary (for details see 6.6). If the CMD instance's content of the element can be derived from other values, the element <AutoValue>

may be used to give indication about the derivation function. The CCSL does not prescribe or suggest a specific set of derivation functions.

Table 13 describes the element for defining CMD elements and its direct descendants. The described structure and order shall be followed.

Table 13 — CCSL document: definition of CMD elements

Name	Value type	Occurrences	Description
<Element>	xs:complexType		Root element of every CMD element definition.
@name	xs:NCName	1	Name of the CMD element.
@ConceptLink	xs:anyURI	0 or 1	Concept link.
@ValueScheme	xs:string (name of an XML Schema datatype)	0 or 1	Allowed data type (default: string).
@CardinalityMin	xs:nonNegativeInteger	0 or 1	Minimum number of times this CMD element has to occur (default: 1).
@CardinalityMax	xs:nonNegativeInteger or “unbounded”	0 or 1	Maximum number of times this element may occur (default: 1).
@Multilingual	xs:boolean	0 or 1	Indication that the CMD element can have values in multiple languages (default: false).
<Documentation>	xs:string	0 to unbounded	Documentation about the purpose of the CMD element.
@xml:lang	xs:language	0 or 1	The language tag of the language used by the documentation.
<AttributeList>	xs:complexType	0 or 1	Additional CMD attributes specified by the creator of the CMD component.
<Attribute>	xs:complexType	1 to unbounded	An additional CMD attribute.
<ValueScheme>	xs:complexType	0 or 1	Value scheme based on a regular expression or a specified controlled vocabulary. See 6.6 for details.
<AutoValue>	xs:string	0 to unbounded	Derivation rules for the CMD element’s content.

Additional constraints

— For the defined CMD element, each documentation shall have a unique `@xml:lang` value; and there shall not be more than one documentation with an empty or missing `@xml:lang`.

— A CMD element should have either a `@ValueScheme` or a `<ValueScheme>`.

— The value of the minimum cardinality shall be lower or equal to the value of the maximum cardinality.

— Within the attribute list each CMD attribute shall have a unique name.

NOTE If `@Multilingual` has the value true and `@ValueScheme` has the value `string`, the value of `@CardinalityMax` shall be ignored and defaults to unbounded.

If `@ValueScheme` does not have the value `string`, the value of multilingual shall be ignored.

If the CMD element has a `<ValueScheme>` the data type defaults to `string`.

EXAMPLE 1 CMD element definition.

This example shows the definition of a CMD element.


```
<Element
         name="Name"
         ConceptLink="http://hdl.handle.net/11459/CCR_C-4160_192be757-0d8f-f4fe-
         b10b-d3d50de92482"
         CardinalityMin="1"
         CardinalityMax="1"
         ValueScheme="string"
         Multilingual="false">
       <Documentation>The name of the web service or set of web services.
       </Documentation>
</Element>
```

EXAMPLE 2 CMD element definition with auto value.

This example shows the definition of a CMD element with an (informative) auto value derivation rule, i.e., instantiates the element with the date and time at the moment of creation.

```
<Element
    name="CreationDate"
    ValueScheme="dateTime">
      <!-- 'now' is an informative example of an
        derivation function -->
      <AutoValue>now</AutoValue>
</Element>
```

6.5 CMD attribute definition

Both the CMD element and component description allow the specification of additional CMD attributes. Every CMD attribute definition shall contain a @name attribute and may contain other attributes or elements for a more detailed description.

Table 14 describes the element for defining CMD attributes and its direct descendants. The described structure and order shall be followed.

Table 14 — CCSL document: attribute definition

Name	Value type	Occurrences	Description
<Attribute>	xs:complexType		Root element of every CMD attribute definition.
@name	xs:NCName	1	Name of the CMD attribute.
@ConceptLink	xs:anyURI	0 or 1	Concept link.
@ValueScheme	xs:string (name of an XML Schema datatype)	0 or 1	Allowed data type (default: string).
@Required	xs:boolean	0 or 1	Indication if CMD attribute is required (default: false).
<Documentation>	xs:string	0 to unbounded	Documentation about the purpose of the CMD attribute.
@xml:lang	xs:language	0 or 1	The language tag of the language used by the documentation.
<ValueScheme>	xs:complexType	0 or 1	Value scheme based on a regular expression or a specified controlled vocabulary. See 6.6 for details.
<AutoValue>	xs:string	0 to unbounded	Derivation rules for the CMD attribute's content.

