The Component MetaData Infrastructure (CMDI) – the Non-Component Header

Thorsten Trippel

2012-01, last processed January 10, 2012

Contents

1 Introduction

The *Component MetaData Infrastructure* (CMDI, pronounced /sImDi:/) has gained an increased pace for describing various types of language resources. In the meantime the component model is used by European Projects such as CLARIN, MetaShare and FlareNet, hosted at various working groups and institutions.

The core idea of the CMDI model is that various types of language resources require different sets of metadata; a lexical resource – usually described in terms of number of lexical keywords/lexemes, definitions, etc. – can hardly be described adequately by the same data categories as a corpus, using the number of words, language, type-token ratio, genre, etc. This is even more true for other resources, for example, psycho-linguistic experiments, field work recordings, language technology corpora, phonetic trancriptions, etc. Though some portions of descriptive patterns may be similar, others are not. Defining a core set of data categories has proven to be controversial and hard, if not impossible throughout the (sub-)disciplines.

The CMDI model allows the definition of metadata sets for each type of resource according to the requirements for an adequate description, while at the same time allowing the reuse of structures that are used for the description of other resources. Each of the reusable parts of the model is bound into a component; components contain either further components or lists of data categories. The data categories as such are defined elsewhere in a data category repository such as ISOcat (`http://www.isocat.org`, the reference implementation of ISO 12620:2009) and refer to their definitions by persistent identifiers. A component which is used to describe a class of resources and which is not embedded into other components is called a profile.

The structure and options of the component definition is described in detail elsewhere. This file is supposed to document and describe the non-component header of CMDI instances.

The following is a snippet of a header for a CMDI file before the components section.

```
<?xml version="1.0" encoding="UTF-8"?>
<CMD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="http://www.clarin.eu/cmd/"
    CMDVersion="1.1"
    xsi:schemaLocation="http://www.clarin.eu/cmd/
    http://catalog.clarin.eu/ds/ComponentRegistry/rest/registry/profiles/clarin.eu:cr1:p_1290431694580/xsd">

    <Header>
        <MdCreator>Reinhild Barkey</MdCreator>
        <MdCreationDate>2011-03-31</MdCreationDate>
        <MdSelfLink>http://hdl.handle.net/XXXX/XXXXXXXXXXXX</MdSelfLink>
        <MdProfile>
        http://catalog.clarin.eu/ds/ComponentRegistry/rest/registry/profiles/clarin.eu:cr1:p_1290431694580/xsd
        </MdProfile>
        <MdCollectionDisplayName>Tbinger Seminar fr Sprachwissenschaft</MdCollectionDisplayName>
    </Header>

    <Resources>
        <ResourceProxyList>
            <ResourceProxy id="resprox1">
                <ResourceType mimetype="application/xml">Resource</ResourceType>
                <ResourceRef>http://hdl.handle.net/TheResourcePID</ResourceRef>
            </ResourceProxy>
        </ResourceProxyList>
        <JournalFileProxyList>
            <JournalFileProxy>
                <JournalFileRef>http://hdl.handle.net/ThePIDtoPROVENANCEfile</JournalFileRef>
            </JournalFileProxy>
        </JournalFileProxyList>
        <ResourceRelationList>
            <ResourceRelation>
                <RelationType>Somehow they combine</RelationType>
                <Res1 ref="resprox1"/>
                <Res2 ref="resprox1"/>
            </ResourceRelation>
        </ResourceRelationList>
        <IsPartOfList>
            <IsPartOf>http://hdl.handle.net/SomeOtherBiggerResourceThisIsPartOf</IsPartOf>
            <IsPartOf>http://hdl.handle.net/SomeOtherEvenBiggerResourceThisIsPartOf</IsPartOf>
        </IsPartOfList>
    </Resources>

    <Components>
    ...
    </Components>
</CMD>
```

The root element of CMDI metadata files is the CMD namespace currently bound to the clarin.eu/cmd-namespace. The attribute CMDVersion provides information for later developments of CMDI. CMDI is intended to be upward-compatible, that is, profiles that were valid according to an earlier CMDI version will be valid according to new versions.

Note that CMDI profiles provide a namespace. Hence, to be valid, the schema location, a pair of the namespace and the URI for the schema, need to be provided. In this example, the CMDI namespace is defined as the default namespace.

The CMDI file is broadly divided into three parts: the header, resources and components. This document focuses on the header and resource part.

