CMDI Explorer

Denis Arnold
Leibniz-Institut für Deutsche Sprache
Mannheim, Germany
arnold@ids-mannheim.de

Ben Campbell
Eberhard Karls Universität
Tübingen, Germany
ben.campbell@uni-tuebingen.de

Thomas Eckart
Universität Leipzig
Leipzig, Germany
teckart@informatik.uni-leipzig.de

Bernhard Fisseni
Leibniz-Institut für Deutsche Sprache
Mannheim, Germany
fisseni@ids-mannheim.de

Thorsten Trippel
Eberhard Karls Universität
Tübingen, Germany
thorsten.trippel@uni-tuebingen.de

Claus Zinn
Eberhard Karls Universität
Tübingen, Germany
claus.zinn@uni-tuebingen.de

Abstract

We present CMDI Explorer, a tool that empowers users to easily explore the contents of complex CMDI records and to process selected parts of them with little effort. The tool allows users, for instance, to analyse virtual collections represented by CMDI records, and to send collection items to other CLARIN services such as the Switchboard for subsequent processing. CMDI Explorer hence adds functionality that many users felt was lacking from the CLARIN tool space.

1 Motivation

A scientific resource often comprises many different parts. A proper description of such a resource with metadata according to CLARIN standards will yield a rich metadata record that lists each of the significant parts with detailed information. Consider the following example. A doctoral project that investigates the acquisition of language in small children might involve a number of experiments where babies are exposed to various visual and auditive stimuli, where eye tracking and other sensor data is used to observe their reactions, and where various Python and R scripts are employed to manipulate and analyse such data automatically. To describe such study, the doctoral candidate will attach, for instance, the media type to each stimulus, describe the nature of the sensor data, or refer to each of the processing scripts and the order they need to be executed. Rich metadata makes it easier for others to follow-up on research, say, when trying to reproduce research results, or to build a follow-up project on existing work, say by conducting a meta-study where the work of our doctoral student is taken to be one of many similar studies. A proper description of the meta-study, in turn, will yield a yet more complex metadata record, now describing the meta-study and how it has used the individual studies in the amalgamation.

Reading and processing complex metadata is no trivial matter. In this paper, we propose a tool that supports researchers in working with highly structured metadata and its associated research data.

2 Background

With CMDI being the de-facto standard for metadata in the CLARIN world, our community has built a good range of tools that process, in some way or another, CMDI metadata (Broeder et al., 2012).

The CLARIN *Virtual Language Observatory* (VLO; https://vlo.clarin.eu) gives researchers access to hundreds of thousands of language-related resources via their metadata descriptions (Van Uytvanck et al., 2012). At regular intervals, its back-end engine harvests CMDI-based metadata from many different metadata providers. It needs to analyse these CMDI records, which adhere to many

Denis Arnold, Ben Campbell, Thomas Eckart, Bernhard Fisseni, Thorsten Trippel and Claus Zinn 2021. CMDI Explorer. *Selected papers from the CLARIN Annual Conference 2020*. Linköping Electronic Conference Proceedings 180: 180 8–15.

different metadata profiles, for content to correctly fill the various facets (*e.g.*, *language*, *resource type*, *modality*, *format*) that users will use to conduct faceted search in the VLO front-end. Once users have found a resource of interest, its individual metadata records are presented in a tabbed user interface, which includes a listing of its constituent parts via a simple hierarchical representation (*cf.* Fig. 1). Many descriptions in the VLO, however, are highly structured CMDI records. Navigating such metadata in such a tabbed environment, where a tabular entry points to a complex structure, involves following persistent identifiers attached to substructures *manually*. Hence, it can take some time to identify a sub-tree's leaves where, say, the auditive stimuli of a study of interest can be found.

Figure 1: A simple structured CMDI record in the VLO.

The CLARIN *Virtual Collection Registry* (VCR; `https://collections.clarin.eu`) enables scientists to assemble resources of interest into a virtual collection via persistent identifiers (PIDs) that refer to their individual metadata (Elbers, 2017). With virtual collections themselves being referred to by PIDs, scientists can easily create collections that have references to simple elements (such as single publications), and to complex elements (such as other virtual collections). It is hence easily possible to construct highly-structured virtual collections. The VCR is the primary entry point for scholars to create new virtual collections, share them with others, and browse through shared collection records. The portal offers some basic search functionality; it also provides a lean presentation of associated resources (*cf.* Fig. 2), which however, fails short at mirroring the potentially hierarchical structure of a collection.

