

CERN Digital Memory Platform CERNBox file(s) upload integration

CERNBox file(s) upload via ownCloud's file picker

Aleksa Majkic^{1,2*}, Antonio Vivace² and Jean-Yves Le Meur²

¹Faculty of Electrical Engineering, University of Banja Luka, Patre 5,
Banja Luka, 78000, Bosnia and Herzegovina.

²IT-CA-IR, Digital Memory, CERN, Espl. des Particules 1, Meyrin,
1211, Geneva, Switzerland.

*Corresponding author. E-mail: aleksamcode@gmail.com;
Supervisor(s): antonio.vivace@cern.ch; jean-yves.le.meur@cern.ch;

Abstract

Constant production of an unprecedented data scale in various domains has led to a need to preserve the newly created information in such a way that it will be accessible in the future. Large institutions, like CERN, produce research data of an enormous importance which cannot be produced again in the future. This problem, known as digital preservation, has led to the design of a standard for long-term digital storage, known as OAIS standard. CERN's Digital Memory project, that is fully compliant with OAIS, provides researchers with a way to archive their data that will be available for future generations. In this project, which is a part of Digital Memory project, we have taken on a challenge of implementing a way to upload data from CERNBox, in addition to already existing sources like CERN Document Server (CDS), Indico, Gitlab, CERN Open Data and CodiMD, to the OAIS platform. To develop this functionality, we had to implement a REST API endpoint on the Digital Memory Platform using Python's framework Django in order to handle download of data from CERNBox, in addition to implementing ownCloud's file picker to the front-end of the platform.

Keywords: CERN, Digital Memory, OAIS, CERNBox, ownCloud, file picker, Python, Django, JavaScript, React

1 Introduction

The idea behind this project was to create an easy way for users to upload resources in order to allow for long-term preservation of data from one of the file hosting solutions called CERNBox¹. Currently, we are able to download users data and as a future task we want to package it consistently and process it with metadata validation and checksum calculations. With every new source addition, users are having less and less excuses for not preserving their data for the future. In order to create this feature, I have been developing both the front-end and the back-end part. The front-end part consisted of ownCloud's file picker integration with the Digital Memory Platform, and the back-end part consisted of developing an endpoint which would be able to handle downloading and archiving of files which are sent through the front-end part.

2 CERN Digital Memory Platform

2.1 OAIS Platform

The CERN Digital Memory Platform² handles the archive workflow and harvesting or download in case of the CERNBox upload. It consists of a REST API build using Django³. The goal of the projects was to create an easy way for users to upload resources from CERNBox. Selected user files are first downloaded after which the SIPs should get produced using BagIt Create⁴. For every upload request, files are stored in a unique temporary directory that gets its name derived from a time stamp.

All the tasks on the back-end work asynchronously in order not to keep the entire process on hold. This is the reason Celery⁵, a task queue manager, is used to run different tasks. It can execute asynchronously tasks outside the HTTP request-response cycle, which yields in better server response times. Celery integration with Django uses a data store and a message broker called Redis⁶. Ultimately, data is stored in a PostgreSQL⁷ relational database.

2.1.1 Archive

The first thing the back-end creates, after the new endpoint receives a list of links that point to the user files, is an Archive, a core element of OAIS Platform. It represents the whole history of a resource that the user is trying to preserve, and it can go through a number of different operations, which can transform uploaded data, like updating metadata in the Archive. When uploading data from CERNBox, archive's id is set to a unique value using a time stamp and the source is always marked as *CERNBox*.

While Archive created when sending files from other repositories can be “staged”, this is not a case when uploading files using CERNBox. Staged area, which is used to display a staged archive, is a preliminary area where the user can check which records are selected for archiving.

¹CERNBox is a digital repository at CERN which is based on ownCloud.

²<https://gitlab.cern.ch/digitalmemory/oais-platform>

³<https://www.djangoproject.com>

⁴<https://gitlab.cern.ch/digitalmemory/bagit-create>

⁵<https://docs.celeryq.dev/en/stable>

⁶<https://redis.io>

⁷<https://www.postgresql.org>

2.1.2 Step

Every step is a distinct operation on the Archive. This operation can be the initial download of the user content from the source repository, the validation of the package or the checksum validation. Each step is retrievable and independent from each other. In addition, every step has at any given moment a status that determines if the step is being executed or if it has finished.