2 Terms and Definitions

3 The Header

The header-element is a container element intended to provide information on the metadata file as such, not the resource that is described by the metadata file. To make this more explicit and human-readable, the data categories contained in the header are prefixed by `Md` for *Metadata*. The following elements are part of the header, all of these elements are optional:

MdCreator (optional): Name of the person who created the metadata file. This is defined as a string.

MdCreationDate (optional): Date of the creation of the metadata file. This is defined as the data type `date`, i.e. the date is specified in the form yyyy-mm-dd (four digits for the year, followed by a dash, followed by two digits for the month, followed by a dash, followed by two digits for the day of the month).

MdSelfLink (optional): Persistent identifier for the metadata file (see [**?**]) in the form of a URI.

MdProfile (optional, redundant): URI of the profile used to create this metadata file. This information is redundant as it is also part of the value of the schemalocation attribute of the root element. However, this is left in for compatibility, its use is deprecated.

MdCollectionDisplayName (optional): The name for a collection as it is supposed to be displayed by an application. This element is used because metadata is often shared and institutions display the names of the collections in applications.

It is always recommended to fill in all possible fields here. The idea for these fields is to structure the data and make information available, providing some background for the users of the metadata.

Potential problems, intentionally left vague, are how to deal with changed metadata files: should the MdCreator and MdCreationDate be adjusted? If yes, how persistent is the MdSelfLink? As the metadata is created during the archiving state of a resource, potential updates are currently not dealt with.

Should we have an MdMaintained (optional, 0..unbound) with a triple MdMaintainer, MdMaintenanceDate, MdMaintenanceRemark (optional)?

4 The Resources Section

The resources section in a metadata file lists all information relevant for the individual resource, but does not describe the resource as such. The description is part of the components, the resource section provides the location of the resource or its parts if it consists of more than one, provenance information on the resource, information on the relation between the parts of the resource, if applicable, and information on a greater body the resource is part of, also if applicable.

4.1 The Resource Proxy List

The resource proxy list defines metadata file internal placeholders, called proxies, for each part of a resource. For example, if a resource consists of one specific file, this file is referenced in the `ResourceRef` element, which holds the PID of this file, in the form of a URI. As resources can be composed of other resources, which are identified by their metadata, the `ResourceType` element specifies if the PID refers to metadata (another metadata file) or a resource, such as a binary file or data. To further specify the type `ResourceType` it takes `mimetype` as an attribute, with the value specifying the mimetype of the referenced resource. Providing the mimetype is optional.

We do not use the mimetype attribute yet…

Resources can consist of more than one data streams or files, hence the `ResourceProxyList` may contain more than one `ResourceProxy`. To be able to refer to each of these parts individually, each `ResourceProxy` receives an `id` attribute for internal reference within the metadata file.

4.2 The Journal File Proxy List

For many resources that are developed over a longer period of time, changes and updates are frequent. Provenance data is not part of the CMDI model, but it is possible to store provenance data outside of the metadata file in sensible forms. Provenance metadata is refered to as `JournalFile` in CMDI documents. The `JournalFileProxyList` contains the list of all JournalFiles for a resource, the `JournalFileRef` holds the URI as a reference to the JournalFile containing the provenance data.

Mail from Dieter van Uytvanck, 28 Sep 2010 17:32:51 +0200, see also http://www.clarin.eu/cmd/e md-instance.cmdi

4.3 The Resource Relation List

Resource files do not exist independent of each other if a resource consists of more than one file. For example, audio files and transcriptions are related to each other. The ResourceProxyList only lists these files, the `ResourceRelationList` makes the relation between pairs of files explicit. For this purpose the `ResourceRelation` contains a triple of elements defining a directed relation between a Resource 1, which is referenced by a `ref`-pointer to an id from the `ResourceProxy`, and a Resource 2 respectively. The relation between the two is given as a string in the `RelationType` element

Res1 and Res2 are awkward names, this is a directed relation, so why not source and target? Should the Relation type be a closed vocabulary? What are potential values?

A good example would be helpful here.

4.4 The Is-Part-of List

Resources that are defined in bundles are listed under `ResourceProxy`. The individual parts can be seen as independent resources as well, such as a subcorpus that can also be distributed on its own. To point out that a resource is part of a larger unit or created as part of a larger unit, the `IsPartOfList` is introduced referring to one or more larger units by giving the PID of the larger units with the `IsPartOf` element.

5 Summary

The non-component part of CMDI-files is rather static, referring to the individual resource files, the relation between them, the relation to provenance data and the creation of the metadata. The creation and options of this part of the CMDI documents is essential for the archiving process and for referencing to the archive from within CMDI files distributed to other institutions.