Title	CMDI 1.2 specification
Version	1
Author(s)	CMDI Taskforce
Date	2016-10-20
Status	Final
Distribution	Public
ID	CE-2016-0880

Component Metadata Infrastructure (CMDI)

Component Metadata Specification

Version 1.2 - 20 October 2016[1]

1. 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 metadata records 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 address 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 (Broeder *et al*, 2010 and 2012).

In CMDI the metadata lifecycle starts with the need of a metadata modeller to create a dedicated metadata profile for a specific type of resources. The modeller can browse and search a registry for components and profiles that are suitable or come close to meeting her 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. A component registry, e.g., the *CLARIN Component Registry*, 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 (Durco & Windhouwer, 2013). These semantic descriptions can be stored in a semantic registry, e.g., the *CLARIN Concept Registry*. 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 (Van Uytvanck *et al,* 2012).

[1] This version corresponds with version 1.2.1 of the CMDI toolkit and any higher patch release, i.e., any release with a version number between 1.2.1 and 1.3.0

1.1. History

CMDI has been developed in the context of the European CLARIN infrastructure 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 the CMDI toolkit 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 throughout CLARIN's construction phase. 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. This specification describes version 1.2. This new version adds functionality and also fixes some issues. These changes are highlighted in CE-2014-0318. The transition from 1.1 to 1.2 is supported by version 1.2 of the CMDI toolkit.

1.2. Scope

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

The scope of this specification is to describe 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).

In ISO 24622-1:2015 the component metadata model has been standardized. The present specification is compliant with this ISO standard, and also extends and constrains it at various places (see also the red parts in the UML class diagram below):

- 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.

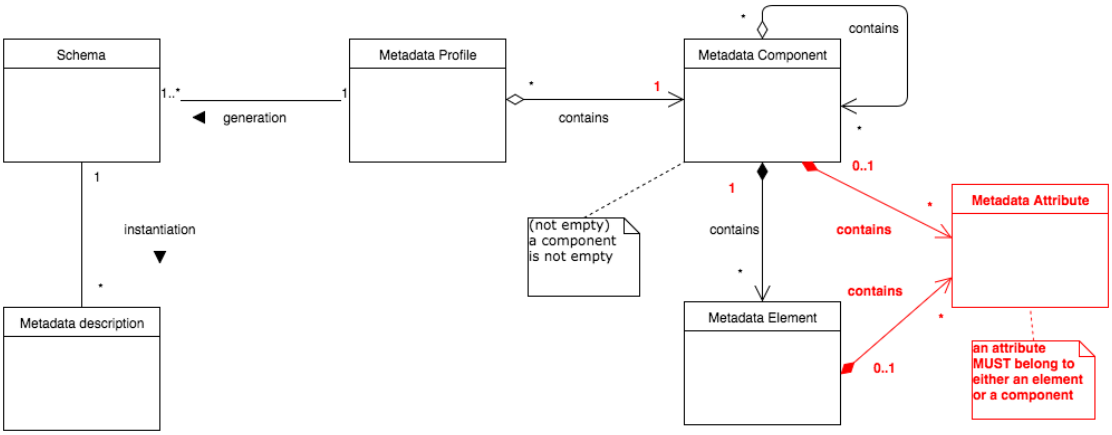

Figure 1 Component metadata model and its extensions

1.3. Terminology

The keywords **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL** in this document are to be interpreted as described in IETF RFC 2119.

1.4. Glossary

1.4.1. General

- **CLARIN infrastructure**, CLARIN
  - The infrastructure governed by the CLARIN ERIC.
- **concept**
  - An abstract idea conceived in the mind or generalised from particular instances (cf. Merriam-Webster, definition of *concept*).
- **concept link**
  - A reference from a CMD profile, CMD component, CMD element, CMD attribute or a value in a controlled vocabulary to an entry in a semantic registry via a URI, typically a persistent identifier.
- **concept registry**
  - A semantic registry maintaining concepts, e.g., the *CLARIN Concept Registry* as used in the CLARIN infrastructure.
- **controlled vocabulary**, closed/open vocabulary
  - A 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.
- **data category**
  - The result of the specification of a given data field (ISO 12620:2009).
- **language tag**
  - A textual code "used to help identify languages, whether spoken, written, signed, or otherwise signaled, for the purpose of communication. This includes constructed and artificial languages but excludes languages not intended primarily for human communication, such as programming languages." (IETF BCP 47)
- **media type**, MIME type
  - A type which specifies the nature of the data as described in IETF RFC 6838.
- **metadata**
  - A resource that is a description of another resource, usually given as a set of properties in the form of attribute-value pairs. This description may contain information about the resource, aspects or parts of the resource and/or artefacts and actors connected to the resource.
- **persistent identifier**, PID
  - Unique Uniform Resource Identifier that assures permanent access for a resource by providing access to it independently of its physical location or current ownership.
- **resource**
  - An entity, possibly digitally accessible, that can be described in terms of its content and technical properties, referenced by a Uniform Resource Identifier.

- **semantic registry**
  - A directory of (authoritative) definitions of terms, concepts or data categories, or the system maintaining it. These registries should also provide persistent identifiers for their entries.
- **term**
  - A verbal designation of a general concept in a specific subject field (ISO 1087-1:2000).
- **Uniform Resource Identifier**, URI
  - An identifier for resources as described in IETF RFC 3986.