2.1.3 REST API

The REST API is an API architecture style that uses HTTP requests to access and use data. The implemented endpoint for CERNBox is a POST endpoint, meaning the front-end in addition to “hitting” an endpoint needs to send some auxiliary information, in this case, a list of publicly accessible file links.

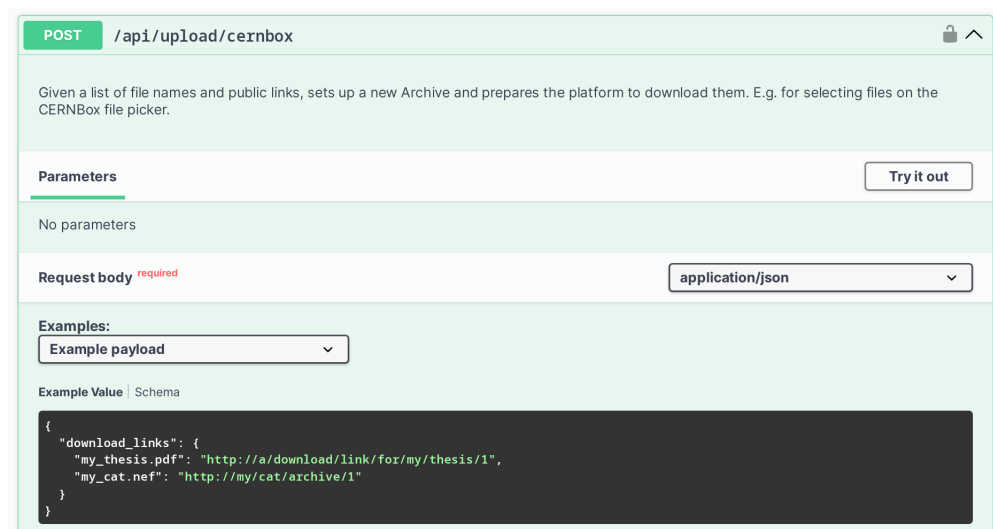


Fig. 2.1: Swagger UI Homepage screenshot

Developers, researches and archivists, can easily look up the Swagger UI web page for the API routes documentation. There they can see examples and make API calls directly.

2.2 OAIS User interface

The features exposed by the OAIS platform are accessible through the CERN Digital Memory web interface⁸. It provides the visualization of the archive and download process, the triggering of new jobs and additional information about the whole archiving process. This way, users are provided with a simple and intuitive way to archive and track the archival process through the UI.

For the UI development, we have used an open-source front-end JavaScript library called React⁹. There are many reasons why React was chosen for the front-end, including the existing familiarity and experience of the IT department with it, since the same stack is used on InvenioRDM¹⁰ and Indico¹¹.

⁸<https://gitlab.cern.ch/digitalmemory/oais-web>

⁹<https://react.dev>

¹⁰<https://inveniordm.web.cern.ch>

¹¹<https://indico.cern.ch>

2.2.1 CERNBox

Once the users have been successfully logged in, they will first see the homepage, from which they can navigate to any number of various pages using the navigation bar at the top. In order for users to add files to the platform, they must first navigate to the [Add Resources](#) page, which is the main input page of the application. Here, users can choose from which repository to add files to the platform. Naturally, this is where we have added a CERNBox upload feature, which is why we had to implement the file picker in order to allow users to choose which files to upload from their private repository.

2.2.2 Archive list & Archive details

After users choose which files to upload from CERNBox, a new Archive will be created and added to the list of already existing archives which can be viewed in the [Archive](#) page. By clicking on the Archive and opening the [Archive Details](#) page, auxiliary information will be displayed including the archive's Title, Record ID, Source, Tags etc.

The [Archive Details](#) page, as shown in the picture below (Figure 2.2), is a page where users can keep track of the archiving status. At the moment, the only available step in the [Pipeline overview](#) section is **Download files from CERNBox**, but ultimately an additional three steps will be made available:

