Initial definition of protocols and APIs

Project acronym: CS3MESH4EOSC

Deliverable D3.1: Initial Definition of Protocols and APIs

Contractual delivery date	30-09-2020
Actual delivery date	16-10-2020
Grant Agreement no.	863353
Work Package	WP3
Nature of Deliverable	R (Report)
Dissemination Level	PU (Public)
Lead Partner	CERN
Document ID	CS3MESH4EOSC-20-006
Authors	Hugo Gonzalez Labrador (CERN), Guido Aben (AARNET), David Antos (CESNET), Maciej Brzezniak (PSNC), Daniel Muller (WWU), Jakub Moscicki (CERN), Alessandro Petraro (CUBBIT), Antoon Prins (SURFSARA), Marcin Sieprawski (AILLERON), Ron Trompert (SURFSARA)

Disclaimer:

The document reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 863353

Table of Contents

This document is licensed under a Creative Commons Attribution 4.0 International (CC BY-4.0) License, with the exception of the EU, EOSC, CS3MESH4EOSC and CS3 logos, as well as any other resources marked as such.

1 Introduction

This document describes the development and definition of a set of vendor-neutral protocols and APIs necessary to build the full Science Mesh service, including federated sharing of data, access to application functionality and infrastructure integration. Our approach is based on:

1. maturing and extending protocols already in use by CS3 community (e.g. OCM, CS3APIs) with the aim to advance these protocols from TRL6 to TRL9;

2. adopting other existing standard protocols wherever appropriate (e.g. WOPI, WebDAV, ...).

This document is divided in several sections.

Core APIs identifies the core APIs and protocols for the Science Mesh service: OCM and CS3APIs.

Use Cases further explores the usage of these interfaces in the four reference application workflows developed in the context of Work Package 4: Data Science Environments, Open Data Systems, Collaborative Document Editing and On-Demand Data Transfers.

EFSS integration with IOP section describes the architecture connecting EFSS with the Interoperability Platform (IOP). IOP provides the reference implementation for the Core and Use-Case APIs.

Integration of Peer-to-Peer architecture with IOP is described based on the Cubbit implementation.

ScienceMesh Infrastructure APIs are presented for monitoring endpoints.

2 Core APIS

This section describes the core APIs, protocols and their purpose and role in the Science Mesh service. As a design principle, industry standards for the transport protocol layer (for example, HTTP and gRPC) are used to interconnect server-side components and to achieve maximal interoperability. The choice of the transport protocol is based on the following criteria: quality of implementation; wide adoption in the industry and in open-source projects, support for multiple programming languages (to increase integration with all systems based on different technologies) and enterprise-grade tooling (debuggers, testing frameworks).

2.1 Open Cloud Mesh (OCM)

2.1.1 Introduction

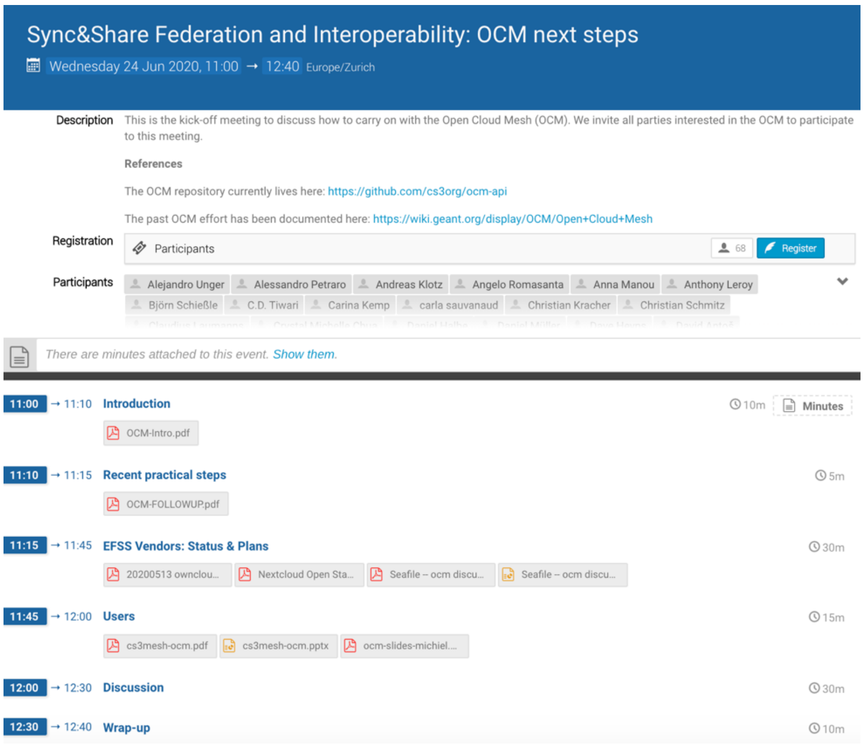
Open Cloud Mesh (OCM) was started in 2015 as a joint international initiative, under the umbrella of the GÉANT Association, to develop a web-based protocol for universal file access beyond individual clouds which would enable a creation of a globally interconnected mesh of research clouds without sacrificing any of the advantages in privacy, control and security an on-premises cloud provides. OCM defines a vendor-neutral, common file access layer across an organization and/or across globally interconnected organizations, regardless of the user data locations and choice of clouds.

The first phase of the OCM development ended with the initial version OCM v.0.0.3 which was published in an open repository[1] and presented at the CS3 2017 Workshop in Amsterdam (cs3community.org). Further OCM stakeholder discussions had been held at the yearly CS3 Workshops in 2018, 2019 and 2020

Following the initial API specification, the software companies ownCloud, Nextcloud and Seafile developed initial implementations. However, no further interoperability tests have been conducted between these different platforms. Consequently, the potential of OCM has not yet been fully realized in spite of sustained interest among the CS3 workshop participants, and beyond, in the wider Education and Research Community.

2.1.2 Advancing OCM

As a follow up of the discussions held at the CS3 2020 Workshop in Copenhagen, the OCM Workshop was organized in June 2020 (project M6) in the context of WP3 activities to develop and promote neutral APIs.


Sync&Share Federation and Interoperability: OCM next steps

Wednesday 24 Jun 2020, 11:00 → 12:40 Europe/Zurich

Description: This is the kick-off meeting to discuss how to carry on with the Open Cloud Mesh (OCM). We invite all parties interested in the OCM to participate to this meeting.

References

The OCM repository currently lives here: https://github.com/cs3org/ocm-api

The past OCM effort has been documented here: https://wiki.geant.org/display/OCM/Open+Cloud+Mesh

Registration: Participants 68 Register

Participants: Alejandro Unger, Alessandro Petraro, Andreas Klotz, Angelo Romasanta, Anna Manou, Anthony Leroy, Björn Schießle, C.D. Tiwari, Carina Kemp, carla sauvanaud, Christian Kracher, Christian Schmitz, Claudius Laumanns, Crystal Michelle Chua, Daniel Halbe, Daniel Müller, Dave Heyns, David Antoš

There are minutes attached to this event. Show them.

11:00 → 11:10 Introduction — 10m — Minutes
OCM-Intro.pdf

11:10 → 11:15 Recent practical steps — 5m
OCM-FOLLOWUP.pdf

11:15 → 11:45 EFSS Vendors: Status & Plans — 30m
20200513 ownclou... | Nextcloud Open Sta... | Seafile – ocm discu... | Seafile – ocm discu...

11:45 → 12:00 Users — 15m
cs3mesh-ocm.pdf | cs3mesh-ocm.pptx | ocm-slides-michiel....

12:00 → 12:30 Discussion — 30m

12:30 → 12:40 Wrap-up — 10m

Figure 1. Agenda of the first OCM Workshop (https://indico.cern.ch/event/928998)

The workshop was attended by 68 participants from the traditional CS3 community: educational and research institutions, NRENs, contributors to web standards and projects such as SOLID2, and the

[1] https://github.com/GEANT/OCM-API
[2] https://en.wikipedia.org/wiki/Solid_(web_decentralization_project

principal EFSS vendors (Owncloud, Nextcloud, Seafile). The main discussion points:

- Transfer of the OCM API repository to the community-based CS3 Organization on Github3 to provide a stronger drive for the evolution and adoption of the OCM protocol;
- Review of the existing proposals for changes and agreement on the schedule to tag and release OCM v1.0.0
- Summary of plans of EFSS vendors on further OCM support and development
- Presentation of ScienceMesh as a user of OCM
- Discussion on web standardisation, protocols and vendor-independent testing.