1.4.2. CMDI

- **CCSL**, CMDI Component Specification Language
  - XML based language for describing components and profiles according to the CMD model.
- **CMD attribute**
  - A unit within a CMD element that describes the level at which properties of a CMD element can be provided by means of value scheme constrained atomic values.
- **CMD component**, component
  - A reusable, structured template for the description of (an aspect of) a resource, defined by means of a CMD specification document with the potential of including other CMD components, either through reference or inline definition.
- **CMD component registry**, component registry
  - A service where CMD specifications can be registered and accessed.
- **CMD element**, element definition
  - A unit within a CMD component that describes the level of the metadata instance that can carry atomic values governed by a value scheme, and does not contain further levels except for that of the CMD attribute.
- **CMD instance**, metadata instance, CMDI file, metadata record, CMD record
  - A file that conforms to the general CMDI instance structure as described in this specification, and at the instance payload level follows the specific structure defined by the CMD profile it relates to.
- **CMD instance envelope**
  - The section of a CMD instance which is structured uniformly for all instances, and contains the CMD instance header and the list of resource proxies which may be referenced from the CMD instance payload section.
- **CMD instance header**
  - The section of a CMD instance marked as 'header', providing information on that metadata instance as such, not the resource that is described by the metadata file.
- **CMD instance payload**
  - The section of a metadata instance that follows the structure defined by the profile it references and contains the description of the resources to which that metadata instance relates.
- **CMD model**, Component Metadata model
  - The component based metadata model described in the present specification.

- **CMD profile**, profile definition, profile
  - A structured template for the description of a class of resources providing the complete structure for an instance payload by means of a hierarchy of CMD components.
- **CMD profile schema**
  - A schema definition by which the correctness of a CMD instance with respect to the CMD profile it pertains to can be evaluated. May be expressed as XML Schema but also in other XML schema languages.
- **CMD root component**
  - The CMD component that is defined at the highest level within a CMD profile that may have one or more child components but no siblings. In the CMD instance payload, it is instantiated exactly once.
- **CMD specification**, component specification/definition, profile specification/definition
  - The representation of a CMD component or CMD profile, expressed using the constructs of CCSL.
- **CMD specification header**, component header, profile header
  - The section of a CMD specification marked as 'header', providing information on that specification as such that is not part of the defined structure.
- **CMDI**, Component Metadata Infrastructure
  - Metadata description framework consisting of the CMD model and infrastructure to process instances of (parts of) the model.
- **inline CMD component**
  - A CMD component that is created and stored within another component and cannot be addressed from other components.
- **resource proxy**, CMD resource reference
  - A representation of a resource within a metadata instance containing a Uniform Resource Identifier as a reference to the resource itself and an indication of its nature.
- **resource proxy reference**
  - A reference from any point within the instance payload to any of the resource proxies.
- **value scheme**
  - A set of constraints governing the range of values allowed for a specific CMD element or CMD attribute in a metadata instance, expressed in terms of an XML Schema datatype, controlled vocabulary, or regular expression.

1.4.3. XML

- **foreign attribute**
  - An XML attribute defined in a namespace other than those declared in CMDI, to be included in CMD instance as additional information targeted to specific receivers or applications.
- **namespace**
  - An XML namespace as described in W3C XML Namespaces.
- **regular expression**
  - An expression that constrains the set of permissible values, as described in XML Schema Regular Expressions (W3C XSD Part 2: Datatypes, appendix F Regular expressions).
- **XML**
  - Extensible Markup Language as described by W3C recommendation (W3C XML).

- **XML attribute**
  - A property of an XML element as defined in W3C XML.
- **XML attribute declaration**
  - A constituent of an XML Schema that constrains the structure and content of a specific XML attribute, in accordance with W3C XSD , section 3.2 "Attribute Declarations".
- **XML container element**
  - An XML element that has one or more XML elements as its descendants.
- **XML document**
  - A well-formed document as defined in the W3C XML recommendation (W3C XML, definition of XML Document).
- **XML element**
  - A constituent of an XML document as defined in W3C XML.
- **XML element declaration**
  - A constituent of an XML Schema that constrains the structure and content of a specific XML element, in accordance with W3C XSD , section 3.3 "Element Declarations".
- **XML Schema**
  - A document that complies with the W3C XML Schema recommendation (W3C XSD).
- **XML Schema datatype**
  - A predefined set of permissible content within a section of an XML document as described in "W3C XSD Part 2: Datatypes".

1.5. Normative References

IETF BCP 47
Tags for Identifying Languages, September 2009, https://tools.ietf.org/rfc/bcp/bcp47.txt

IETF RFC 2119
Key words for use in RFCs to Indicate Requirement Levels, March 1997, https://www.ietf.org/rfc/rfc2119.txt

IETF RFC 3023
XML Media Types, January 2001, https://tools.ietf.org/rfc/rfc3023.txt

IETF RFC 3986
Uniform Resource Identifier (URI): Generic Syntax, January 2005, https://tools.ietf.org/rfc/rfc3986.txt

IETF RFC 6838
Media Type Specifications and Registration Procedures, January 2013, https://tools.ietf.org/rfc/rfc6838.txt

ISO 24622-1:2015
Language resource management - Component metadata infrastructure (CMDI) - Part 1: The component metadata model, ISO, 1 February 2015, http://www.iso.org/iso/catalogue_detail.htm?csnumber=37336

W3C XML
Extensible Markup Language (XML) 1.0 (Fifth Edition), T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler and F. Yergeau (eds.), W3C Recommendation 26 November 2008, http://www.w3.org/TR/2008/REC-xml-20081126/

W3C XML Namespaces
Namespaces in XML 1.0 (Third Edition), T. Bray, D. Hollander, A. Layman, R. Tobin and H. S. Thompson (eds.), W3C Recommendation 8 December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

W3C XSD
XML Schema Part 1: Structures (Second Edition), H. S. Thompson, D. Beech, M. Maloney and N. Mendelsohn (eds.), W3C Recommendation 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

W3C XSD Part 2: Datatypes
XML Schema Part 2: Datatypes (Second Edition), P.V. Biron and A. Malhotra (eds.), W3C Recommendation 02 May 2001, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

1.6. Typographic and XML Namespace conventions

The following typographic conventions for XML fragments will be used throughout this specification:

- `<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 namespaces denoted by the prefix *prefix*.
- `string`
  The literal *string* must be used either as element content or attribute value.
- `xs:type`
  The XML schema type with name *type*.