- **Validate**
- **Checksum**
- **Upload from CERNBox**

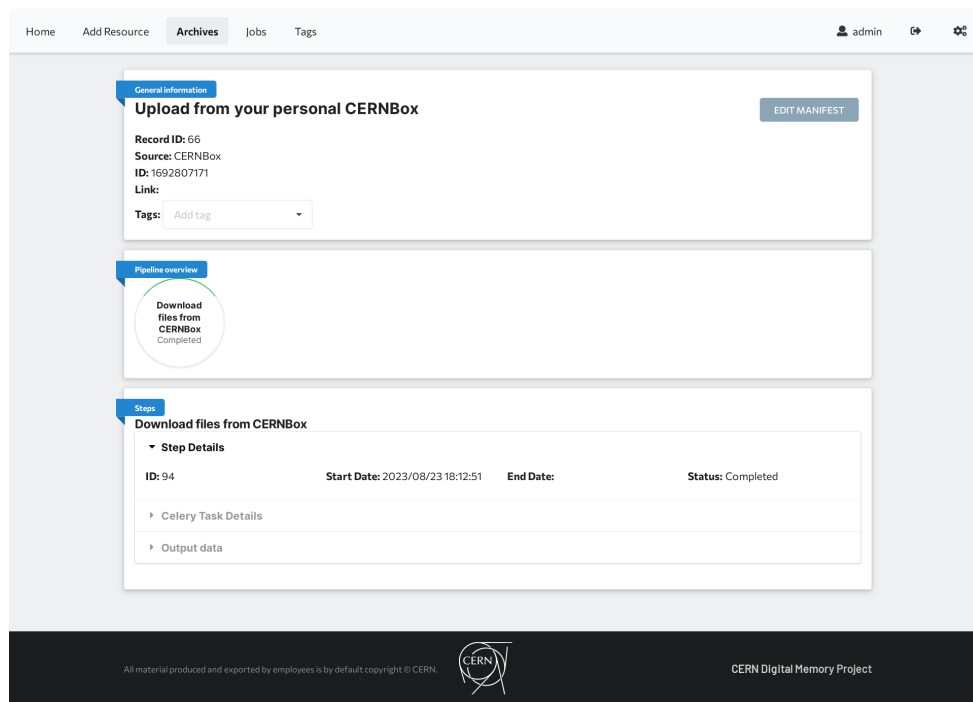


Fig. 2.2: OAIS User Interface - Archives Details page

2.2.3 File picker

File picker is a web component written in Vue.js¹² which can be integrated into the existing OAIS front-end. It connects to an ownCloud server and it enables users to select resources which are then provided in a response to a fired event. In this project, in place of the ownCloud server, we are using the CERNBox which is a customised instance of the ownCloud. While the file picker provides an object representation of the resources, it does not provide the actual content itself. In order to use resources and perform action on it, we had to utilize the API within the OAIS front-end. As a result, the file picker enables users to select multiple resources, and it brings resources from within CERNBox into our front-end web application.

Successive depiction of objects interaction during a successful upload of user files from CERNBox is shown in the diagram below (Figure 2.3). User performs file selection through the usage of the file picker after which the links to user files are forwarded to the parent window that makes an API call to the appropriate endpoint with the given links.

