FAIR-enabling deposit workflow design (D2.3)

Project	HORIZON-ZEN - EU research programme beneficiary depositing solution in Zenodo.
Work package	WP2
Deliverable no.	D2.3 (Public)
Authors	Lars Holm Nielsen (CERN) Alex Ioanndis (CERN)
Reviewers	Jose Benito Gonzalez Lopez (CERN)
Version	1.0
Date	2024-03-31
DOI	https://10.5281/zenodo.10901185

Funded by the European Union

Funded by the European Union under Horizon Europe grant agreement number 101122956

Table of contents

Funded by the European Union

Introduction

This report provides a detailed overview of the FAIR-enabling deposit workflow design for the HORIZON ZEN project. The overall goal is to **make it easier** for EU programme beneficiaries to comply with the related open science requirements in Horizon Europe as detailed in the D2.2 Content and curation policy for the Zenodo-communities. Specifically for the project, the goals of this deliverable is design the features to:

- Harmonize metadata curation among different EU Project Communities and ensure metadata complies with curation policy for the EU Open Research Repository.
- Provide end-users with actionable feedback on how to improve FAIRness of their research outputs.

The related expected results from the project proposal are:

- Deposit workflow for complying with Horizon Europe requirements (SO1).
- Automated curation checks to lower required curation efforts (SO2).
- Addition of discipline-specific metadata/vocabularies for a minimum of two disciplines (SO3).
- At least two existing FAIR data assessment tools integrated into the Zenodo depositing workflow providing actionable advice to end-users on how to increase FAIRness of deposited content (SO3).

FAIR-enabling deposit workflow: UX

A crucial requirement for the FAIR-enabling deposit workflow is to **simplify** the workflow for researchers to deposit FAIR data. The overarching objective is to enable reuse of research outputs by enhancing the quality of the deposited research materials. This means that the design is as much about the **user experience and automation** as it is about measuring FAIRness.

Approach: Small improvements for a large audiences, all research outputs
Our approach to the design of the FAIR-enabling deposit workflow is to value quantity over quality - i.e. we focus on:

- smaller improvements in FAIRness for a large audience, rather than larger improvements in FAIRness for a small audience.
- improvements which can benefit all research outputs when possible instead of only research data.

This means our key focus areas are:

Improve overall FAIRness of Zenodo
During the design process we've tested several FAIR evaluation tools. All of the tools are designed to probe the repository from the outside. To a large extent this is by design due to the FAIR principles themselves and because the tools are focusing primarily on testing metadata.

This means FAIR evaluation tools predominantly test capabilities of a given repository software rather than the individual research output given that the metadata has been provided to the repository.

> *Example: A repository usually does not register a DOI (at least in a discoverable way) until a given research output is published. FAIR principle F1, "(Meta)data are assigned a globally unique and persistent identifier" thus cannot be tested until the research output is published because the identifier and/or landing page does not yet exist.*

Thus, in line with our approach, we will firstly focus on running existing evaluation tools on a subset of records on Zenodo, and implement improvements for these, as this will improve overall FAIRness for all research output.

Metadata curation
With the overall improvement of the FAIRness of Zenodo itself, we next need to focus on obtaining high quality metadata for each individual research output. As an example, Zenodo already today allows for easy auto-completion of authors while ensuring each author has a persistent identifier.

Nobody likes to fill endless forms, thus the main approach is to simplify as much as possible the deposit workflow. We will improve Zenodo in two areas:

- Automated curation checks - through automated checks we provided targeted guidance to researchers and reviewers as well as help them provide important metadata.
- Subject information - the initial key driver for automating curation beyond basic checks is understanding the subject of a research output as this can be used to suggest e.g. disciplinary metadata/vocabularies as well as use community specific testing tools.

Data curation
With improvements in overall FAIRness and metadata, the next focus is on the data itself. The simplest improvement on the repository side would be to inform users if they are not using open/scientific file formats. Open/scientific file formats is a first most basic level of FAIRness by increasing the chance of being able read the data in the future. This is best addressed already from the beginning of a research project through e.g. the use of a data management plan. However providing a last check in the repository will at the bare minimum help to educate users, as most users are unaware of the problems in using e.g. proprietary file formats.