The following XML namespace names and prefixes are used throughout this specification. The column "Recommended Syntax" indicates which syntax variant **SHOULD** be used by the toolkit and other creators of CMDI related documents.

Prefix	Namespace Name	Comment	Recommen-ded Syntax
cmd	http://www.clarin.eu/cmd/1	CMDI instance (general/enve-lope)	prefixed
cmdp	http://www.clarin.eu/cmd/1/profiles/{profileId}	CMDI payload (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.

2. Structure of CMDI files

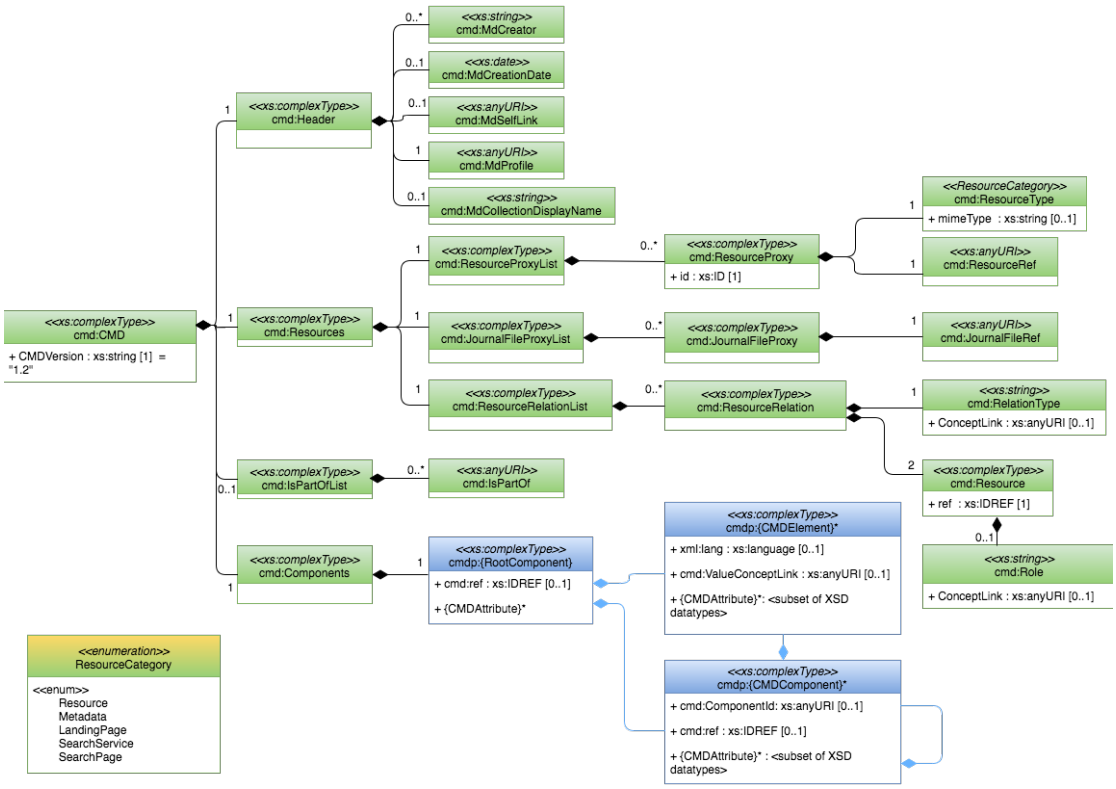

Figure 2 The structure of a CMDI file (CMD instance)

Colour scheme: Green boxes represent elements that are potentially present in all CMDI files (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 to be found in the tables below.

A CMDI file 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 CMDI files 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*).

2.1. The main structure

A CMDI file has the root element `<cmd:CMD>` with one attribute and 4 sub-elements that appear in mandatory order as described in the following table:

Name	Value type	Occurrences	Description
`<cmd:CMD>`	`xs:complexType`		The root element of the CMDI file.
`@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.

Name	Value type	Occurrences	Description
<cmd:Components>	xs:complexType	1	This element contains the profile specific section of the CMDI file. 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, which (profile specific) substructure exists in the 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 the "Typographic and XML Namespace conventions") **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 **MUST** be representative of the party that introduces the extension. Therefore, 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 sections.

Example 1 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/clarin.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>
```

2.2. The `<Header>` element

The header of a CMDI instance mainly contains administrative information about the metadata, that is metadata about the CMDI instance itself. The included elements **MUST** follow the structure and order described in this table:

Name	Value type	Occurrences	Description
`<cmd:Header>`	`xs:complexType`		Encapsulates core administrative data about the CMDI file.
`<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 PID (**RECOMMENDED**) or a URL.
`<cmd:MdProfile>`	`xs:anyURI`	1	The CMDI 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 2 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>
```

2.3. 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 **MUST** follow the structure and order described in this table:

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>).

