CDS-Videos-Transfer

Luis Vitor Zerkowski

Supervisor: Nicola Tarocco

November 17, 2023

Abstract

The CDS-Videos-Transfer application is a project aimed at facilitating and automatizing the migration of digitized videos from CDS to CDS Videos. The objective is to develop a local web application that enables seamless transfer of videos related to the Digital Memory Project from the CDS repository to CDS Videos, CERN'S official institutional repository for hosting and organizing multimedia content. In parallel with the software development, the CDS Videos web application and CDS-Dojson library have been improved, expanding their functions to ensure the continuous improvement of these components in the CDS ecosystem.

1 Introduction

The Digital Memory Project is an initiative that curates and preserves CERN's multimedia archive of the 20th century, capturing significant moments, seminars, conferences, and research events conducted at the laboratory a long time ago. To this end, the initiative seeks to find and gather old tapes lost in CERN's sea of archives and digitize them so that they can be preserved and made more accessible to the general public. Several of these videos have already been migrated to CDS Videos, CERN's official institutional repository for multimedia content, but there are still around 4,000 videos that still need to be transferred.

The intrinsic importance of the content of these videos requires a system to migrate and manage the data in CDS Videos [CDSc] effectively, ensuring the transfer of as many videos as possible without metadata loss. It was with this migration in mind that the idea of the CDS-Videos-Transfer project was born. The project's goal is to develop a local web application that enables seamless transfer of videos related to the Digital Memory Project from the CDS repository [CDSa] to CDS Videos. To accomplish what it proposes, the application can be understood and split into three key steps:

- **Local Web Interface for Record Selection:** The web application features an intuitive interface allowing users to enter record IDs or textual queries to start the metadata extraction process from CDS and respective conversion to CDS Videos schema. It also contains the 'Migrate All' option to let the user start transferring all Digital Memory Project records.
- **Metadata Analysis and Download:** Leveraging CDS-Dojson [CDSb], a `MARCXML`-to-`JSON` conversion tool developed for CDS, the project shows the record's metadata on both formats, so the user can evaluate the quality of the data extraction. At this point, the user can choose whether to download the metadata in the new `JSON` format or simply upload the records to CDS Videos.

- **Video Transferring:** If the user chooses the upload option, the application redirects to a video transfer page that shows the progress of the upload to the new platform. If the upload of a video does not work, the service displays the video that couldn't be transferred and the problem in the record metadata that led to the malfunction. The progress state of the migration is recorded on a database to make sure one can always return to the same point on the process.

For pipeline integration and effective service development, some adjustments also had to be made in both CDS Videos and CDS-Dojson. There were, hence, contributions to the CDS Videos and CDS-Dojson codebases, expanding their functions to ensure metadata extraction improvement and compatibility with CDS-Videos-Transfer.

2 Existing Tools

The first step in any migration process involves converting the data from the source format to the desired format, and in the case of CDS-Videos-Transfer it's no different. The metadata of the records stored on CDS is entirely in `MARCXML`, a formatting standard that combines the tags and codes of the `Marc21` with the markup language `XML`, while on the CDS Videos platform the standard used is JSON. To avoid re-implementing everything from scratch, a `MARCXML`-to-`JSON` metadata conversion library was used: CDS-Dojson. This package implements parsing functions for `MARCXML`, extracting tags and mapping these tags to `JSON` properties according to the `JSON` schema designed and used by CDS Videos.

The impact of using CDS-Dojson on the progress of the project was clear, allowing more development time to be devoted to the video migration platform. With extensive use of the package, however, it became clear that the code was already a bit outdated and couldn't handle a significant percentage of the videos to be migrated. In the following section, the difficulties and solutions involving the library are discussed.

3 Challenges

3.0.1 CDS-Dojson

Despite the simplicity of the project idea, many challenges made the implementation of the system much more difficult. The disorganized metadata found in the CDS records was the primary issue, making it hard for CDS-Dojson to handle all the tags. A lot of tags were simply being ignored in the format conversion. Others presented data capturing problems due to different content formatting for the collection of old videos in use. And for a slightly more serious problem of data loss, records with multiple videos were treated as if they were multiple files for the same video, not to mention the erroneous indexing of some tags. In the end, the pre-processing and cleaning of metadata turned out to be the most time-consuming task, making the CDS-Dojson updates an important part of the project.