Improve overall FAIRness of Zenodo

The RDA FAIR Data Maturity Model is the reference model establishing a set of indicators and maturity levels for testing FAIRness which can be implemented by multiple tools. As mentioned earlier, *given the metadata has been provided in the repository*, most of the indicators test capabilities of the repository software rather than the record itself. As an example, out of 16 metrics in a tool, only 2 provides insight into the record itself (see “Appendix”). Thus the first approach is to run multiple tools on Zenodo to evaluate and identify improvements of Zenodo itself, rather than at an individual record level.

FAIR evaluation tools

We have selected the below a set of tools listed on FAIRassist.org that can perform automated tests (all semi-automatic and manual tests have been excluded):

- F-UJI
- FAIR Checker
- FAIR Enough
- FAIR EVA
- FAIR Evaluator
- OpenAIRE Validator - FAIR assessment

Most of these tools are either already or in progress of adopting the RDA FAIR Data Maturity Model for testing.

Repository issues

Initial testing has shown that most of the identified issues for the repository software are related to having links to files, size and their type information as well as some missing provenance and access right information which is available in the metadata.

The key improvements identified via tests with F-UJI, FAIR Evaluator and FAIR Checker:

- Implement support for FAIR Signposting.
- Integrate disciplinary-specific fields into embedded JSON-LD.
- Integrate file information into metadata.

Integration of third-party tools in deposit workflow

Testing unpublished records
Currently, none of the above FAIR evaluation tools provide means to easily test FAIRness of an unpublished record. As shown in the appendix, it’s possible to grant a FAIR evaluation tool access to a restricted draft page on Zenodo, however the tool will score lower and raise issues

which are no longer relevant once published. Similarly nearly all the other tools will score almost identical for all Zenodo records except for a few metrics.

Specificity of guidance
The output results from the FAIR evaluation tools are not consistent and use language focused on e.g. semantic terms rather than something understandable by an end-user. In addition, while the tools can e.g. identify an author is missing a persistent identifier, they don't pinpoint the exact location of the problem (e.g. it will specify you should use person identifiers, but not point to if a specific author in the creators or contributors field is missing the persistent identifier.

Integration of existing FAIR evaluation tools
We understand why the FAIR evaluation tools have been designed in the way they are, and the tools themselves are very useful for testing Zenodo as a repository. However, we do not see a meaningful method by which we can integrate the current generation of FAIR evaluation tools into the deposit workflow to help users increase FAIRness of their deposited material due to only a few metrics providing any useful insight for end-users, the inability of testing unpublished records and the inability to pinpoint issues directly for the end-user. It is possible to integrate third-party FAIR evaluation tools, but the information provided will serve little to no value for end-users nor will it help increase actual FAIRness of the deposited material.

Alternative
We are still convinced that the development of interfaces for third-party FAIR evaluation tools to integrate directly into the deposit workflow will eventually allow future improvements to the FAIR tools to integrate. We therefore propose to develop the file format checks (one of the FAIR metrics which will yield different results per record) as a third-party application to demonstrate the concept and enable future integrations. The design is detailed in the Data Curation section and the feature is extracted from the F-UJI tool itself. This will allow demonstrating how a third-party application can provide direct input into the repository on identified issues and how to fix these for an end-user.

Funded by
the European Union

Metadata curation

Metadata is created and curated in Zenodo in two places:

- During the submission by the researcher (see figure 1)
- During the review of the submission by the community curator and researcher (see figure 2)

The following sections refer to the **deposit form** (figure 1) and **review request** (figure 2).