2.3.1. 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 **MUST** follow the structure and order described in this table:

Name	Value type	Occurrences	Description
<cmd:ResourceProxyList>	xs:complexType		Contains a list of resource proxies (see below).
<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 CMDI file.
<cmd:ResourceType>	xs:string ("Resource", "Metadata", "LandingPage", "SearchService", "SearchPage"; see below for a description of 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 PID (**RECOMMENDED**) or a URL.

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 resources that is a web page that provides the original context of the described resource, e.g., a "deep link" into a repository system.

2.3.2. 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 **MUST** follow the structure and order described in this table:

Name	Value type	Occurrences	Description
`<cmd:JournalFileProxyList>`	`xs:complexType`		Contains a list of journal file proxies (see below).
`<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 PID (**RECOMMENDED**) or a URL.

Notes

- The actual content and layout of the journal file is outside the scope of this specification.

2.3.3. 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 **MUST** follow the structure and order described in this table:

If these parts are present they **MUST** appear in this order:

Name	Value type	Occurrences	Description
`<cmd:ResourceRelationList>`	`xs:complexType`		Contains a list of resource relations (see below).
`<cmd:ResourceRelation>`	`xs:complexType`	0 to unbounded	A representation of a relation between 2 resource proxies listed in

Name	Value type	Occurrences	Description
			<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), 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), indicating the semantics of <cmd:Role>.

Example 3 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 4 A minimally specified relation between resource files

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 5 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>
```

2.4. The IsPartOf List

`<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 **MUST** follow the structure and order described in this table:

Name	Value type	Occurrences	Description
`<cmd:IsPartOfList>`	`xs:complexType`		Contains a list of `<cmd:IsPartOf>` (see below).
`<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 PID (**RECOMMENDED**) or a URL.

Notes

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

Example 6 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>
```

2.5. The components

This section of the CMDI file 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 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. The next table describes the mandatory structure and order of this section as a function of the definition of a specific CMD Profile:

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.
`@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 component 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 a CMD element specification.
`<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`

Name	Value type	Occurrences	Description
			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 component 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 7 CMD instance payload

This example shows various (optional) aspects of the payload of an 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 components, elements and attributes defined in a CMD profile (`cmdp:*` and `@*`).

```
<cmd:Components>
     <cmdp:ToolService>
          …
          <cmdp:GeneralInfo>
               <cmdp:Name>Adelheid</cmdp:Name>
               <cmdp:Description>
                    <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>
```

3. The CMDI Component Specification Language (CCSL)

The CMDI Component Specification Language (CCSL) 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 profile specification) the complete payload structure of the CMD instance. It is also basis for the generation of the XML schema file that is used to validate a CMD instance (see section "Transformation of CCSL into a CMD profile schema definition" for details).