Additional constraints

For the defined CMD attribute, each documentation shall have a unique `@xml:lang` value; and there shall not be more than one documentation with an empty or missing `@xml:lang`.

A CMD attribute should have either a `@ValueScheme` or a `<ValueScheme>`.

NOTE If the CMD attribute has a `<ValueScheme>`, the data type defaults to `string`.

EXAMPLE CMD attribute definition.

This example shows a definition of a CMD attribute.

```
<Attribute
   name="CoreVersion"
   ConceptLink="http://hdl.handle.net/11459/CCR_C-2547_7883d382-b3ce-8ab4-7052-
     0138525a8ba1"
   Required="true">
     <ValueScheme>
         …
     </ValueScheme>
</Attribute>
```

6.6 Value schemes for CMD elements and CMD attributes

Apart from standard XML schema datatypes, the content of a CMD element or CMD attribute instance can be restricted or guided by two means. The `<ValueScheme>` element may contain either an XML element `<pattern>` with the specification of a regular expression the element/attribute should comply with, or the definition of a controlled vocabulary. CMDI 1.2 supports two types of controlled vocabularies:

— a closed vocabulary where all allowed values are specified with optional attributes for every value to include a concept link and a description of the specific value, or

— an open vocabulary by referring to an external controlled vocabulary via a Uniform Resource Identifier specified in `@URI`, where, in this case, the external vocabulary provides suggested values.

A closed vocabulary may also refer to an external vocabulary via a Uniform Resource Identifier, where the external service may be used during metadata creation or search. The optional XML attributes `@ValueProperty` and `@ValueLanguage` can be used to give more information about the property and language to be used in the specified external controlled vocabulary.

The order and structure described in Table 15 shall be followed when specifying value schemes.

Table 15 — CCSL document: value scheme definition

Name	Value type	Occurrences	Description
`<ValueScheme>`	`xs:complexType`		Specification of the value scheme of an element or attribute.
│ `<pattern>`	`xs:string`	0 or 1	Specification of a regular expression the element/attribute should comply with.
│ `<Vocabulary>`	`xs:complexType`	0 or 1	Specification of a controlled vocabulary.
│ │ `@URI`	`xs:anyURI`	0 or 1	Uniform Resource Identifier of an external controlled vocabulary.

Table 15 *(continued)*

Name	Value type	Occurrences	Description
@ValueProperty	xs:string	0 or 1	Property in the external vocabulary entry that provides the value.
@ValueLanguage	xs:language	0 or 1	Preferred language in the external vocabulary.
<enumeration>	xs:complexType	0 or 1	Enumeration of items from a controlled vocabulary.
<appinfo>	xs:string	0 to 1	End-user guidance about the value of the controlled vocabulary as a whole.
<item>	xs:string	1 to unbounded	An item from a controlled vocabulary.
@ConceptLink	xs:anyURI	0 or 1	Concept link of item value.
@AppInfo	xs:string	0 or 1	End-user guidance about the value of this controlled vocabulary item.

Additional constraints

— In an enumeration, each item value shall be unique.

— A <ValueScheme> shall have either a <pattern>, or a <Vocabulary> with a non-empty <enumeration>, or a @URI.

NOTE A controlled vocabulary with a non-empty enumeration of permissible values constitutes a closed vocabulary. Using @URI, an external controlled vocabulary provided by a vocabulary service, e.g., the CLARIN vocabulary service CLAVAS[19], can be associated with the closed vocabulary, which allows tools to use the service's facilities to find a value.

The @URI can also be used for an open vocabulary where the facilities of the vocabulary service can be used to find suggestions for an applicable value.

EXAMPLE 1 Value scheme with enumeration (closed vocabulary).