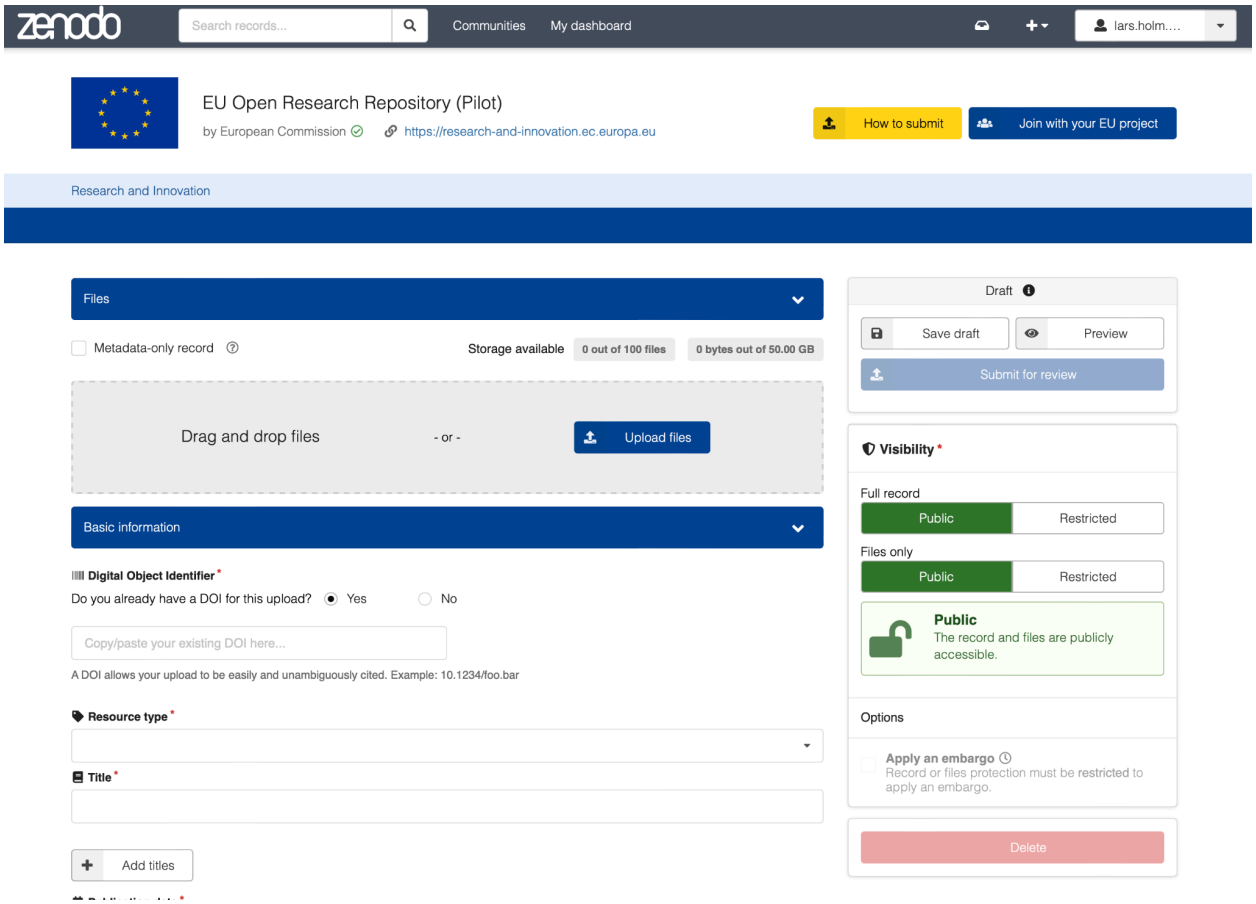

Figure 1: Deposit form (submission by researcher)