The workshop resulted in the following practical actions:

- Review of the most prominent Pull Requests to version 0.0.3 in the Github repository[4] (June 2020);
- merge of Pull Request #37[5] in agreement with all EFSS vendors (July 2020);
- release of version 1.0.0 of the OCM API[6] (September 2020).

OCM v1.0.0 API has been exported to a persistent repository: DOI 10.5281/zenodo.4045360[7].

In addition, the workshop participants agreed to organize regular meetings to track evolution of OCM as well as starting of the "OCM user" forum organized by the former OCM coordinator at GÉANT.

2.1.3 Extensions

The existing OCM protocol allows to initiate a "rendezvous" between two systems to exchange data. The most common use-case is to allow access to a given dataset at a source system ("provider") to a target system ("consumer"), a mechanism which is usually known as "sharing". The existing specification has been crafted with the following assumption in mind: "*the provider knows the consumer (both endpoint and user) when it creates a share with the consumer. How this is known is not part of this spec*."

While this "omission" can be seen as a way to keep the protocol simple by focusing on its core functionality, it deprives OCM of what could otherwise be a powerful vendor-agnostic user discovery mechanism. And such a mechanism would be a key enabler of a large-scale cross-vendor distributed system such as ScienceMesh. The mechanism envisioned to share and exchange data in Science Mesh must be:

- Simple and easy to use by end-users as if they were sharing data with users in the same platform

[3] https://github.com/cs3org/ocm-api
[4] https://github.com/cs3org/ocm-api
[5] https://github.com/cs3org/OCM-API/pull/37
[6] https://github.com/cs3org/OCM-API/releases/tag/v1.0.0
[7] https://zenodo.org/record/4045360#.X2r92ZMza-Y

- Respectful of the privacy of the user

A survey was conducted amongst the project partners to understand the readiness of their infrastructures to connect to EduGain. The results were collected from 9 infrastructures, out of which 7 operate ownCloud, one Seafile and one a proprietary solution. A single site is directly connected to eduGAIN, 6 use their national academic identity federations, one none of those. Out of the federated ones, two accept all identities from the federation, while others perform various levels of attribute filtering, ranging from simple organisation affiliation to quite complex rules based on combination of attributes for various organisation types, depending on eligibility of users to consume infrastructure services.

The results of the survey highlighted even more the need to have a de-centralized approach to user discovery at this phase of the project to enrol existing sites while they prepare for adopting EduGain.

The Invitation Workflow has been developed by the CS3MESH4EOSC project to meet these requirements as a vendor-neutral extension of the OCM.

2.1.3.1 *OCM Invitation Workflow*

Current Open Cloud Mesh API implements sharing that requires the user to know an exact identity of the party in the remote system. While such API functionality is necessary to implement practical scenarios, from a user experience point of view, such knowledge cannot be reasonably expected. The invitation to sharing can be sent via a separate communication channel; and e-mail address is probably the most common identification the users will know and commonly use. We have designed a scenario based on this observation.

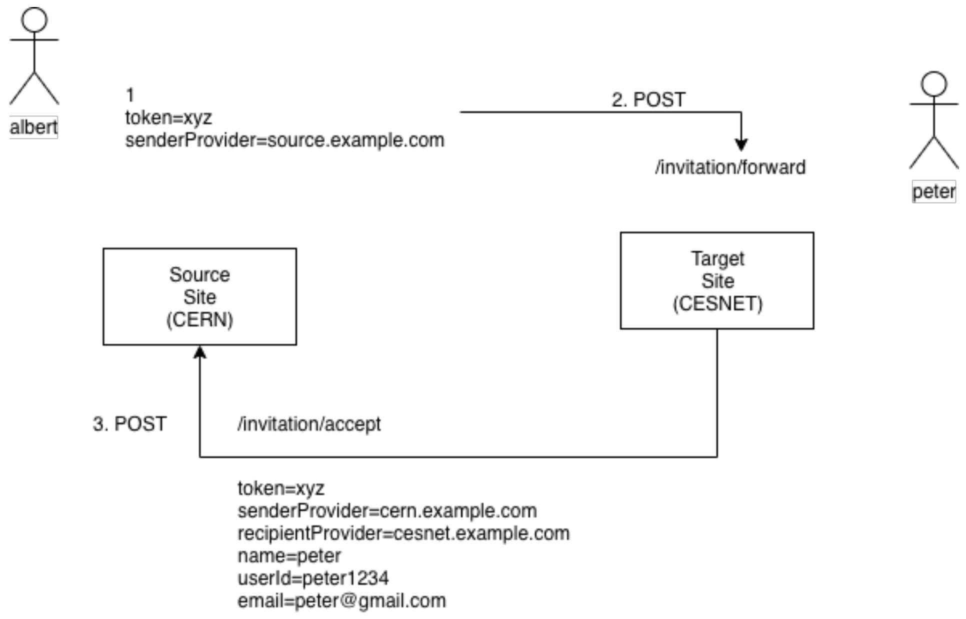


Figure 2. Invitation-Work flow (OCM)

1. The first step is to generate a unique token to be used to authorise the remote system to accept the invitation. The generation of the token is outside the scope of this extension.

2. The token and the domain of the source system are transmitted to the remote system. The mechanism to send this information is outside the scope of the API, implementers have a vast range of alternatives to achieve it (email, QR code, ⋯).

3. The information (token and sender domain) arrives to the target system. The target system obtains the user identity of the logged in user and send it back to the source with the domain of the target system. The source system knows now the identity at the target system to initiate an OCM share.

The extension is proposed for inclusion into OCM in the following Pull Request 41[8].

Two new endpoints are introduced:

- **/ocm /invites/forward (POST):** this endpoint is available on the target system and is used to forward an invitation received through an out-of-band channel (like email).

- **/ocm /invites/accept (POST):** this endpoint is available on the source system and is used to accept the invitation to trigger the default OCM workflow and establish the rendezvous

2.2 CS3APIs

The Cloud Sync and Share Services APIs (**CS3APIs**) provide an interoperable connection among sync and share platforms, storage providers and application providers; decreasing the burden of porting applications developed for different EFSS platforms. This also allows connect the EFSS for existing or new storage repositories over a well-defined metadata control plane. For site administrators of sync and share services, CS3APIs allow to reuse existing connectors to other research services (such as digital repositories) independently of the technology stack. Finally, interoperability decreases the risk of vendor lock-in and ease migration to other CS3APIs-compatible software stacks.

The official documentation of the APIs is available online: https://cs3org.github.io/cs3apis/.