The CLARIN *Language Resource Switchboard* (`https://switchboard.clarin.eu`) makes it easy for users to identify and invoke software tools that can process a language-related resource in one way or another (Zinn, 2018). The Switchboard's tool space, however, is geared towards, so to speak, the leaves of CMDI record trees, the actual scientific resources such as their text or audio files. The Switchboard cannot, for instance, handle a CMDI file that, say, describes a *plain* set of text files, which users will want to batch-process one by one with the same chosen tool.

Both the VLO and the VCR stop analysing CMDI files when it comes to resolving hierarchical structures marked by `ResourceProxyLists` of type `Metadata`. Unaware of the deep hierarchical structures behind a CMDI file, both VCR and VLO fail to offer users the crucial capability to navigate through them. Hence, users cannot easily explore those structures, select parts of them, say, to download them for off-line processing, or to send them to the Switchboard for a further analysis.

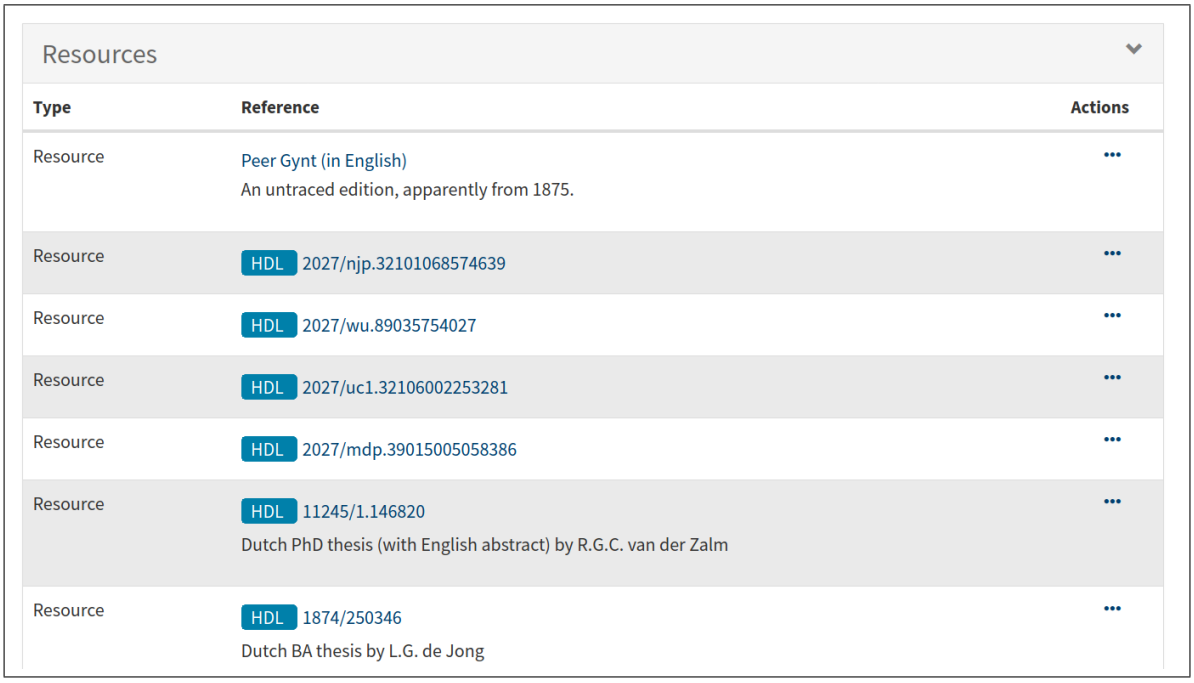

Figure 2: A simple structured CMDI record in the VCR.

Other tools face similar deficits. The CLARIN community offers a number of converters from CMDI to bibliographic metadata standards such as Dublin Core or MARC 21 (Zinn et al., 2016). These converters either show similar shortcomings when it comes to processing highly structured CMDI files, or are tailored to specific CMDI profiles where hence structural complexities are known in advance.

There are a couple of tools that go the extra mile of processing highly structured CMDI files: *SMC Browser* (Ďurčo, 2013), see `https://clarin.oeaw.ac.at/smc-browser/index.html`) and *Curation Module* (King et al., 2016; Ostojic et al., 2017), see `https://curate.acdh.oeaw.ac.at/`) Both applications focus on a computer-assisted quality assessment of CMDI files. The Curation Module aims at providing statistical information relevant to the evaluation of the usability of a metadata instance. This includes link resolution checks and an evaluation of a record's adequacy for faceted search in the VLO. The focus on quality assurance makes these tools mostly suited for metadata creators and publishers, but not for the general user.