This example shows a value scheme with a reference to an external controlled vocabulary and an embedded enumeration, i.e., a closed vocabulary.

```
<ValueScheme>
     <Vocabulary URI="http://openskos.meertens.knaw.nl/iso-639-3"
       ValueProperty="skos:notation">
         <enumeration>
               <item AppInfo="Ghotuo (aaa)"
                   ConceptLink="http://cdb.iso.org/lg/CDB-00132443-001">aaa</item>
               <item AppInfo="Alumu-Tesu (aab)"
                   ConceptLink="http://cdb.iso.org/lg/CDB-00133770-001">aab</item>
               <item AppInfo="Ari (aac)"
                   ConceptLink="http://cdb.iso.org/lg/CDB-00133769-001">aac</item>
               <item AppInfo="Amal (aad)"
                   ConceptLink="http://cdb.iso.org/lg/CDB-00133768-001">aad</item>
               <item AppInfo="Arbëreshë Albanian (aae)"
                   ConceptLink="http://cdb.iso.org/lg/CDB-00133767-001">aae</item>
               <item AppInfo="Aranadan (aaf)"
                   ConceptLink="http://cdb.iso.org/lg/CDB-00133766-001">aaf</item>
    …
         </enumeration>
     </Vocabulary>
</ValueScheme>
```

EXAMPLE 2 Value scheme without enumeration but with external vocabulary (open vocabulary).

This example shows a value scheme with a reference to an external controlled vocabulary without an embedded enumeration, i.e., an open vocabulary. The external vocabulary suggests preferred labels for known organizations, but it is allowed to use other names (preferably the names of unknown organizations).

```
<ValueScheme>
        <Vocabulary URI="http://openskos.meertens.knaw.nl/Organisations"
            ValueProperty="skos:prefLabel"/>
</ValueScheme>
```

EXAMPLE 3 Value scheme with pattern.

This example shows a value scheme with a regular expression for a time stamp.

```
<ValueScheme>
      <pattern>[0-9][0-9]:[0-9][0-9]:[0-9][0-9]:?[0-9]*</pattern>
</ValueScheme>
```


6.7 Cue attributes

CMDI profiles provide the blueprint for a logical structuring of CMD instances. However, they provide very little explicit information about how the information contained or to be entered in CMD instances should be dealt with in applications that process or generate CMD instances. CMDI 1.2 therefore allows for the augmentation of CMD components, CMD elements and CMD attributes in the definitions of CMD profiles with ‘cues for tools’ that provide suggestions for the way metadata content could be presented (e.g. by specifying certain typographical characteristics or specifying a set of elements that can be grouped together visually) or handled in some other way (e.g. by enabling or disabling spell checking or using a specific input method).

Such information, to be processed by viewers, editors and catalogues alike, has the potential of leading to a more uniform (across applications), visually pleasing and user-friendly mode of working with metadata. The processing of such cues should always be optional for applications handling CMD instances.

For this purpose, all specifications of CMD attributes, CMD elements, and CMD components may contain additional attributes in the cue namespace. These may be used to give information about how the CMD instance payload contained in the respective part of the CMD instance should be presented. Cues are grouped in component specific styles. Different styles for the same CMD component may be developed. The CCSL does not prescribe or suggest a specific set of cue attributes.

EXAMPLE 1 Cue for CMD element.

This example shows a cue for a CMD element, i.e., its display priority within its component and a label which can be used when multiple instantiations are shown together. Note that this is a hypothetical cue that is not necessarily supported by any specific applications.

```
<Element
   name="Name"
   …
   cue:DisplayPriority="1"
   cue:PluralLabel="Names" >
 …
</Element>
```

EXAMPLE 2 Cue for CMD component.

This example shows two potential cues for CMD components: one cue specifying an ordered preference for the display value of the “Person” component, and another cue for the “Address” component to indicate that its contents should be shown at the same level as its parent’s contents. Note that these are hypothetical cues that are not necessarily supported by any specific applications.

```
<Component name="Person"
   cue:LabelElement="Name,Initials,Id ">
     <Element name="Name" CardinalityMin ="0" />
     <Element name="Initials" CardinalityMin ="0" />
     <Element name="Id" CardinalityMin ="0" />
```