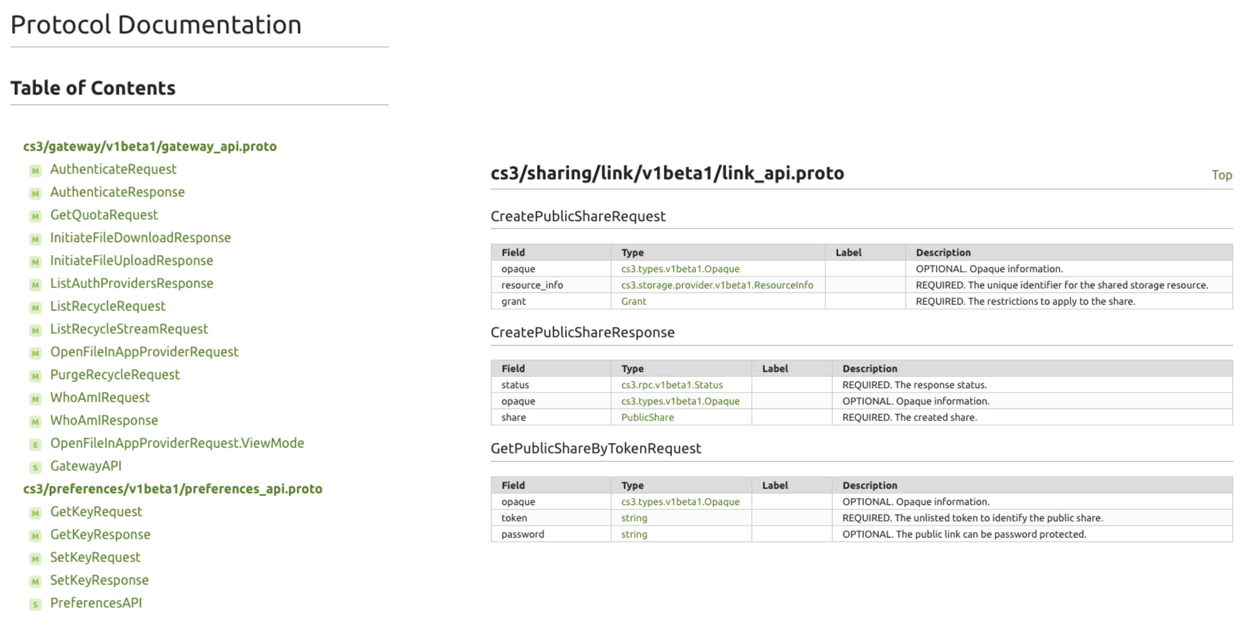

Protocol Documentation

Table of Contents

cs3/gateway/v1beta1/gateway_api.proto
- M AuthenticateRequest
- M AuthenticateResponse
- M GetQuotaRequest
- M InitiateFileDownloadResponse
- M InitiateFileUploadResponse
- M ListAuthProvidersResponse
- M ListRecycleRequest
- M ListRecycleStreamRequest
- M OpenFileInAppProviderRequest
- M PurgeRecycleRequest
- M WhoAmIRequest
- M WhoAmIResponse
- E OpenFileInAppProviderRequest.ViewMode
- S GatewayAPI

cs3/preferences/v1beta1/preferences_api.proto
- M GetKeyRequest
- M GetKeyResponse
- M SetKeyRequest
- M SetKeyResponse
- S PreferencesAPI

cs3/sharing/link/v1beta1/link_api.proto Top

CreatePublicShareRequest

Field	Type	Label	Description
opaque	cs3.types.v1beta1.Opaque		OPTIONAL. Opaque information.
resource_info	cs3.storage.provider.v1beta1.ResourceInfo		REQUIRED. The unique identifier for the shared storage resource.
grant	Grant		REQUIRED. The restrictions to apply to the share.

CreatePublicShareResponse

Field	Type	Label	Description
status	cs3.rpc.v1beta1.Status		REQUIRED. The response status.
opaque	cs3.types.v1beta1.Opaque		OPTIONAL. Opaque information.
share	PublicShare		REQUIRED. The created share.

GetPublicShareByTokenRequest

Field	Type	Label	Description
opaque	cs3.types.v1beta1.Opaque		OPTIONAL. Opaque information.
token	string		REQUIRED. The unlisted token to identify the public share.
password	string		OPTIONAL. The public link can be password protected.

Figure 3. CS3APIs documentation (sample).

[8] https://github.com/cs3org/OCM-API/pull/41

The CS3APIs are split into modules that can be implemented independently depending on the desired level of integration.

- Gateway API: it provides a façade API to clients, abstracting them from complex interactions amongst the rest of the APIs.
- Preferences API: providers endpoints for storing user preferences (for example, default text editor, theme, ...).
- AppProvider API: provides endpoints to expose third-party applications to be integrated into EFSS UIs (for example, Collabora).
- AppRegistry API: provides endpoints to discover applications for different mime types and file extensions.
- AuthProvider API: provides endpoints to authenticate users based on different strategies (basic auth, OIDC, service accounts, ...).
- AuthRegistry API: provides endpoints to discover authentication providers.
- IdentityUser API: provides endpoints to obtain user information.
- SharingCollaboration API: provides endpoints for sharing with user and group of the system.
- SharingLink API: provides endpoints to create public links.
- SharingOCM API: provides endpoints to create OCM shares.
- StorageProvider API: provides endpoints to manipulate an underlying storage system.
- StorageRegistry API: provides endpoints to discover available storage providers.
- OCMCore API: provides endpoints to accept OCM shares.
- OCMInvite API: provides endpoints to create OCM invites.
- OCMProvider API: provides endpoints to verify OCM sources.

2.3 WebDAV

WebDAV (Web Distributed Authoring and Versioning) is an extension of the Hypertext Transfer Protocol (HTTP) that allows clients to perform remote Web content authoring operations. WebDAV is defined in RFC 4918 by a working group of the Internet Engineering Task Force.

The WebDAV protocol provides a framework for users to create, change and move documents on a server. The most important features of the WebDAV protocol include the maintenance of properties about an author or modification date, namespace management, collections, and overwrite protection. Maintenance of properties includes such entities as the creation, removal, and querying of file information. Namespace management deals with the ability to copy and move resources within a server's namespace. Collections deal with the creation, removal, and listing of various resources. Lastly, overwrite protection handles aspects related to locking of files.

WebDAV is extensively used already in many EFSS vendors (ownCloud[9], nextCloud[10], SeaFile[11], Pydio[12]) to manipulate the namespace and to download an upload of files. ownCloud and nextCloud go a step further to rely on WebDAV semantics for their respective desktop and mobile synchronization applications. Based on the existing built-in support by vendors and de-facto transfer protocol promoted in OCM, ScienceMesh will leverage on it.

2.4 WOPI

The Web Application Open Platform Interface (WOPI) protocol defines APIs used to access and change documents stored on a storage system, with emphasis on collaborative workflows. Applications utilize this functionality by accessing various endpoints offered by a WOPI server adhering to the WOPI protocol specifications. This allows them to work with files in a storage system-agnostic way.

The IOP developed for the CS3MESH4EOSC project includes such a WOPI server that implements the WOPI protocol and uses a CS3APIs compliant storage system as its file backend. This means that this WOPI server serves as a bridge between any WOPI client and any underlying CS3APIs provider (e.g. *Reva*). Instead of having to work with the CS3APIs directly, a WOPI client can rely on the WOPI server to deal with stored documents, shared locks, and so on.

A number of web applications for collaborative editing already support the WOPI protocol and can act as a WOPI client. These applications can be integrated by simply having them use our built-in WOPI server. This also means that no changes need to be made to the editing software. One such example is Collabora CODE, which enables collaborative editing of OpenOffice documents and has already been deployed in the CS3MESH4EOSC project.

3 Use-Cases

In this section we present initial specification of APIs and application workflows for the CS3MESH4EOSC project. The application components interact with local EFSS service via the IOP (Interoperability Platform) using the CS3APIs.

[9] https://doc.owncloud.com/server/10.5/
[10] https://docs.nextcloud.com/server/15/user_manual/files/access_webdav.html
[11] https://download.seafile.com/published/seafile-manual/home.md
[12] https://pydio.com/en/docs/v8/setup-webdav-server-access