In sum, the VLO, VCR, and the Switchboard would profit from software that crosses navigational boundaries. The CLARIN *CMDI Explorer* aims at complementing (and supporting) the CLARIN tool space with the much-needed functionality of handling complex CMDI metadata. It provides a simple way of accessing all data files associated with a resource described by a CMDI metadata file. For this, it accesses the `ResourceProxyList` of a (possibly recursive) CMDI file and provides a navigable tree overview of all files associated with a collection. Each individual file can then either be downloaded directly, or it can be send to the Switchboard for further processing. CMDI Explorer also allows users to download all data files (or a selection thereof), depending on license restriction, for off-line usage.

3 CMDI Explorer

Consider the fragment of a CMDI file given in Fig. 3. The fragment has been taken from a CMDI file that describes data associated with a PhD dissertation[1] (Dima, 2019), and which will be our running example for the remaining part of this paper. Let us consider the nine `ResourceProxy` children in the fragment in more detail. There are eight resources of type `Metadata` and one resource of type `LandingPage`. The first child contains a self-reference to the CMDI file itself while the second child

[1] See `http://hdl.handle.net/11022/0000-0007-CFE2-1`.

```
<ResourceProxyList>
  <ResourceProxy id="metadata0000-0007-CFE2-1">
    <ResourceType mimetype="application/x-cmdi+xml">Metadata</ResourceType>
    <ResourceRef>https://hdl.handle.net/11022/0000-0007-CFE2-1@CMDI.xml</ResourceRef>
  </ResourceProxy>
  <ResourceProxy id="landingpage0000-0007-CFE2-1">
    <ResourceType>LandingPage</ResourceType>
    <ResourceRef>https://hdl.handle.net/11022/0000-0007-CFE2-1</ResourceRef>
  </ResourceProxy>
  <ResourceProxy id="Res10000-0007-CFE2-1">
    <ResourceType mimetype="application/x-cmdi+xml">Metadata</ResourceType>
    <ResourceRef>https://hdl.handle.net/11022/0000-0007-CFD8-D</ResourceRef>
  </ResourceProxy>
  <ResourceProxy id="Res20000-0007-CFE2-1">
    <ResourceType mimetype="application/x-cmdi+xml">Metadata</ResourceType>
    <ResourceRef>https://hdl.handle.net/11022/0000-0007-CFD9-C</ResourceRef>
  </ResourceProxy>
  <ResourceProxy id="Res30000-0007-CFE2-1">
    <ResourceType mimetype="application/x-cmdi+xml">Metadata</ResourceType>
    <ResourceRef>https://hdl.handle.net/11022/0000-0007-CFD7-E</ResourceRef>
  </ResourceProxy>
  <ResourceProxy id="Res40000-0007-CFE2-1">
    <ResourceType mimetype="application/x-cmdi+xml">Metadata</ResourceType>
    <ResourceRef>https://hdl.handle.net/11022/0000-0007-CFDC-9</ResourceRef>
  </ResourceProxy>
  <ResourceProxy id="Res50000-0007-CFE2-1">
    <ResourceType mimetype="application/x-cmdi+xml">Metadata</ResourceType>
    <ResourceRef>https://hdl.handle.net/11022/0000-0007-CFDB-A</ResourceRef>
  </ResourceProxy>
  <ResourceProxy id="Res60000-0007-CFE2-1">
    <ResourceType mimetype="application/x-cmdi+xml">Metadata</ResourceType>
    <ResourceRef>https://hdl.handle.net/11022/0000-0007-CFDA-B</ResourceRef>
  </ResourceProxy>
  <ResourceProxy id="Res70000-0007-CFE2-1">
    <ResourceType mimetype="application/x-cmdi+xml">Metadata</ResourceType>
    <ResourceRef>https://hdl.handle.net/11022/0000-0007-CFD6-F</ResourceRef>
  </ResourceProxy>
</ResourceProxyList>
```

Figure 3: A CMDI fragment, where references to resources are listed.