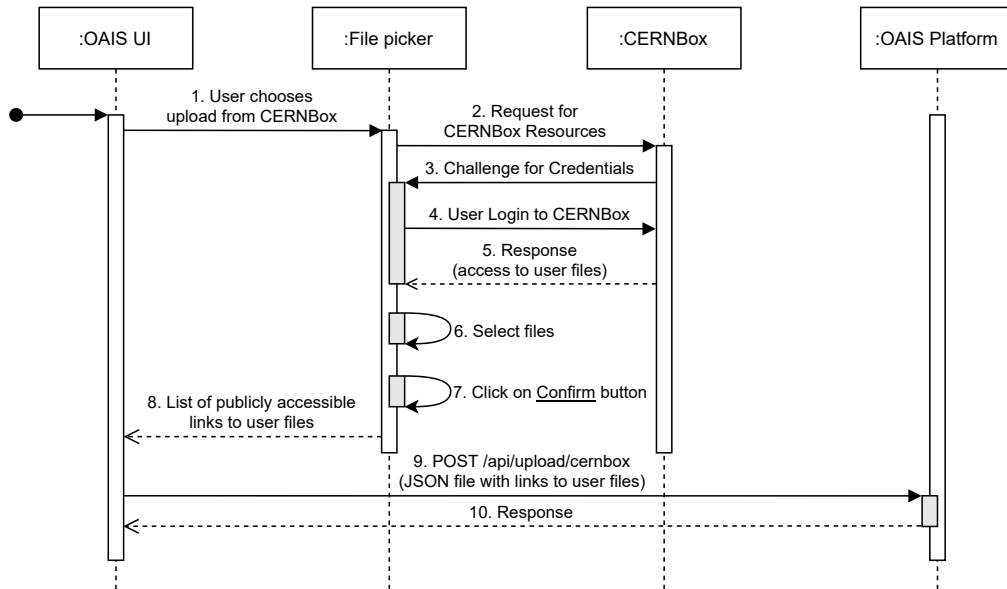


Fig. 2.3: Example of a successful workflow for user files upload from CERNBox

Moreover, it is important to note, our starting point for integration was the modified file picker from the *changes-for-indico*¹³ branch. The modified version had additional changes made for Indico in order to remove unnecessary options and add additional features that were also useful in our platform, such as disabled directory selection.

¹²<https://vuejs.org>

¹³<https://github.com/javfg/file-picker/tree/changes-for-indico>

2.2.4 File picker wrapper

The file picker wrapper (Figure 2.4) is a thin layer that acts as a bridge (common interface) between the file picker and the front-end application which is implementing it in a such way that allows us to create file links which will later be utilized, in the back-end, to download user files. The addition of the wrapper at the same time adds architecture complexity and simplifies the implementation of the file picker. By adding a wrapper, which handles the logic of creating download links, the process of the file picker implementation to different projects has been simplified as developers no longer have to implement the logic of creating download links themselves. The cost of using the wrapper on the other hand is a slightly more complex architecture and communication between the file picker and the host (parent) application through the usage of window post messages.

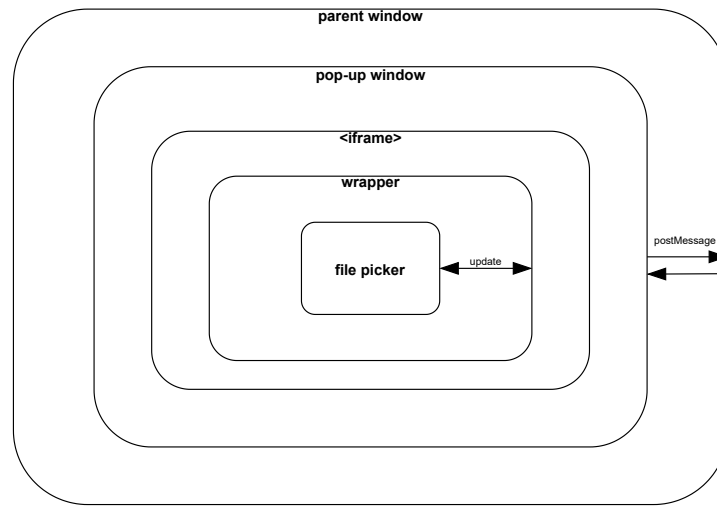


Fig. 2.4: OAIS User Interface - File picker wrapper architecture

2.2.4.1 CERNBox file links

When sharing links to access user private files, there are two options, using public links or using links with Bearer tokens. Due to time constraints during the development, we proceeded with a simplified workflow using public links that have an expiry time of 60 minutes. As a result, the file picker wrapper needed to be called using the following URL parameters: *?publicLink=true,publicLinkDuration=60*. Usage of the URL parameters could have been avoided by setting the *publicLink* and *publicLinkDuration* values in the wrapper, instead of sending them through the link and then parsing them in the wrapper. However, this would mean adding additional changes to the file picker wrapper that are Digital Memory Platform specific, and consequently the flexibility to easily change these values in the future would be decreased.

2.2.5 Post message communication

The `window.postMessage()` method safely enables cross-origin communication between window¹⁴ objects. Specifically, it allows us to communicate between a page in our front-end, which allows us to open a file picker, and a pop-up which has a file picker wrapper embedded in as an iframe. With this we are able to safely circumvent the same-origin¹⁵ restriction. It is also important to note that the Window Post message method is not the same as the HTTP POST method¹⁶.

Once the changes occur on the file picker, they are propagated to the wrapper which sends a post message to the parent every time a user changes the current file selection. While the parent window constantly receives post messages, only after the user clicks on the Confirm button inside the pop-up window is the post message used to obtain the file download links.

3 Results

The work done in this project provided a way to integrate the CERNBox service into the Digital Memory Platform. Laying the groundwork needed in order to use it like any of the supported “upstream sources”, enables the end users (and systems) to trigger a preservation pipeline using files originally uploaded to CERNBox.