Regarding the missing tags, a spreadsheet was created to establish the best actions to be taken for each tag that was not being handled. Most of the tags were allocated to a new property containing various digitization data. Some others were left out of the migration because they were legacy data that no longer made sense for the current platform. Finally, a few tags were added to the CDS Videos `JSON` schema, either in existing properties or in new properties. For the tags with a capture problem, the solutions were simpler, since most of them were just receiving unexpected values or unexpected value formatting. With minor `regex` adjustments, variable casting to different types and exception handling, the problems were all solved.

The non-identification of multiple videos problem, however, presented more risk to the migration and required further care to address. With all video files being placed in the same `JSON` record, the CDS Videos platform would interpret the multiple videos as additional files of the same video, affecting the individuality of the content. To overcome this, every record with multiple videos had to be split into multiple records, one for each video. By using the index code of each video, most tags could be allocated to their appropriate videos. It is worth noting, though, that the system still suffers from problems related to indexing, since many tags were not indexed or have indexing not compatible with the videos.

3.0.2 CDS Videos

Due to the substantial volume of videos requiring migration, the conventional approach of downloading all content locally and then re-uploading it to the new platform also represented a problem, but for processing time. A crucial project decision, thus, involved leveraging the CDS Videos API to upload videos remotely through links. In this way, all we have to do is provide the platform with a URI so that it triggers the download of the desired video. Encountering a malfunction in this tool, however, led to a critical phase of the project: rectifying the API.

Utilizing the multiprocessing capabilities offered by the `Celery` [CEL] library added complexity to identify the root cause of errors in the API calls. Fortunately, the underlying bug itself was relatively straightforward to address. The main issue was the order of video's post-processing operations. To resolve this, the solution involved maintaining a state to mark that the video file should be downloaded, instead of expecting the upload by the user. This successfully solved the problem.

4 CDS-Videos-Transfer

The development of the migration service, from a technical point of view, was designed to keep the implementation quite straightforward and easy to be both understood and modified in the future. The application was implemented using `Flask` as backend and primarily `JavaScript` on the frontend. Both parts of the project have their implementation steps described in the subsections below.

4.0.1 Backend

The application has three main pages for processing:

1. The homepage contains a short introductory text, two input fields and a button to migrate all records. The input fields allow the user to enter one or more record IDs to be processed, or a textual query to select a collection of records. When the user uses the input fields, the route separates the single record ID and the multiple record IDs or query option and sends a dictionary with such data to the `json_generator` route, which is responsible for the more intrinsic steps of metadata processing. If the button to migrate all records is pressed, the route is redirected to `uploading_json` directly and a database with all records and their migration state is accessed to start the video transferring process from where it left off.

   Figure 1 illustrates the homepage features.

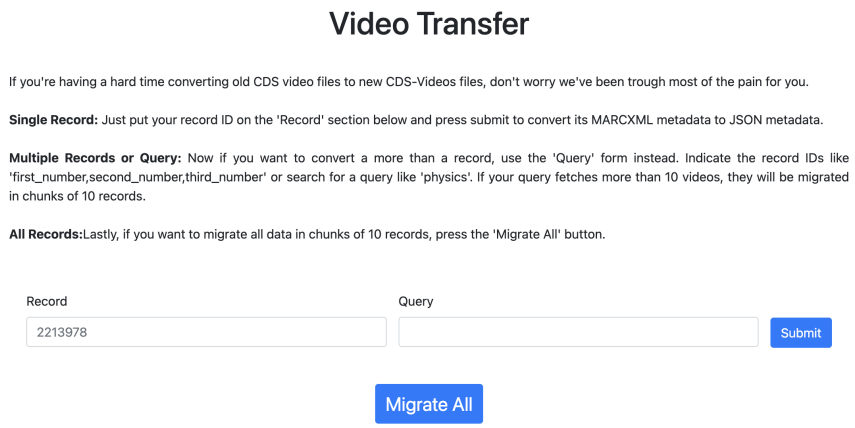


Figure 1: Root web page

2. The second step will take the user's input form in the homepage and start interpreting the type of processing requested. If a single record ID has been submitted, the preference will be to process

it. But if only the query input was used, then the query data is properly sanitized and parsed so that either a single textual query or multiple record IDs are processed.