informs readers that the metadata description of Dima's PhD has a landing page. Readers can copy the respective handle and paste it in their browser to view the landing page's content. In our case, the landing page points to the TALAR research data repository. Here, users can enjoy to read the CMDI file in a more user-friendly manner. The remaining seven `ResourceProxy` children refer to the actual research data sets used in Dima's thesis. Users need to navigate to another fragment of Dima's CMDI file (`ResourceProxyListInfo`) to get more, albeit sparse, information about these data objects, see Fig. 4. To obtain more detailed information about each dataset, users will need to navigate back to the first fragment (see Fig. 3) to take the handle given in the `ResourceProxyList` and resolve it in the browser (which takes care of multiple redirects). In our running example, each handle points to the TALAR repository that holds the corresponding dataset. In the web interface of TALAR showing the

```
<cmdp:ResourceProxyListInfo>
    <cmdp:ResourceProxyInfo xmlns:ns1="http://www.clarin.eu/cmd/1" ns1:ref="Res10000-0007-CFE2-1">
        <cmdp:ResProxItemName>German Noun-Noun Compounds Dataset for Compositionality Tests</cmdp:ResProxItemName>
        <cmdp:ResProxFileName>deu-comp-nn-only</cmdp:ResProxFileName>
    </cmdp:ResourceProxyInfo>
    <cmdp:ResourceProxyInfo xmlns:ns1="http://www.clarin.eu/cmd/1" ns1:ref="Res20000-0007-CFE2-1"> [3 lines]
    <cmdp:ResourceProxyInfo xmlns:ns1="http://www.clarin.eu/cmd/1" ns1:ref="Res30000-0007-CFE2-1"> [3 lines]
    <cmdp:ResourceProxyInfo xmlns:ns1="http://www.clarin.eu/cmd/1" ns1:ref="Res40000-0007-CFE2-1"> [3 lines]
    <cmdp:ResourceProxyInfo xmlns:ns1="http://www.clarin.eu/cmd/1" ns1:ref="Res50000-0007-CFE2-1"> [3 lines]
    <cmdp:ResourceProxyInfo xmlns:ns1="http://www.clarin.eu/cmd/1" ns1:ref="Res60000-0007-CFE2-1"> [3 lines]
    <cmdp:ResourceProxyInfo xmlns:ns1="http://www.clarin.eu/cmd/1" ns1:ref="Res70000-0007-CFE2-1"> [4 lines]
</cmdp:ResourceProxyListInfo>
```

Figure 4: A CMDI fragment, where resource references are further described.

dataset's description, users can then, if interested, navigate to the section "Data files" to download the data streams associated with the data set one by one. Users will then need to repeat the process for the other six datasets, which is a rather tedious enterprise.

In the given example, this enterprise is tedious but relatively smooth: each handle associated with a dataset resolves to the metadata resource that is nicely displayed in the TALAR user interface; and each time, users only need to navigate through the same user-friendly interface to get download access to the research datasets. In general, datasets may refer to landing pages that are hosted on different servers and where landing pages may then differ significantly from each other, hence making is harder for the user to download the datasets of interest.

CMDI Explorer aims at giving users easy access to such complex collections of datasets. Our new software automatically follows the `ResourceProxyList` children of a CMDI metadata file, accumulates relevant information, and presents the resulting tree structure to users. Users can then navigate through the tree structure to identify and download research data they find relevant.

In line with the other pillars of the CLARIN infrastructure, CMDI Explorer is implemented as a web-based application. Similar to the Switchboard, users have three options to enter their CMDI metadata: they can enter a PID that resolves to the metadata, upload a metadata file from their local computer, or copy and paste metadata they have found elsewhere. Similar to the Switchboard, CMDI Explorer will have an open communication channel with the VLO and the VCR. That is, VLO and VCR users will get the option to send a CMDI file to CMDI Explorer for further analysis.

CMDI Explorer is designed to analyse highly recursive CMDI files and to provide a tree-based representation and visualisation of its entire structure, both in the browser and as a downloadable HTML and JSON file. CMDI Explorer can also work with resources composed of constituents that are described using different CMDI profiles: As it only relies on the contents of `ResourceProxyList`, it will just assemble all files referenced there irrespectively of their role in the CMDI file. As it is designed to work with collections, CMDI Explorer can operate across repositories, provided the metadata is freely available, *i.e.*, not behind an Authentication and Authorization Infrastructure (AAI) wall.

The implementation CMDI Explorer uses `Java` for the back-end and `react-js` for the front-end.