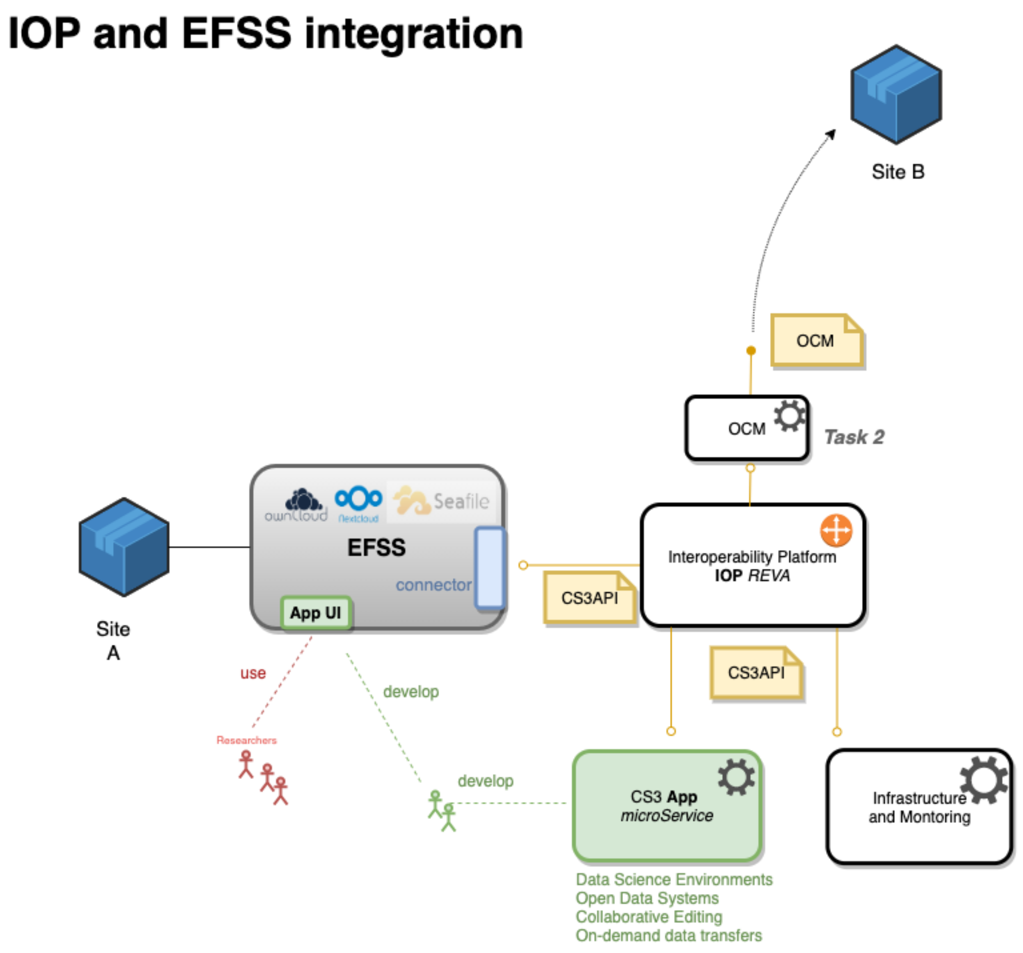


Figure 4. IOP and EFSS Integration Architecture

3.1 Data Science Environments

The concept of distributed Data Science Environments is to facilitate collaborative research and enable sharing of computational tools, algorithms, and resources across the ScienceMesh. This is to allow users to access remote execution environments to replay (and modify) analysis algorithms without the need to set-up accounts upfront in the remote services. Interconnection of a web-based Jupyter environments with the web-based EFSS environment is ideally suited for this task.

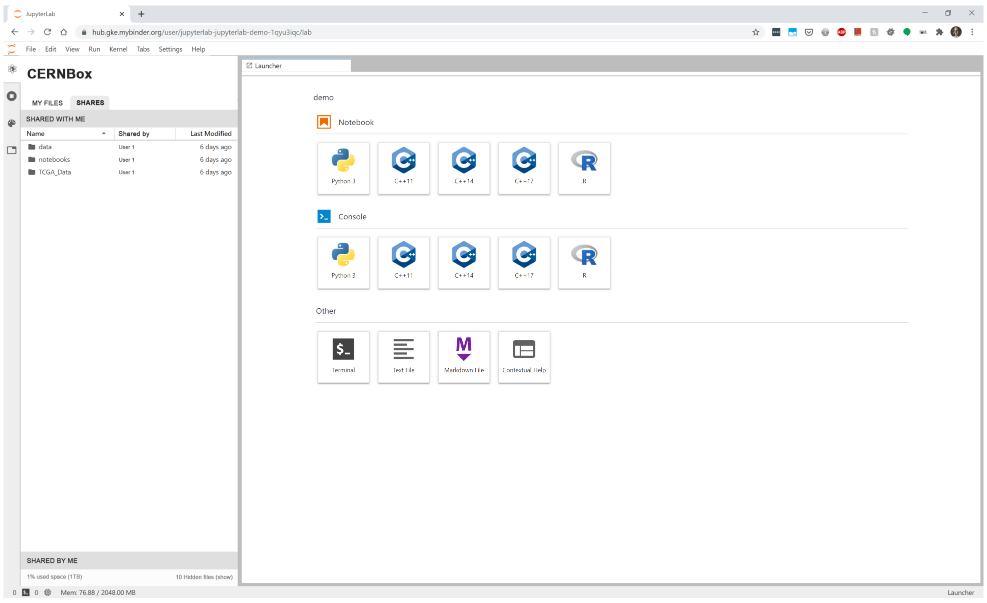
The integration is between Jupyter and EFSS services is achieved by a JupyterLab extension which uses CS3APIs to communicate with local EFSS and, indirectly, via OCM, with remote sites/environments. In this way access to federated resources is provided directly from the Jupyter service, thus enabling researchers to perform data science tasks in a federated cloud. This allows a user to move back-and-forth between a local file system and a federated Science Mesh within one cohesive platform shared by users and with easy access to the science tools.

From a user perspective installation and configuration of stand-alone applications will no longer be needed. Instead a simple browser will suffice to access the complete science environment.

Technical description can be found below.

3.2.1 Jupyter extensions

Cs3api4Lab is a JupyterLab extension that allows the integration with the IOP platform via the CS3APIs for basic file operations (retrieval, save) as well as for sharing functionalities.



Name	Shared by	Last Modified
data	User 1	6 days ago
notebooks	User 1	6 days ago
TCGA_Data	User 1	6 days ago

Figure 5. Prototype of the extension running inside JupyterLab

The plugin consists of two parts:

- **backend** – written in python, it is used for connecting Jupyter to the IOP platform. It replaces the original Jupyter ContentsManager and Checkpoints components and it adds a new Rest API for additional functionalities (i.e sharing). The backend establishes connections using the gRPC protocol and uses the methods provided by the CS3APIs.

- **frontend** – written in TypeScript, React, HTML and CSS, it replaces the default JupyterLab file manager and adds new interface elements for additional share functionalities. This includes extending the file manager by adding a shared by/with tab, adding sharing buttons and entries in the context menus and adding new modal windows to display file information and sharing status.

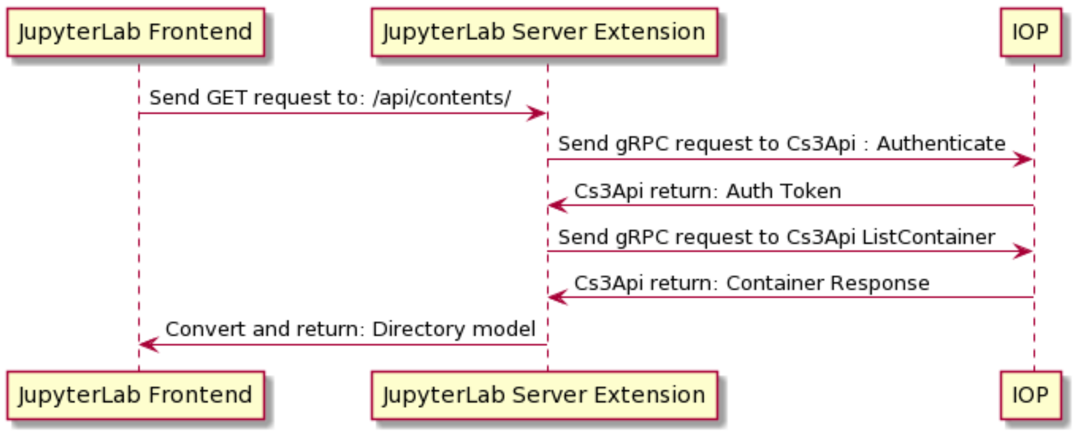

JupyterLab Frontend
JupyterLab Server Extension
IOP
Send GET request to: /api/contents/
Send gRPC request to Cs3Api : Authenticate
Cs3Api return: Auth Token
Send gRPC request to Cs3Api ListContainer
Cs3Api return: Container Response
Convert and return: Directory model

Figure 6. An example of reading a directory from an IOP using the JupyterLab extension

The back-end provides the following REST endpoints for integration with the front-end:

API for content operations:

- **/api/contents/{path} (GET)** – This endpoint returns a model for a file, notebook or directory.
  The directory model contains a list of models, the notebook model contains a notebook data in JSON format, the file model contains file data in a text or base64 format. This endpoint uses the CS3API methods: Authenticate, ListContainer, InitiateFileDownload.

- **/api/contents/{path} (PUT)** – Saves a file or a notebook in the specified file path. For the notebook model, the content key is saved in a JSON format. For the file format, the key context is kept unchanged. If the JSON body contains "{"copy_from":"[path/to/]OtherNotebook.ipynb" } ", the file copy operation is performed. This endpoint uses the CS3API methods: Authenticate, ListContainer, CreateContainer, InitiateFileUpload, InitiateFileDownload

- **/api/contents/{path} (PATCH)** - Renames a file, a notebook or a directory without re-uploading content. This endpoint uses the CS3APImethods: Authenticate, ListContainer, Move

- **/api/contents/{path} (POST)** - Creates a new file, notebook or directory in the specified path. Server always decides on the file or directory name. This endpoint uses the CS3API methods: Authenticate, ListContainer, CreateContainer, InitiateFileUpload, InitiateFileDownload

- **/api/contents/{path} (DELETE)** - Deletes a file or directory in the given path. This endpoint uses the CS3API methods: Authenticate, Delete

API for checkpoints operations:

- **/api/contents/{path}/checkpoints (GET)** - Lists checkpoints for a file.