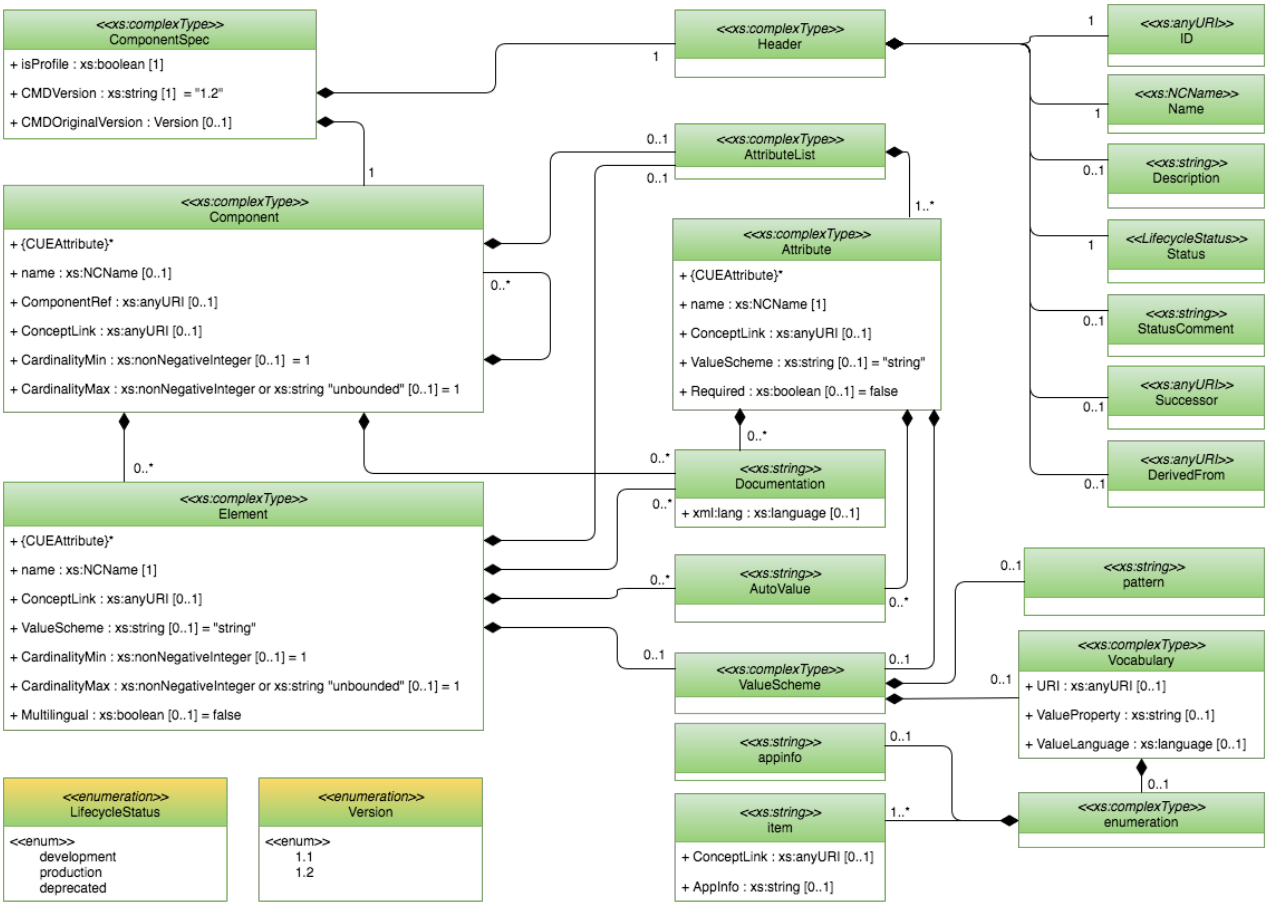

Figure 3 Structure of a CCSL document

A CCSL document **MUST** contain a CCSL header and the actual CMD component description. Its root element **MUST** contain an XML attribute `@isProfile` to indicate if the document specifies a CMD profile or a CMD component and it **MUST** 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.

The following table describes the root element and its direct descendants. The described structure and order **MUST** be followed.

Name	Valuetype	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 definition.
`@CMDVersion`	`xs:string` ("1.2")	1	CMDI version of this CMD specification.

Name	Valuetype	Occurrences	Description
@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 root component in case of a profile specification).

Example 8 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>
```

3.1. 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 **MUST** 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>.

The following table describes the header element and its direct descendants. The described structure and order **MUST** be followed.

Name	Valuetype	Occurrences	Description
<Header>	xs:complexType		Descriptive information about the component.
<ID>	xs:anyURI	1	ID of the component specification.
<Name>	xs:NCName	1	Name of the component.
<Description>	xs:string	0 or 1	Description of the component.
<Status>	xs:string ("development", "production", "deprecated"; see below for a 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 component, if

Name	Valuetype	Occurrences	Description
			available.
<DerivedFrom>	xs:anyURI	0 or 1	ID of component from which this component is derived, if available.

Status values

- **development**
  - The component specification is under construction, i.e., can undergo change at any moment, and therefore only to be used for testing purposes.
- **production**
  - The component specification is stable and will not be changed anymore, i.e., can be used for production-level metadata instances.
- **deprecated**
  - Usage of this component specification is discouraged, and usage of a successor component specification, if present, is encouraged.

Additional constraints

- A successor **SHOULD** only be present if the status of the CMD component is deprecated.

Example 9 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 10 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>
```

3.2. CMD component definition

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

The following table describes the element for defining CMD components and its direct descendants. The described structure and order **MUST** be followed.

Name	Valuetype	Occurrences	Description
<Component>	xs:complexType		Root element of every CMD component definition.
@name	xs:NCName	0 or 1	Name of the component.
@ComponentRef	xs:anyURI	0 or 1	Reference to an existing component 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 component has to occur (default: 1).
@CardinalityMax	xs:nonNegativeInteger or "unbounded"	0 or 1	Maximum number of times this component may occur (default: 1).
<Documentation>	xs:string	0 to unbounded	Documentation about the purpose of the 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 attributes specified by the component creator.
<Attribute>	xs:complexType	1 to unbounded	An additional attribute.
<Element>	xs:complexType	0 to unbounded	The elements of this component.
<Component>	xs:complexType	0 to unbounded	The components nested in this component.

Additional constraints

- A CMD component **MUST** have either a name or a reference to an existing 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 **MUST** both be 1.
- The value of the minimum cardinality **MUST** be lower or equal to the value of the maximum cardinality.
- For this CMD component, each documentation **MUST** have a unique @xml:lang value. And there **MUST** not be more than one documentation with an empty or missing @xml:lang.
- Within the attribute list each CMD attribute **MUST** have a unique name.
- The CMD elements and CMD components, which are direct descendants of this component, **MUST** all have different names.
- A CMD component **MUST NOT** to be a descendant of itself.

Example 11 CMD component definition

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>
```

3.3. CMD element definition

CMD elements are a template for storing atomic values governed by a value scheme in a CMD instance. The CCSL specification of an CMD element **MUST** contain the name of the element and **MAY** contain a concept link, the value schema, and information about the allowed cardinality of the element. Furthermore, it **MAY** be indicated if the element allows for values in more than one language, in which case an unlimited upper cardinality bound is implied. A CMD element **MUST** either have one of the standard XML schema datatypes assigned to it, or be constrained by using regular expressions or vocabularies. The latter can be specified by giving the complete list of allowed values or by stating the URI of an external vocabulary (for details see "Value schemes for elements and attributes"). If the 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.

The following table describes the element for defining CMD elements and its direct descendants. The described structure and order **MUST** be followed.

Name	Valuetype	Occurrences	Description
`<Element>`	`xs:complexType`		Root element of every CMD element definition.
`@name`	`xs:NCName`	1	Name of the 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 element has to occur (default: 1).
`@CardinalityMax`	`xs:nonNegativeInteger` or "unbounded"	0 or 1	Maximum number of times this element may occur (default: 1).

Name	Valuetype	Occurrences	Description
@Multilingual	xs:boolean	0 or 1	Indication that the element can have values in multiple languages (default: false).
<Documentation>	xs:string	0 to unbounded	Documentation about the purpose of the 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 attributes specified by the component creator.
<Attribute>	xs:complexType	1 to unbounded	An additional attribute.
<ValueScheme>	xs:complexType	0 or 1	Value scheme based on a regular expression or a specified vocabulary. See "Value schemes for elements and attributes" for details.
<AutoValue>	xs:string	0 to unbounded	Derivation rules for the element's content.

Additional constraints

- For the defined CMD element, each documentation **MUST** have a unique `@xml:lang` value. And there **MUST 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 **MUST** be lower or equal to the value of the maximum cardinality.
- Within the attribute list each CMD attribute **MUST** have a unique name.

Notes

- If multilingual has the value `true` and `@ValueScheme` has the value `string`, the value of `@CardinalityMax` **MUST** be ignored and defaults to unbounded.
- If `@ValueScheme` has not the value `string` the value of multilingual **MUST** be ignored.
- If the CMD element has a `<ValueScheme>` the data type defaults to `string`.

Example 12 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 13 CMD element definition with auto value

This example shows the definition of a CMD element with an (informative) auto value derivation rule, i.e., instantiate 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>
```