Back-end. The algorithm for retrieving the data associated with a collection is as follows: first, the CMDI XML data associated with the PID of the collection is retrieved. The `ResourceProxyList` is then extracted and each resource is analyzed. Resources with the resource type `Resource` are downloaded and saved as files. For resources with the resource type `Metadata`, the CMDI XML data is retrieved and the process is repeated recursively until all information for all files associated with the collection has been obtained. All information is then stored in a tree structure corresponding to the structure of the collection with the node names based on the names of the various collections and files, with the corresponding PID being added as a prefix to each file name in order to avoid any name collisions.

There were a number of challenges involved with the extraction of the data. Firstly, it is not always straightforward to extract the CMDI XML data associated with a PID. For each PID URL, there are a number of redirects to go through before arriving at the final URL.[2] Moreover, the path that the redirects take is also affected by the value of the `Accept` header. It was found that this needed to be set to `application/x-cmdi+xml, application/xml+cmdi, application/xml` for both following the redirects as well as extracting the data from the URL to cover the different possible Media Types the CMDI XML data could have as its `Content-Type`.

There are also a number of CMDI metadata resources in certain collections that are mislabelled as `Resource` as their resource type, when the resource type should be `Metadata`. In order to solve this, all resources labelled as `Resource` with an XML, CMDI, or no Media Type are analyzed and checked if they can be analyzed as a CMDI file, i.e., it is checked whether the file contains a `ResourceProxyList` which is non-empty. If so, then the resource is analyzed as CMDI metadata.

[2]For example, the handle `http://hdl.handle.net/11022/0000-0007-CFE2-1` redirects to `https://weblicht.sfs.uni-tuebingen.de/PidResolver/erdora/SFB833/A03/Compositional-Model-Sem/comp-models-sem-interp`, which in turn redirects to `https://talar.sfb833.uni-tuebingen.de/erdora/cmdi/SFB833/A03/Compositional-Model-Sem/comp-models-sem-interp`.

Another problem is that some resources are inaccessible for various reasons such as the need for authentication. Inaccessibility of resources is indicated in the data tree and the corresponding HTML or JSON files. Moreover, some collections are 'circular', *i.e.*, a resource in the CMDI metadata of one resource will eventually lead back to the original resource. CMDI Explorer keeps a record of resources encountered so far, and any resource that has already been analyzed is simply not included in the tree that is being constructed.

Front-end. Fig. 5 shows the main interface, presenting users the three options to enter their input. Once the metadata has been submitted, CMDI Explorer's back-end attempts to extract its underlying