After properly parsing the inputs, the route fetches data from CDS website. If it's up to 200 records, then a request is sent to CDS [CDSd] with some special URL building that guarantees a `MARCXML` response. But if there are more than 200 records involved in the query, then several requests are sent in a pagination fashion also using a special URL building that guarantees both the `MARCXML` response and also the proper pagination indexing. This procedure is necessary in order to avoid overloading the CDS platform.

Once the `MARCXML` is acquired, a processing step is implemented to split all the records in their own `MARCXML` structures, a necessary step to guarantee proper functioning of the CDS-Dojson library. With all single record `MARCXML`s in hands, the route uses CDS-Dojson functions to first create a `Python` dictionary-like blob with all the tags and their respective codes for each record and then generates a `JSON` based on the blob and a predefined CDS Videos `JSON` schema.

The `json_generator` then displays a simple interface with both the blobs and the `JSON`s, so the user can compare the original data with the new format data and evaluate the information loss. In addition, the route provides a link for the user to download the `JSON`s if they wish and also an upload button to directly trigger the videos migration process to CDS Videos. Lastly, users are provided with an option to return to the home page and restart the entire process.

Figure 2 illustrates the output of the transformation.

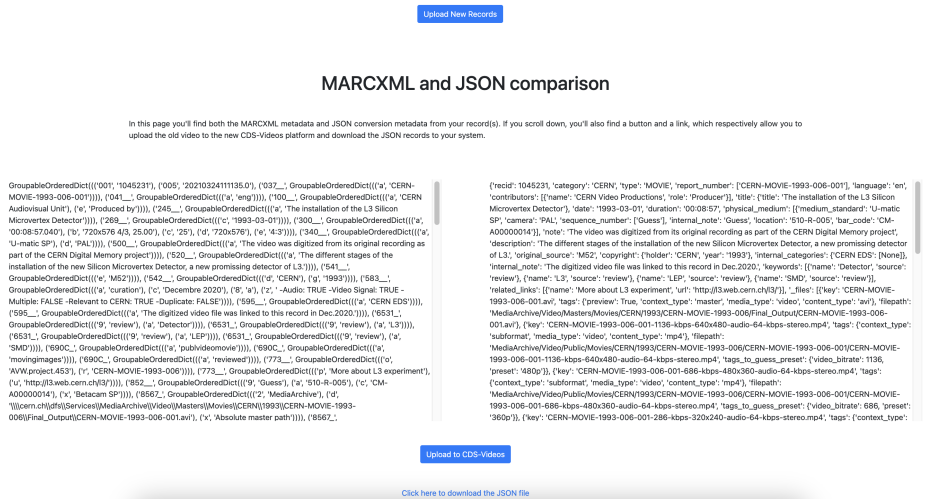
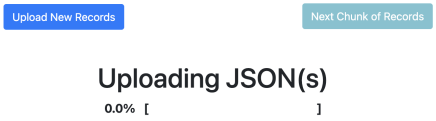


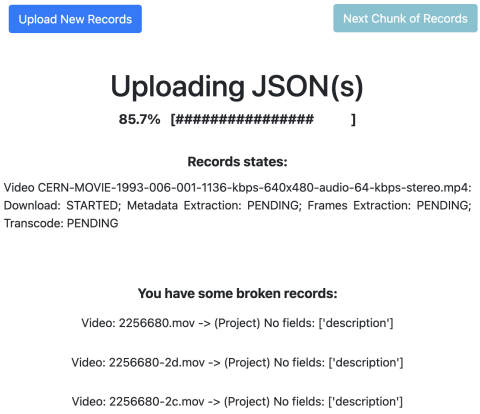
Figure 2: JSON generator web page.