3.4. CMD attribute definition

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

The following table describes the element for defining CMD attributes and its direct descendants. The described structure and order **MUST** be followed.

Name	Valuetype	Occurrences	Description
`<Attribute>`	`xs:complexType`		Root element of every CMD attribute definition.
`@name`	`xs:NCName`	1	Name of the 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 attribute is required (default: false).
`<Documentation>`	`xs:string`	0 to unbounded	Documentation about the purpose of the 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 vocabulary. See “Value schemes for elements and attributes” for details.
`<AutoValue>`	`xs:string`	0 to unbounded	Derivation rules for the attribute’s content.

Additional constraints

- For the defined CMD attribute, each documentation **MUST** have a unique `@xml:lang` value. And there **MUST NOT** be more than one documentation with an empty or missing `@xml:lang`.
- A CMD attribute **SHOULD** have either a `@ValueScheme` or a `<ValueScheme>`.

Notes

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

Example 14 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>
```

3.5. Value schemes for elements and attributes

Apart from standard XML schema datatypes the content of a CMD element or 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 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 vocabulary via a URI 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 URI, 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 vocabulary.

The order and structure described in the following table **MUST** be followed when specifying value schemes:

Name	Valuetype	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 CMD vocabulary.
`@URI`	`xs:anyURI`	0 or 1	URI of an external vocabulary.
`@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.

Name	Valuetype	Occurrences	Description
<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 **MUST** be unique.
- A <ValueScheme> must have either a <pattern>, or a <Vocabulary> with a non-empty <enumeration>, or a @URI.

Notes

- A vocabulary with a non-empty enumeration of permissible values constitutes a closed vocabulary. Using @URI, an external vocabulary provided by a vocabulary service, e.g., the CLARIN vocabulary service CLAVAS, 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 15 Value scheme with enumeration (closed vocabulary)

This example shows a value scheme with a reference to an external 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 16 Value scheme without enumeration but with external vocabulary (open vocabulary)

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

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

Example 17 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>
```

3.6. Cue attributes

CMDI profiles provide the blueprint for a logical structuring of metadata instances. However, they provide very little explicit information about how the information contained or to be entered in CMDI instances should be dealt with in applications that process or generate metadata documents. CMDI 1.2 therefore allows for the augmentation of components, elements and attributes in profile definitions 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. 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 CMDI instances.