- **/api/contents/{path}/checkpoints (POST)** - Creates a new checkpoint for a file.

- **/api/contents/{path}/checkpoints/{checkpoint_id}** (POST) - Restores a file from a checkpoint.

- /api/contents/{path}/checkpoints/{checkpoint_id} (DELETE) – Deletes a checkpoint for a given file.

API for share operations:

- **/api/cs3/shares (POST)** - Creates a share. This endpoint uses the CS3API methods: Authenticate, CreateShare.
- **/api/cs3/shares (DELETE)** - Deletes a share. This endpoint uses the CS3API methods: Authenticate, RemoveShare.
- **/api/cs3/shares (PUT)** - Updates a share. This endpoint uses the CS3APImethods: Authenticate, UpdateShare
- **/api/cs3/shares/list (GET)** - Gets all given shares. This endpoint uses the CS3API methods: Authenticate, ListShares
- **/api/cs3/shares/received (GET)** - Gets all received shares. This endpoint uses the CS3API methods: Authenticate, ListReceivedShares.
- **/api/cs3/shares/received (PUT)** – Updates received share. This endpoint uses the CS3API methods: Authenticate, UpdateReceivedShare
- **/api/cs3/shares/file (GET)** - Returns grantees for file. This endpoint uses the CS3API methods: Authenticate, ListShares

3.2 Open Data Systems

Open Data Systems help the ScienceMesh users to handle structured data and use it for activities typically associated with such data (archive, deposit and publish). The following interactions should be possible:

1. Users should be able to create structured datasets out of unstructured data present inside Science Mesh or about to be ingested into Science Mesh – that is, delimit what data and in what ordered format (typically directory trees) is included in the dataset about to be defined. The project will leverage on the RO-Crate specification its structured data format.

2. Users should be able to manipulate the metadata. For this purpose, the Describo [13] tool provided by UTS will be a starting point, closely collaborating with the authors to explore its capabilities and integration.

3. Users should be able to augment the structured dataset with appropriate metadata, either user generated or harvested from environment data. Here, too, Describo acts as the design pattern.

4. Users should be able to perform scientific workflow actions (publish, archive, etc) based

[13] https://eresearch.uts.edu.au/tools/describo/

on RO-Crates[14], either those already present in the system or RO-Crates that must be generated on demand, to fulfil the task. ScieboRDS [15] is our design example here, and WP4.2 will work closely with the ScieboRDS team to maximise synergies. Some of these workflows will start inside the ScieboRDS application environment (e.g., "a scientist wants to publish a paper with a certain dataset attached"), hand over to the Describo environment ("select the relevant data to be crated up") and then hand back to ScieboRDS ("here's the crated and metadata-augmented data the scientist wanted to publish").

In light of these functionalities, the following technical considerations apply:

In the general situation, users use the web version of Describo tool as a plugin to describe files or data with metadata, in line with the following workflow:

1. Select a directory in the cloud system (ownClud, nextCloud, etc).

2. ScieboRDS runs the Describo plugin and gives a path to the directory, authorization token, a profile information.

3. Describo plugin opens an internal editor, check if the RO-Crate file exits, opens an existing file.

4. The user specifies the data by filling in the default form fields (based on the profile) and adding additional fields to the form based on the RO-Crate specification.

5. Describo plugin creates or updates the RO-Crate file with users metadata and stores it in the IOP platform.

6. ScieboRDS open the RO-Crate file stored by Describo plugin and sends it to the external metadata systems (e.g. Zenodoo, etc).

In this workflow, Describo plugin uses the Cs3 API methods:

- **ListContainer** - gets the directory structure

- **InitiateFileUpload** - reads the RO-Crate file and other files

- **InitiateFileDownload** - write the RO-Crate

- **Stat** - check file status for the file is exits operation

and other methods for basic file operations, revalidate authentication token, etc

The diagram below shows the general Describo plugin structure and planned components. The web version of Describo has two general parts:

- Frontend - run on the user web browser, visualizing user interface, get input from the user, crate RO-Crate file, preview created crate file, profile selection/customization

- Backend - run on the server, connecting to the IOP and ScieboRDS platform, convert data

[14] https://eresearch.uts.edu.au/tools/describo/
[15] https://www.research-data-services.org/de/

from an external system and send to the frontend part

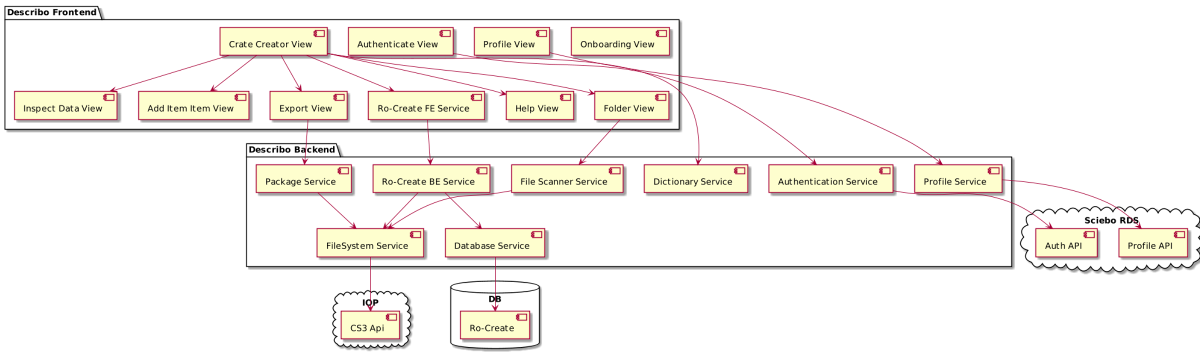


Figure 7. The general diagram of Describo plugin components

3.3 Collaborative Documents

Collaboratively editing a document, usually from within a web browser, requires the application to be able to synchronize the document data to and from the underlying storage. This means that the application needs to be able to read and write the document no matter whether it resides on the user's sync and share system or if it has only been shared with him by another user.

While such an application can achieve this by directly using the CS3APIs, another simpler approach is based on the WOPI protocol (cf. 2.4) as it natively supports high-level endpoints to manage file locking and document collaboration.

A typical workflow to open a document in a WOPI-enabled application is as follows:

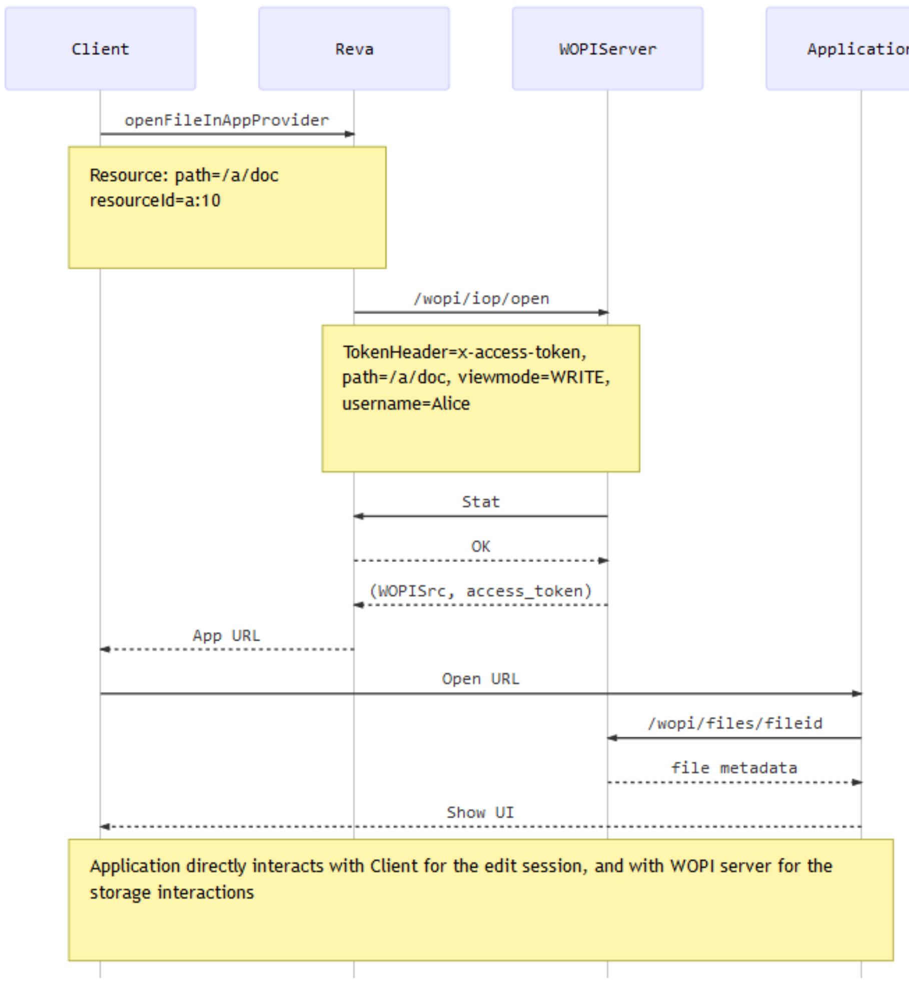


Figure 8. Work-flow to open a document with WOPI

API for open:

- **openFileInAppProvider (GET)** - Given a file reference (either in a local storage or in a remote share), generate an application URL to open it. A view mode parameter specifies whether the file is to be opened in view-only mode, read-only (download allowed) mode, or full editing mode. Currently, only WOPI-enabled applications are supported, whereas in the future other applications are foreseen to be integrated within this workflow.

3.4 On-demand data transfers

On-demand data transfer fulfils the use-case where it is not possible to extend processing capabilities to remote sites. Instead the data, typically in the TB size range, needs to be transferred close to where the processing capabilities are. Two modes of operation for this will be made available within the Science Mesh: 1) transfers initiated and handled directly (point-to-point) between EFSS services; 2) transfers initiated by EFSS service but offloaded to a secondary service (such as FTS).

The first file transfer operation mode implemented in the Science Mesh is the direct file transfer between EFSS storage back-ends. The actual data transfer is performed through the WebDAV implementation of the EFSS and is orchestrated by a higher-level service.

A file transfer interface in CS3APIs specifies three methods and a transfer object interface containing the transfer ID and status. The three CS3API file transfer methods are:

- CreateTransfer – creates a transfer and returns the transfer object;
- GetTransferStatus – returns the status of a previously created transfer object;
- CancelTransfer – cancels a previously created, not yet finished, transfer object.

The transfer workflow includes a transfer initiation by the *CreateTransfer* method and subsequent queries and actions based on the transfer ID: *GetTransferStatus* or *CancelTransfer*.

The next section explains more about how this will be implemented for direct file transfer using rclone as the transfer orchestrator.

3.4.1 RClone based transfers

Rclone [16] is a command line program to manage files on cloud storage. It is mature, open source software with an active support community. A prominent feature of rclone is its ability to orchestrate (coordinate) WebDAV based file transfer between remotes. WebDAV protocol support is common for any EFSS. This makes rclone our primary candidate for orchestrating direct file transfer.

Two main interfaces play a role in direct file transfer (figure 8). At the source EFSS/IOP the CS3APIS support initiation, requesting transfer status, and cancellation of a file transfer. A file transfer can be seen as a special share and this has to be accepted by (the user at) the destination. To support this the current OCM specification will be extended with a special data transfer protocol type which can be used in the OCM share request. For accepting (or optionally declining) the transfer the OCM notification post mechanism is used. OCM notification is also used for cancellation of a running transfer job from the receiving end.

[16] https://rclone.org/

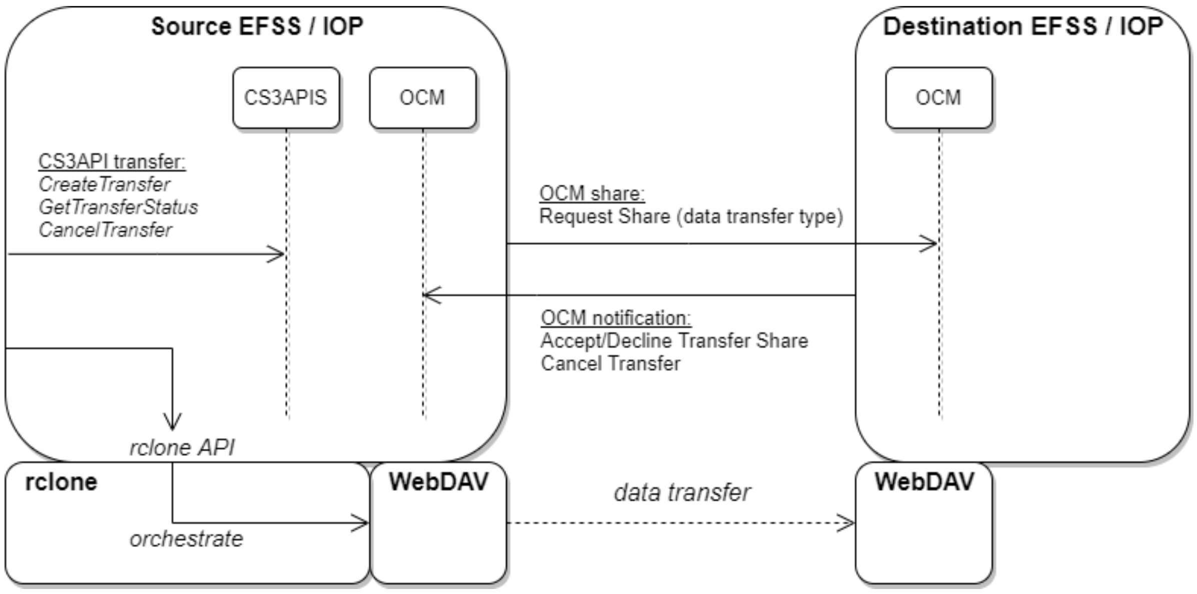


Figure 9. Direct file transfer between source and destination EFSS/IOP is supported through CS3APIS and OCM interfaces, rclone, and WebDAV.

The IOP delegates (through the rclone API) transfers to rclone which runs as a separate service and acts as the orchestrator (figure 8) to coordinate the actual data transfer. Rclone uses the WebDAV implementation of the EFSS to transfer the data and in the process, keeps track of running transfer jobs.

The IOP also keeps track of the data transfers itself independent of rclone, so users can keep track of their transfers.

4 EFSS integration with IOP

Interoperability of EFSS services developed by different vendors is key for ScienceMesh and in this section we describe the role of the Interoperability Platform (IOP) in facilitating this.

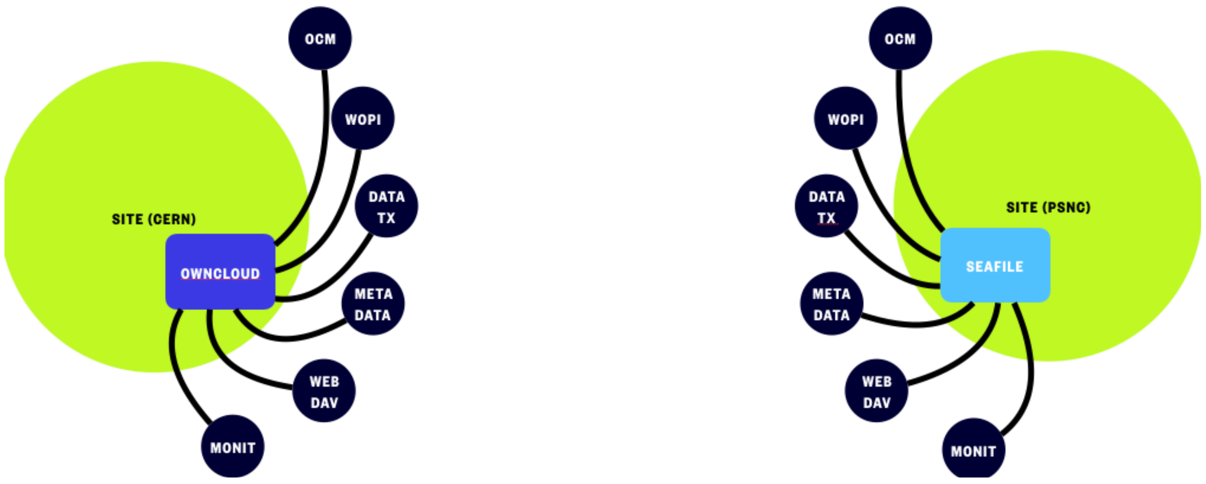


Figure 10. Ideal adoption of the APIs from the vendors

Figure 9 shows an ideal scenario where the vendors provide built-in support for the APIs required by the Science Mesh. However, direct support and integration from EFSS vendors is a far-fetched goal in a short time-scale because it requires the functionality to be re-implemented for each EFSS platform in a consistent and coordinated way to ensure full interoperability across different vendors. To fast-forward the integration of the necessary functionality into EFSS in a fair and consistent way and to enable new sites joining in short time-scale, the **Inter-Operability Platform (IOP)** is introduced as middleware component. IOP may be deployed easily at the sites as an additional service and configured to be connected to the local EFSS. Once a connector between IOP and EFSS is developed, new application workflows and ScienceMesh services may easily be built or evolved on top of interoperable APIs.