4 Conclusion

During my stay at CERN, I had an opportunity to learn, and learn about a wide variety of platforms, technologies, and frameworks without which this project would not exist.

The work on the project was interesting and challenging, which in turn has allowed me to see first-hand the importance of data preservation and to learn about its many challenges. While this area of computer science is not as talked about as some other areas like AI and machine learning, it is equally important and useful.

4.1 Future work

Since the platform is under development, there is more work to be done before the platform could be deployed and made available to the public. What is left to implement in order to make CERNBox file upload available for users is to complete the file picker wrapper implementation in the front-end. After implementing the file picker, users should be able to seamlessly upload file(s) from their own CERNBox to the Digital Memory Platform. In addition, solutions to some of the project’s other problems still need to be decided:

- **Packaging data in a consistent way:** Currently, we are able to download the user’s data, but the idea is to package the data (create SIPs) using BagIt Create which is responsible for package creation according to the CERN SIP Specification. The system should then perform additional steps like validation of the format and checksum validation.
- **Formalizing the file upload limit:** At the moment, there is an arbitrary file amount upload limit set in the code that prevents users from uploading large quantity of files. This limit should be formalized and revisited in the future with concrete performance metrics that justify the chosen limit.

¹⁴<https://developer.mozilla.org/en-US/docs/Web/API/Window>

¹⁵https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

¹⁶<https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST>

- **CERNBox file upload without usage of the expiring public links:** The current approach relies on the creation of the public links with a short life span. This approach creates a potential vulnerability in the system by exposing users file to the public. While the attack surface of the vulnerability is small and an exploit is highly improbable due to utilization of HTTPS, the usage of user tokens to achieve file upload would be preferable over upload using public links. This would require a rewrite of the back-end and the front-end code.
- **Allowing CERNBox upload of the whole directory:** At the moment, there is no exposed endpoint at CERNBox that would allow developers to simply get links for files residing inside the directory or that would allow archiving of its content, which would then be uploaded. This might not be the case in the future, at which point implementing this feature could be easily done. Another approach could be to firstly obtain a list of files residing inside of a selected directory and then to obtain links for download of each individual files that would then be bundled and sent to the back-end part of the platform where they would be downloaded.

4.2 Acknowledgments

Finally, I am grateful for this amazing opportunity to work at CERN; I am grateful for my supervisor that has been there every step of the way to guide me throughout the whole project.

Also, I would be remiss if I did not mention Javier Ferrer¹⁷, who was not only willing to answer all of my question, but was of utmost importance when I first started to work with the file picker for the OAIS user interface.

¹⁷javier.ferrer@cern.ch

Appendix A REST API

API endpoint that allows to upload files from CERNBox

- POST */api/upload/cernbox/*

Appendix B Acronyms

API Application Programming Interface

CERN European Organization for Nuclear Research

ID Identification

OAIS Open Archival Information System

SIP Submission Information Package

UI User Interface

References

- [1] Chelakis K., “Creating an Open Archival Information System compliant archive for CERN”, CERN, Geneva, Dec 2022 [Online]. Available: <https://cds.cern.ch/record/2857666>
- [2] CERN Digital Memory git repositories. [Online]. Available: <https://gitlab.cern.ch/digitalmemory>
- [3] Digital Memory project in two words. [Online]. Accessed: 2023-07-10. Available: <https://digital-memory-project.web.cern.ch>
- [4] Le Meur J.-Y., “The Digital Memory of CERN: status and input for a Preservation Strategic Plan”, CERN, Geneva, Jul 2016 [Online]. Available: <https://cds.cern.ch/record/2200146>
- [5] Le Meur J.-Y., “CERN Digital Preservation Strategy: proposal”, CERN, Geneva, Tech. Rep., Feb 2023. [Online]. Available: <https://cds.cern.ch/record/2856775>
- [6] Liptak P. G., “Digital Memory at CERN - Archiving and preserving existing knowledge”, CERN, Geneva, Sep 2021 [Online]. Available: <https://cds.cern.ch/record/2779856>
- [7] Marcos Sanchez De La Blanca C., “InvenioRDM integration for the CERN Digital Memory platform”, CERN, Geneva, Sep 2022 [Online]. Available: <https://cds.cern.ch/record/2834158>
- [8] Vivace A., Le Meur J.-Y., “The Challenge of Digital Preservation at CERN”, CERN, Geneva, May 2023 [Online]. Available: <https://cds.cern.ch/record/2857550>