For this purpose, all CMD attribute, element, and component specifications **MAY** contain additional attributes in the cue namespace. These **MAY** be used to give information about how the 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 18 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 19 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" />
     <Component name="Address"
       CardinalityMin ="0" CardinalityMax ="1"
       cue:DisplayInline="true ">
         <Element name="Street" />
         <Element name="Place" />
         <Element name="Country" />
     </Component>
</Component>
```

4. Transformation of CCSL into a CMD profile schema definition

A CMD instance document that is serialised as XML according to this specification **SHOULD** contain a reference to the location of a CMD profile schema. The infrastructure **MUST** provide a mechanism to derive such a schema for any specific CMD profile on basis of its definition and that of the CMD components that it references. This section specifies how different aspects of a CMD specification should be transformed into elements of a schema definition. The primary schema language targeted is XML Schema, although the infrastructure **MAY** provide support for other schema languages, such as Relax NG (ISO/IEC 19757-2:2003). A CMD profile schema **MUST** be derived from a CMD profile specification.

4.1. General properties of the CMD profile schema definition

A CMD profile schema **MUST** allow for the evaluation of a CMD instance on all levels of description defined in one specific CMD profile. The schema **MUST** require the presence of a CMD instance envelope as described in section "Structure of CMDI files". The value of the `<cmd:MdProfile>` header item in the CMD instance envelope **SHOULD** only be valid if it is equal to the profile id as specified in the associated CMD profile.

The CMD profile schema **SHOULD** include, as a matter of annotation, a copy of (a subset of) the information contained in the `Header` section of the CMD profile from which it is derived.

The transformation **MAY** make use of component references in the CMD component definition to derive (complex) types that can be reused throughout the schema definition.

The schema **MUST** declare a profile specific payload namespace in addition to the fixed, global namespaces that are used (in particular `cmd` and `cue`). This namespace, with **RECOMMENDED** prefix `cmdp`, **MUST** have the following format: `http://www.clarin.eu/cmd/1/profiles/{profileId}`, where `{profileId}` refers to the identifier of the profile from which the schema is derived in a component registry. All XML elements and XML attributes derived from CMD components, CMD elements **MUST** be qualified and declared in this namespace. XML attributes derived from CMD attributes follow the convention that unprefixed attributes belong to their elements, which do belong to the profile specific payload namespace.

4.2. Interpretation of CMD component definitions in the CCSL

CMD Components which are represented as `<Component>` XML elements in the CCSL, **MUST** be realised as XML element declarations with the following property mapping:

Property	XML schema attribute	Derived from	Use
Name of the XML element	`@name`	`@name`	**REQUIRED**
Minimal number of occurrences	`@minOccurs`	`@CardinalityMin`, or '1' if XML attribute not present	**REQUIRED** [2]
Maximal number of occurrences	`@maxOccurs`	`@CardinalityMax`, or '1' if XML	**REQUIRED** [2]

[2] The implementation may make use of default evaluation of the schema language if it matches these requirements, as is the case with XML Schema, and therefore omit explicit declaration of these properties.

Property	XML schema attribute	Derived from	Use
		attribute not present	
Concept link	`@cmd:ConceptLink`	`@ConceptLink`	**OPTIONAL**
Component id	`@cmd:ComponentId`	`@ComponentId`	**OPTIONAL**

An optional XML Attribute `@cmd:ref` of type *xs:IDREF* **MUST** be allowed on the XML container element derived from any CMD component.

`<Documentation>` XML elements contained in CMD Components **SHOULD** be transformed into documentation elements embedded in the XML element declaration. In these, the content language information contained in the `@xml:lang` XML attribute **SHOULD** be preserved.

XML attributes of CMD Components in the `cue` namespace **SHOULD** be copied into the XML element declaration, in which case the XML attribute name, namespace and value **SHOULD** be preserved.

4.2.1. Document structure prescribed by the schema

The first CMD component defined in the CMD profile (the "root component") **MUST** be mapped as the mandatory, only child element of the `<Components>` XML element of the CMD instance envelope. CMD components that are defined as direct descendants of another CMD component **MUST** be mapped as direct descendants of the XML element declaration to which it is transformed. XML components at the CMD component level in the metadata instance **MUST** be required to be included in the same order as defined in the CMD specification, the first of the resulting XML elements appearing after the last XML element derived from a CMD element at the same level, if present. These descendant CMD Components **MUST** also be mapped to XML element declarations recursively as described in this specification.

CMD elements **MUST** be mapped as direct descendants of the XML element declaration derived from the CMD component of which they are direct descendants, and **MUST** be required to be included in the same order as defined in the CMD specification.

CMD attributes that are defined in the CCSL within `<Attribute>` XML elements within an `<AttributeList>` XML element that is a direct descendant of a CMD Component **MUST** be mapped to XML attribute definitions on the XML container element to which this CMD Component is transformed.

4.3. Interpretation of CMD element definitions in the CCSL

CMD elements, represented as `<Element>` XML elements in the CCSL, **MUST** be realised as XML element declarations with the following property mapping:

Property	XML schema attribute	Derived from	Use
Name of the XML element	`@name`	`@name`	**REQUIRED**
Minimal number of occurrences	`@minOccurs`	`@CardinalityMin` **unless** `@Multilingual` is true, in which case **MUST** be 'unbounded', or '1' if neither XML attribute is present	**REQUIRED** [3]
Maximal	`@maxOccurs`	`@CardinalityMax`, or '1' if XML attribute not	**REQUIRED** [3]

[3] The implementation may make use of default evaluation of the schema language if it matches these requirements, as is the case with XML Schema, and therefore omit explicit declaration of these properties.

Property	XML schema attribute	Derived from	Use
number of occurrences		present	
Type of the XML element	`@type`	See section "Content model for CMD elements and CMD attributes in the schema definition"	
Concept link	`@cmd:ConceptLink`	`@ConceptLink`	**OPTIONAL**
Auto value instruction	`@cmd:AutoValue`	`@AutoValue`	**OPTIONAL**