3. Finally the last page features a progress bar that dynamically updates on the Frontend, synchronized with a Backend list that oversees the records upload process. An update log keeps track of the records successfully transferred to CDS Videos, elucidating the status of the processing steps performed by CDS Videos, such as download, metadata extraction, frame extraction, and transcoding tasks for each video. This log derives from an API endpoint on CDS Videos and serves to follow the progress of the upload. Additionally, should the Backend encounter any obstacles during the upload of a record, an error message is shown in the page. The migration process then proceeds in chunks of ten records, preventing the CDS Videos server and the user from being overwhelmed with huge amounts of data. Once all videos have been fully addressed, regardless of whether they were migrated or not due to an error, a `migration_state` database is updated to record the migration status of each video. Simultaneously, a button is activated to let the user trigger the process for the next chunk. As for the previous route, users are also provided with an option to return to the home page and restart the entire process.

   Figure 3 illustrates the transfer progress page.



(a) Upload on 0%.



(b) Upload in progress.

Figure 3: Uploading JSON web page.

4.0.2 Frontend

`Bootstrap` [BOO] was used as the main styling framework for the webservice. The ease with which the library allows to build a minimalist but still modern design for the page were the decisive factors for using it. When it comes to client-side and server-side interactions, `Jinja2` was naturally used since it natively communicates with the `Flask` API. Besides `Jinja2`, `JavaScript` with `jQuery` [JQ] were also used for such communications because they facilitate sending data to the server without page-loading and still allow dynamic adjustments.

The progress bar and the status of records, as well as the corrupted records section, have been implemented precisely with the above-mentioned tools. While all records are not processed, a `JavaScript` function makes requests to an endpoint `api/progress_list`, which returns the status of each already processed record. With every new processed record, the progress bar is dynamically updated in the `uploading_json` page and the status of the CDS Videos tasks are exhibited. In case of failure, a message is appended to the page, indicating the link of the video that could not be uploaded and the type of error in the video record that made it unable to be uploaded. As soon as the record chunk has finished processing, regardless of success or failure, the frontend requests that the backend update the database with the status of the videos targeted.

5 Conclusion

The CDS-Videos-Transfer web application, summarily, is a technically simple solution that helps with the migration of multimedia content from CDS to CDS Videos. With a focus on archive preservation, metadata extraction, and efficient data transfer, the developed CDS-Videos-Transfer application makes it easy for an operator to migrate digitized videos from CDS to CDS Videos. Thanks to this transfer, old multimedia content becomes accessible for users to watch. It is, therefore, another product within the CDS ecosystem aligned with CERN's ideals of preserving and sharing knowledge worldwide.

The project had its merits and demerits. On the one hand, large parts of code were updated, a lot of documentation on old videos was revised and a new video transfer service was fully implemented. The success of this project resides on the value provided by the new tool CDS-Videos-Transfer: by enriching the transformation rules, all already digitized videos and future ones, can be migrated from CDS to CDS Videos in a reliable way.

On the other hand, several obstacles prevented the system from being effectively used for video migration. The metadata of digitized videos in CDS turned out to be so incoherent and fragmented that made it very challenging to come up with a list of complete transformation rules. This phenomenon led to severe decrease on the number of videos that could be migrated with the time available for the project. In addition, some features were not implemented in time, which made it impossible to

actually transfer the videos, since the purpose of the migration is to be done just once, without having to revisit the migrated records in the future to retrieve the data.

In any case, a solid code structure remains, ready to be used in the near future with only a few adjustments. Futhermore, the learning opportunities were abundant and helped me build a more robust software development foundation that will certainly be of much value in my professional future.

References

[BOO] Bootstrap. https://getbootstrap.com. Last Access: 2023-08-25.

[CDSa] Cds. https://github.com/CERNDocumentServer. Last Access: 2023-08-25.

[CDSb] Cds-dojson. https://github.com/CERNDocumentServer/cds-dojson. Last Access: 2023-08-25.

[CDSc] Cds videos. https://github.com/CERNDocumentServer/cds-videos. Last Access: 2023-08-25.

[CDSd] Cds website. https://cds.cern.ch. Last Access: 2023-08-25.

[CEL] Celery. https://docs.celeryq.dev. Last Access: 2023-08-25.

[JQ] Jquery. https://jquery.com. Last Access: 2023-08-25.