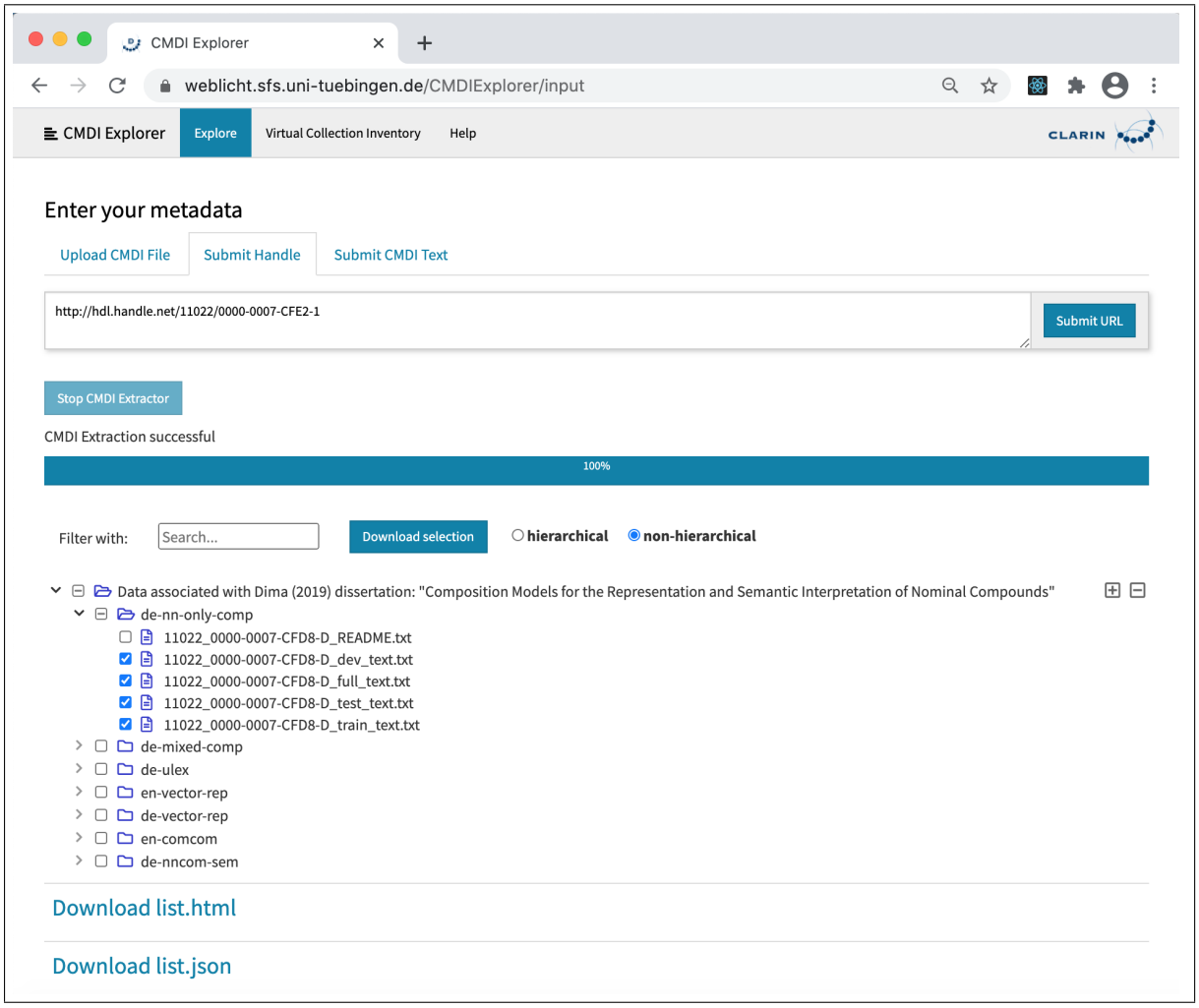

Figure 5: A CMDI record displayed in CMDI Explorer

tree. During extraction, users see a progress bar and live updates to the emerging tree visualisation. Fig. 5 shows a top node together with its immediate children. Each branch of the tree can be unfolded individually, and there are also two controls (+, −) to unfold or fold the entire tree. Leaf nodes are actionable. When users click on a leaf node, a pop-up window appears, see Fig. 6. The window shows some metadata about the selected resource such as its name, size and mediatype. It also gives users three follow-up actions to choose from: (i) copy the handle to the clipboard, (ii) click on the handle so that the respective resource resolves in the browser, and (iii) send the handle to the Switchboard for further processing.

Users can choose to download the entire tree, either in a structure-preserving or in a flatly structured HTML or JSON format (note the bottom two links in Fig. 5). Moreover, users can select nodes (subtrees, leaf nodes) individually. Selected nodes will be added to the 'download basket'; the corresponding data resources of their selection are made available as a ZIP archive file.

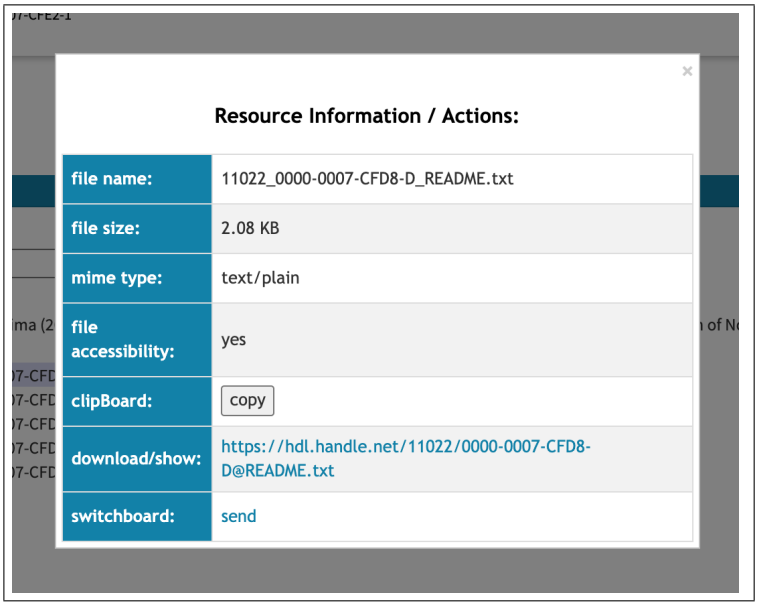

Figure 6: Actions attached to leaf nodes.