`<Documentation>` XML elements contained in CMD elements **SHOULD** be transformed into documentation elements embedded in the XML element declaration. In these, the content language information contained in the `@xml:lang` XML attribute **SHOULD** be preserved.

XML attributes of CMD elements in the 'cue' namespace **SHOULD** be copied into the XML element declaration, in which case the XML attribute name, namespace and value **SHOULD** be preserved.

An optional XML attribute `@cmd:ValueConceptLink` of type *xs:anyURI* **MUST** be allowed on the XML element derived from a CMD element that has a vocabulary with XML attribute `@URI` defined (see section "Content model for CMD elements and CMD attributes in the schema definition").

The derivation of a content model for the XML element declaration on basis of a CMD element is described below.

4.4. Interpretation of CMD attribute definitions in the CCSL

CMD attributes, represented as `<Attribute>` XML elements in the CCSL, **MUST** be realised as XML attribute declarations with the following property mapping:

Property	XML schema attribute	Derived from	Use
Name of the XML element	`@name`	`@name`	**REQUIRED**
Use of the XML attribute	`@use`	'required' if and only if `@Required` is present and equals true, otherwise 'optional'	**REQUIRED** [4]
Type of the XML attribute	`@type`	See section "Content model for CMD elements and CMD attributes in the schema definition"	
Concept link	`@cmd:ConceptLink`	`@ConceptLink`	**OPTIONAL**
Auto value instruction	`@cmd:AutoValue`	`@AutoValue`	**OPTIONAL**

`<Documentation>` XML elements contained in CMD attributes **SHOULD** be transformed into documentation elements embedded in the XML attribute declaration. In these, the content language information contained in the `@xml:lang` XML attribute **SHOULD** be preserved.

XML attributes of CMD attributes in the `cue` namespace **SHOULD** be copied into the XML attribute declaration, in which case the XML attribute name, namespace and value **SHOULD** be preserved.

The derivation of a content model for the XML attribute declaration on basis of a CMD attribute is described below.

[4] The implementation may make use of default evaluation of the schema language if it matches these requirements, as is the case with XML Schema, and therefore omit explicit declaration of these properties.

4.5. Content model for CMD elements and CMD attributes in the schema definition

If a CMD element or CMD attribute in the CCSL has a `@ValueScheme` XML attribute, its value **MUST** be interpreted as the name of the XML Schema datatype (declared in the `@type` attribute of the XML element or attribute declaration in XML Schema) that defines the allowed value range of the XML element/attribute derived from the CMD element/attribute.

Otherwise, if a CMD element or CMD attribute in the CCSL has a descendant XML element `<ValueScheme>` that contains an XML element `<pattern>`, then its text value **MUST** be interpreted as the XML Schema Regular Expressions that defines the allowed value range of the XML element/attribute derived from this CMD element/attribute.

Otherwise, if a CMD element or CMD attribute in the CCSL has a descendant XML element `<ValueScheme>` that contains an XML element `<Vocabulary>`:

- The XML attribute `@URI` of the XML element `<Vocabulary>`, if present, **SHOULD** be transformed into an attribute `cmd:Vocabulary` of the same value on the XML element or attribute declaration in the schema. The XML element declaration should always allow a `@cmd:ValueConceptLink` to retain a link to a specific vocabulary entry.
- The XML attributes `@ValueProperty` and `@ValueLanguage` of the XML element `<Vocabulary>` **SHOULD** be transformed into XML attributes in the `cmd` namespace on the XML element declaration in the case of a CMD element or XML attribute declaration in the case of a CMD attribute.
- The XML elements `<item>` that are descendants of `<enumeration>` contained in `<Vocabulary>` **MUST** be transformed into an enumeration based restriction with values taken from the text content of the `<item>` XML elements. Each enumeration item in the schema **SHOULD** be annotated: the value from the XML attribute `@ConceptLink` by means of an XML attribute `@cmd:ConceptLink`, and the value of the XML attribute `@AppInfo` by means of an attribute `@cmd:label`.

Notes

- `@cmd:Vocabulary`, `@cmd:ValueProperty`, `@cmd:ValueLanguage` and `@cmd:ConceptLink` **MAY** appear as attributes of XML attribute declarations and XML element declarations in the schema document for a CMDI profile, and **MUST NOT** appear in the CMDI instance.

Bibliography

Broeder *et al*, 2010

D. Broeder, M. Kemps-Snijders, D. Van Uytvanck, M. Windhouwer, P. Withers, P. Wittenburg, C. Zinn. A Data Category Registry- and Component-based Metadata Framework. In the *Proceedings of the Seventh International Conference on Language Resources and Evaluation* (LREC 2010), European Language Resources Association (ELRA), Malta, May 19-21, 2010.

Broeder *et al*, 2012

D. Broeder, M. Windhouwer, D. van Uytvanck, T. Goosen, T. Trippel. CMDI: a Component Metadata Infrastructure. In the *Proceedings of the Metadata 2012 Workshop on Describing Language Resources with Metadata: Towards Flexibility and Interoperability in the Documentation of Language Resources*. At LREC 2012, Istanbul, Turkey, May 22, 2012.

CLARIN Component Registry

https://www.clarin.eu/componentregistry

CLARIN Concept Registry

http://www.clarin.eu/ccr/

CLARIN CE 2014-0318

CMDI 1.2 changes - executive summary, Technical Report CD-2014-0318, April 2014,
http://www.clarin.eu/content/cmdi-12-changes-executive-summary

CLARIN ERIC

http://www.clarin.eu/

CLAVAS

https://openskos.meertens.knaw.nl/clavas/

CMDI toolkit

hdl:11372/CMDI-0001

Durco & Windhouwer, 2013

M. Durco, M. Windhouwer. Semantic Mapping in CLARIN Component Metadata. In E. Garoufallou and J. Greenberg (eds.), *Metadata and Semantics Research* (MTSR 2013), CCIS Vol. 390, Springer, Thessaloniki, Greece, November 20-22, 2013.

ISO 1087-1:2000

Terminology work -- Vocabulary -- Part 1: Theory and application, ISO, 15 October 2000,
http://www.iso.org/iso/catalogue_detail.htm?csnumber=20057

ISO 12620:2009

Terminology and other language and content resources -- Specification of data categories and management of a Data Category Registry for language resources, ISO, 15 December 2009,
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=37243
(Withdrawn)

ISO/IEC 19757-2:2003

Information technology -- Document Schema Definition Language (DSDL) -- Part 2: Regular-grammar-based validation -- RELAX NG, December 1, 2003, http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=37605

Merriam-Webster Dictionary and Thesaurus

http://www.merriam-webster.com/dictionary

Van Uytvanck *et al*, 2012

D. Van Uytvanck, H. Stehouwer, L. Lampen. Semantic metadata mapping in practice: the Virtual Language Observatory. In the *Proceedings of the Eighth International Conference on Language Resources and Evaluation* (LREC 2012), European Language Resources Association (ELRA), Istanbul, Turkey, May 23-25, 2012.

Table of Contents

Figures

Examples