The IOP consists of several software components packaged into a single bundle (using Kubernetes Helm charts). The main components are:

- Reva[17]: an open source platform to connect EFSS, storage and application providers in a vendor-neutral and inter-operable way. Reva implements the promoted protocols in Science Mesh and is the reference implementation of the CS3APIs.

- WOPIServer[18]: an open source and vendor-neutral Web-application Open Platform Interface (WOPI) gateway for EFSS systems, to integrate several Office Online platforms including Microsoft Office Online and Collabora Online.

The next figure shows the IOP deployed on two sites. The IOP exposes the interoperable protocols and funnels them over a single API to the EFSS using the CS3APIs. This deployment model has several advantages:

- It is feasible to implement in the given time span of the project (3 years)

- It allows to gradually transition protocols and API implementations to vendors in a smooth and controlled manner.

[17] https://reva.link/
[18] https://github.com/cs3org/wopiserver

- The IOP is vendor-neutral and owned by the community, reducing the risk or having the IOP targeting ad-hoc developments for a particular vendor.

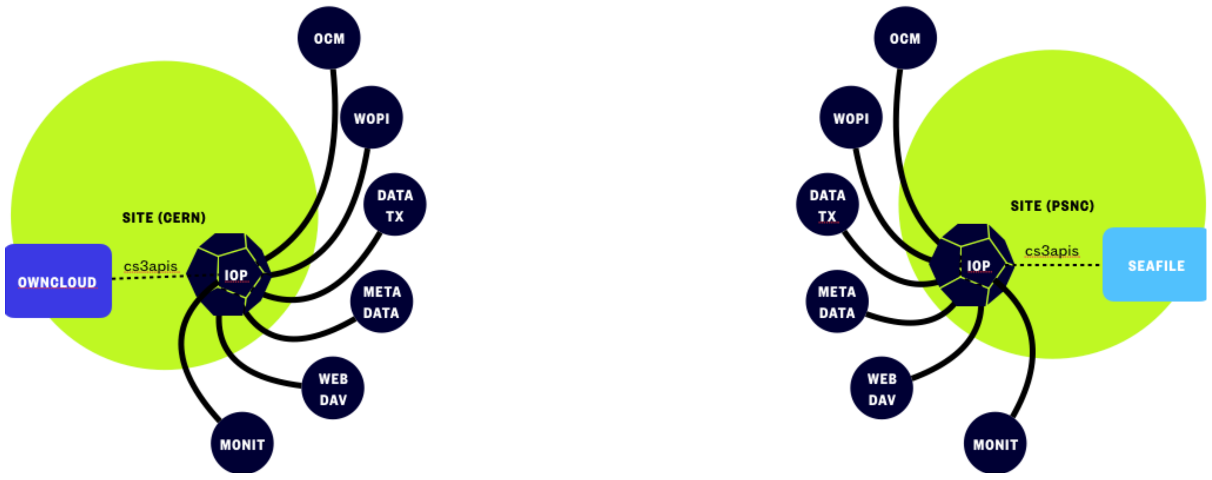


Figure 11. Strategic deployment of IOP and vendor integration

5 Peer-to-Peer architecture integration with IOP: Cubbit

Cubbit [19] is a hybrid cloud architecture consisting of a network of P2P devices (the "Swarm") coordinated by a central optimization server. Differently from a centralized cloud provider architecture, a file uploaded to Cubbit is not stored on a server but is encrypted and split up into several shards that are distributed over the swarm. In the same way, when a file is downloaded from Cubbit, all the shards are recollected, decrypted, and joined back together to retrieve the original payload.

Due to a different architecture, a seamless federation between the Cubbit service and other ScienceMesh nodes requires a specific layer that acts as a "translator" between Cubbit's P2P architecture and more centralized storage architectures.

An internal integration strategy was developed by Cubbit based on the API layer to assure transparent and portable integration. The first set of endpoints was developed and deployed to allow basic file upload and download:

- **/files (POST)**: This endpoint allows a user to upload a file to the Cubbit network. Since Cubbit allows for encryption and redundancy, a user can specify them as options in the request body. If encryption is enabled, an encryption key has to be passed as a mandatory parameter.

- **/files/{file_id} (GET)**: This endpoint allows a user to download a file with the specified file_id from the Cubbit network. If the uploaded file had been encrypted during the upload, the user must give the file key as a body parameter for this request.

[19] https://www.cubbit.io/

Next, the IOP connector for Cubbit was implemented using the simple endpoints described above. The connector allows the Cubbit service to consume OCM shares from ScienceMesh node.

The connector leverages on the following CS3APIs:

- **OcmCoreAPI.CreateOcmShare**: this endpoint allows other partners to create a Share with Cubbit users
- **OcmAPI.ListReceivedOCMShares**: this endpoint allows listing all the received shares

The implementation is based on a basic NodeJS server that is responsible for accepting the OCM requests forwarded by the Reva gateway. These requests are then validated and processed by a Coordinator Cubbit microservice which redirects as appropriate.

The CS3APIs repository has been integrated as a submodule of the Cubbit software stack — this helps the interface (proto-definitions) to stay up-to-day with the CS3APIs upstream.