4 Current State and Future Work

We have built a prototype of CMDI Explorer that implements its core functionality and which is now open for beta testing at `https://weblicht.sfs.uni-tuebingen.de/CMDIExplorer/`. We invite readers to explore the tool and encourage their feedback. Given CMDI Explorer's lineage from the Switchboard's codebase, we expect users to easily grasp its user interface and the functionality it offers.

During testing, we were confronted with the fact that there is a limitation on the size of the files which can be downloaded by the user. Due to a limit in the size of array buffers in web browsers, the maximum file size which can be downloaded is about 2.14 GB, corresponding to the maximum 32-bit signed integer value. To overcome this limitation, we will need to enhance CMDI Explorer's back-end to deliver such files in piecewise fashion, and CMDI Explorer's front-end to reconstruct such files from the pieces delivered to the user.

The assemblage of resources from archives becomes a complex task when resources are protected by usage rights or licenses, and hence by AAI protocols. We are aware of the issue but are unsure which path to take as we do not want to move CMDI Explorer behind a Shibboleth wall.

We are also considering to use CMDI Explorer to hold an inventory of existing virtual collections (see the top navigation bar item "Virtual Collection Inventory" in Fig. 5). Here CMDI Explorer would simply show a manually curated catalogue of virtual collections that may be of interest to a wider CLARIN community. It would give a short description of a given collection together with a handle to access it. When clicking on the handle, CMDI Explorer would then show the tree structure of the corresponding collection.

CMDI Explorer has already been included in the test version of the Switchboard as a metadata processing tool. That is, when users give the Switchboard a CMDI file to process, the Switchboard identifies CMDI Explorer as applicable tool, which can then be directly invoked to show the possibly complex resource tree described by the CMDI metadata. We expect CMDI Explorer to be connected to the VLO and the VCR as well so that users can easily invoke it wherever they find complex CMDI metadata they need to explore further.

Acknowledgements

Our work was funded by the German Federal Ministry of Education and Research (BMBF), the Ministry of Science, Research and Art of the Federal State of Baden-Württemberg (MWK), and CLARIN-D. Emanuel Dima, Willem Elbers, Dirk Goldhahn, Marie Hinrichs and Dieter Van Uytvanck participated in the initial discussion and contributed to the conceptualisation of the explorer.

References

Daan Broeder, Menzo Windhouwer, Dieter Van Uytvanck, Twan Goosen, and Thorsten Trippel. 2012. CMDI: a component metadata infrastructure. In *Describing LRs with metadata: towards flexibility and interoperability in the documentation of LR workshop programme*, volume 1.

Corina Dima. 2019. *Composition Models for the Representation and Semantic Interpretation of Nominal Compounds.* Ph.D. thesis, University of Tuebingen.

Willem Elbers. 2017. Virtual collection registry v2. Technical report, CLARIN ERIC.

Margaret King, Davor Ostojic, Matej Ďurčo, and Go Sugimoto. 2016. Variability of the facet values in the VLO – a case for metadata curation. In Koenraad De Smedt, editor, *Selected Papers from the CLARIN Annual Conference 2015*, pages 25–44, Linköping. Linköping University Electronic Press.

Davor Ostojic, Go Sugimoto, and Matej Ďurčo. 2017. Curation module in action – preliminary findings on VLO metadata quality. In *Proceedings – CLARIN Annual Conference 2016*.

Dieter Van Uytvanck, Herman Stehouwer, and Lari Lampen. 2012. Semantic metadata mapping in practice: the virtual language observatory. In Nicoletta Calzolari et al., editor, *Proceedings of LREC'12*, Istanbul, Turkey. ELRA.

Claus Zinn, Thorsten Trippel, Steve Kaminski, and Emanuel Dima. 2016. Crosswalking from CMDI to Dublin Core and MARC 21. In Nicoletta Calzolari et al., editor, *Proceedings of LREC'16*, Portorož/Paris. ELRA.

Claus Zinn. 2018. The Language Resource Switchboard. *Computational Linguistics*, 44(4):631–639.

Matej Ďurčo. 2013. *SMC4LRT – semantic mapping component for language resources and technology*. Ph.D. thesis, Technische Universität Wien.