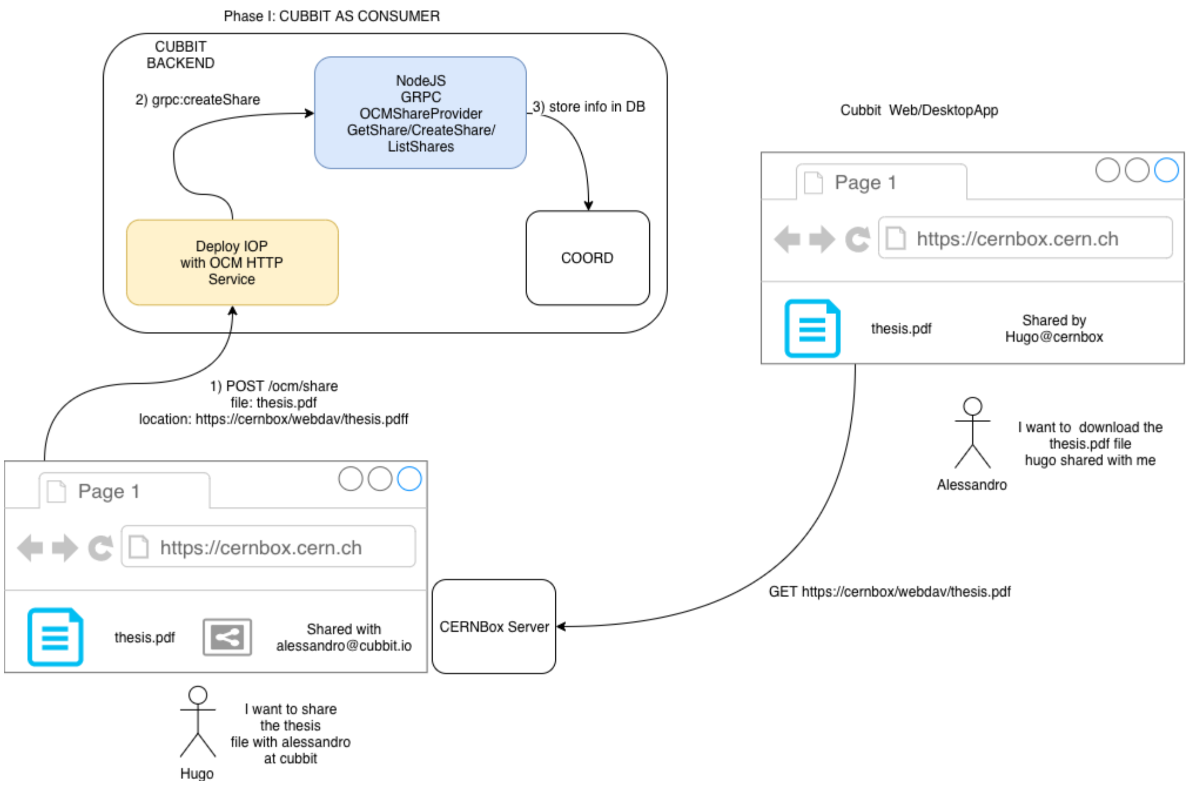

Phase I: CUBBIT AS CONSUMER
CUBBIT BACKEND
2) grpc:createShare
NodeJS GRPC OCMShareProvider GetShare/CreateShare/ ListShares
3) store info in DB
Deploy IOP with OCM HTTP Service
COORD
1) POST /ocm/share file: thesis.pdf location: https://cernbox/webdav/thesis.pdff
Page 1
https://cernbox.cern.ch
thesis.pdf
Shared with alessandro@cubbit.io
CERNBox Server
I want to share the thesis file with alessandro at cubbit
Hugo
Cubbit Web/DesktopApp
Page 1
https://cernbox.cern.ch
thesis.pdf
Shared by Hugo@cernbox
I want to download the thesis.pdf file hugo shared with me
Alessandro
GET https://cernbox/webdav/thesis.pdf

Figure 12. CUBBIT deployment with the IOP

6 ScienceMesh Infrastructure APIs: Monitoring

General monitoring of all Science Mesh providers is implemented with Prometheus[20]. Prometheus is a royalty free, industry-proven time series database used to gather metrics efficiently from multiple targets (i.e., the various mesh providers). It offers a plethora of features for querying the gathered data, and many other services, and applications have direct support for Prometheus built in; this includes Grafana[21], a web service for presenting data in highly customizable dashboards. Due to it being entirely free and its wide range of features and support by 3rd party services and applications, Prometheus was chosen to build the foundation of the monitoring process.

The main endpoint used for monitoring is **/metrics**, which exposes various metrics in the OpenMetrics [22]format and is queried periodically by Prometheus. These metrics include information about the current number of users and groups in the Science Mesh, the total storage and several items related to HTTP I/O. Furthermore, system information is included in these metrics which can be used to perform basic health checks (e.g., it can be checked whether a mesh provider is running the correct version of *Reva).* By exposing all metrics at a single endpoint, only one query needs to be performed by Prometheus per mesh provider, which reduces the strain on the Prometheus system and the overall network traffic caused by the monitoring process.

6.1 Mentix

In order for Prometheus to know about the targets it should monitor, a service called Mentix (short for Mesh Entity Exporter) is used to query the necessary information about each mesh provider from a central database (GOCDB[23]) and export this to Prometheus. This service can also be used by other services which require information about each mesh provider; it exposes the following endpoints:

- **/mentix:** The main endpoint exposes information about each mesh provider, including its name, address, services offered by the providers and more.
- **/mentix/cs3:** This endpoint exposes the mesh provider information in a CS3 API compliant format.
- **/mentix/loc:** This endpoint exposes location information about each mesh provider which can be consumed by Grafana.

The following diagram illustrates how Mentix interacts, as part of a *central component*, with all other services and provides the necessary mesh information:

[20] https://prometheus.io/
[21] https://grafana.com/
[22] https://openmetrics.io/
[23] https://goc.egi.eu/

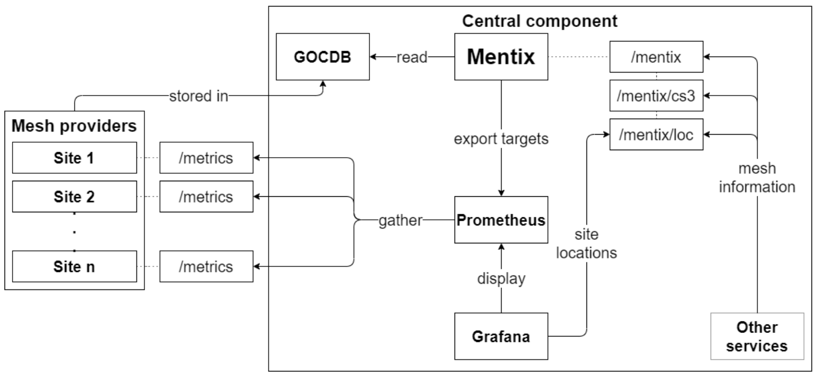
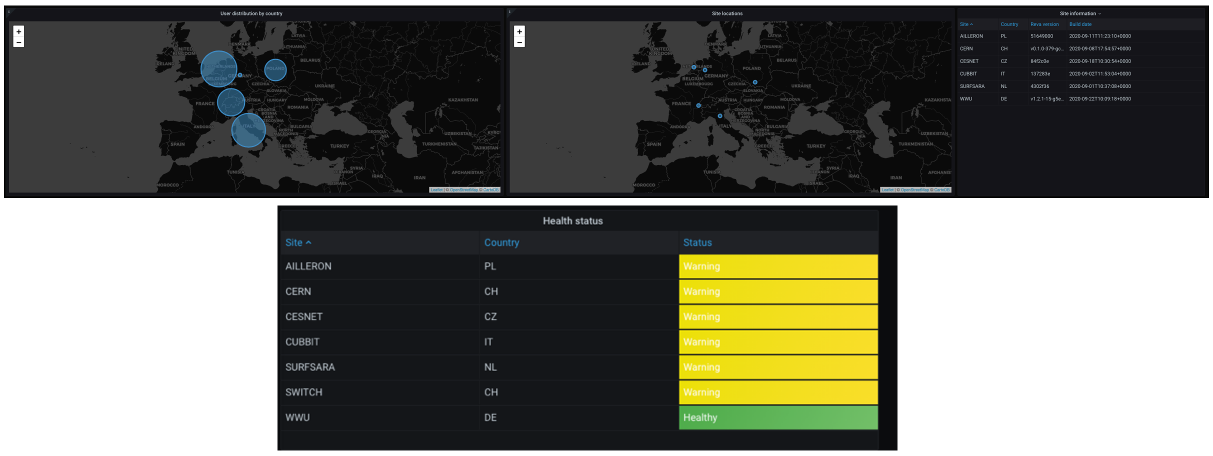


Figure 13. Central Component infrastructure (including Mentix)

While Mentix is not run by the individual Science Mesh nodes , it is an essential part not just in the monitoring process: For example, the mesh directory service, which allows a user to select its mesh provider when accepting a share invitation, also uses the Mentix endpoints to get the necessary information about all providers in the mesh.



Site information

Site	Country	Reva version	Build date
AILLERON	PL	51649000	2020-09-11T11:23:10+0000
CERN	CH	v0.1.0-379-gc...	2020-09-08T17:54:57+0000
CESNET	CZ	84f2c0e	2020-09-18T10:30:54+0000
CUBBIT	IT	137283e	2020-09-02T11:53:04+0000
SURFSARA	NL	4302f36	2020-09-01T10:37:08+0000
WWU	DE	v1.2.1-15-g5e...	2020-09-22T10:09:18+0000

Health status

Site	Country	Status
AILLERON	PL	Warning
CERN	CH	Warning
CESNET	CZ	Warning
CUBBIT	IT	Warning
SURFSARA	NL	Warning
SWITCH	CH	Warning
WWU	DE	Healthy

Figure 14. Monitoring dashboards

7 Conclusion

This document presents the initial definition of APIs and protocols for the Science Mesh federated infrastructure as well initial implementation of several application and infrastructure workflows.

We leverage on existing standards as much as possible and introducing new interfaces only where needed.

We expect the APIs and implementations to mature and as the next step the IOP connectors created in collaboration with major EFSS vendors.

8 Table of figures