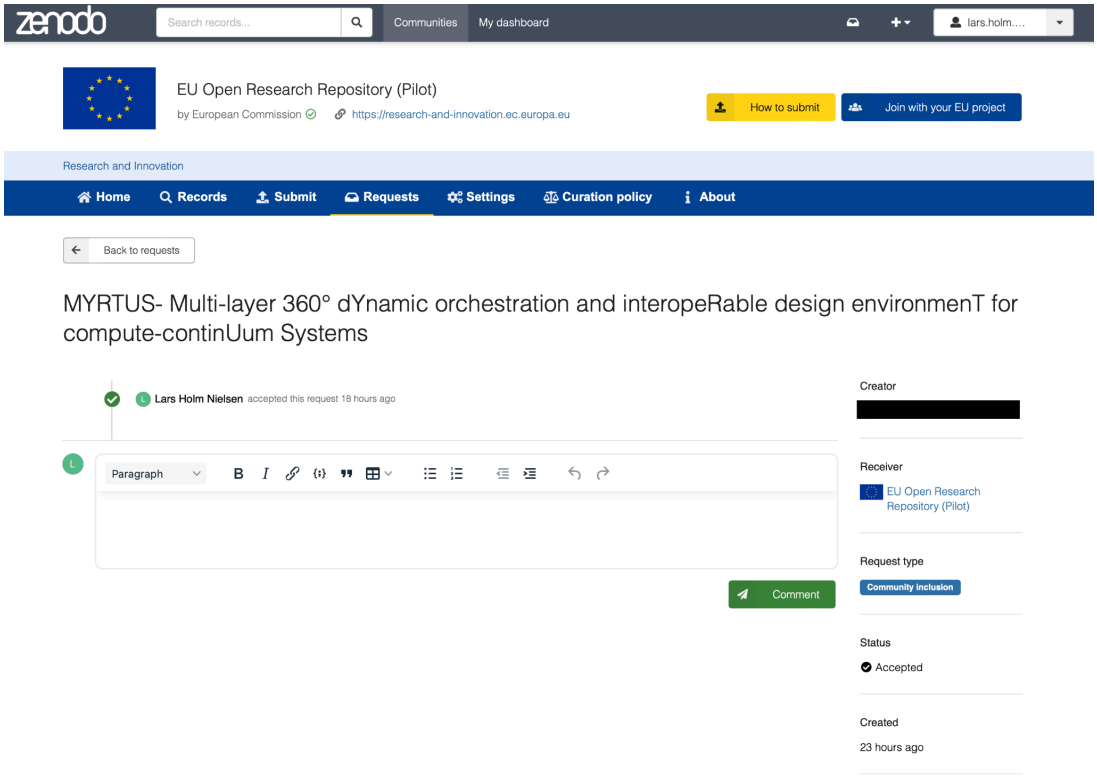

Figure 2: Review request (allows researcher/curator to communicate and edit the record)

Automate and harmonize curation

EU Open Research Repository relies primarily on programme beneficiaries themselves to curate their metadata by allowing projects to set up EU Project Communities in which they can control. A key issue for the EU Open Research Repository is then to ensure all the different EU Project Communities curate information in a harmonized manner and in accordance with the curation policy.

Curation checks

The following automated curation checks must be performed on new research outputs to ensure the metadata record complies with the related open science requirements in Horizon Europe:

- Require:
  - Record is linked to a license (see below for further checks)
  - One of either:
    - Record is linked with at least one grant from the European Commission.
    - Record is linked with the European Commission as a funder AND project name, project acronym and project number is provided.
  - If resource type is a journal article:
    - Record provides journal information.
- Warn:

    - Missing persistent identifiers:
        - Persistent identifiers for authors/contributors are missing.
        - Persistent identifiers for affiliations are missing.
    - Access issue:
        - Visibility is not public (required for journal articles, required for data with exemptions).
    - Possible incompatible license issue:
        - Journal article is not CC-BY.
        - Book/book section is not CY-BY, CC-BY-NC or CC-BY-ND.
        - Any other resource type is not an CC-BY, CC0 or OSI approved license.
- Recommend:
    - Related identifiers are missing.

The checks must be performed on all deposits in the project community and on direct deposits to the EU Open Research Repository.

Each check can provide one of the following status:

- Pass
- Recommend
- Warn
- Fail

Examples:

- Fail: A research output missing grant information would not allow publishing until the grant information was provided.
- Warn: A journal article deposited with restricted visibility should warn the researcher this is not compliant with the Horizon Europe open science requirements.
- Recommend: A research output without related identifiers should recommend (i.e. inform) the user that this is best practice and provide examples how you can link (e.g. to link data and publication).

User interface

The curation checks will initially be integrated into the review request interface (figure 2). This is so that both researcher and community reviewer can see the result of the checks. This also means the checks are only performed on submission to the community, and not on edits after the record is already in the community. Ideally a researcher should already in the deposit form see the information from these automated checks and all edits should also be subject to it. This will for now only be implemented if the resource consumption on the project allows for it.

The review request interface will be adapted with a new status check feature which will allow for multiple status checks to be defined on a community:

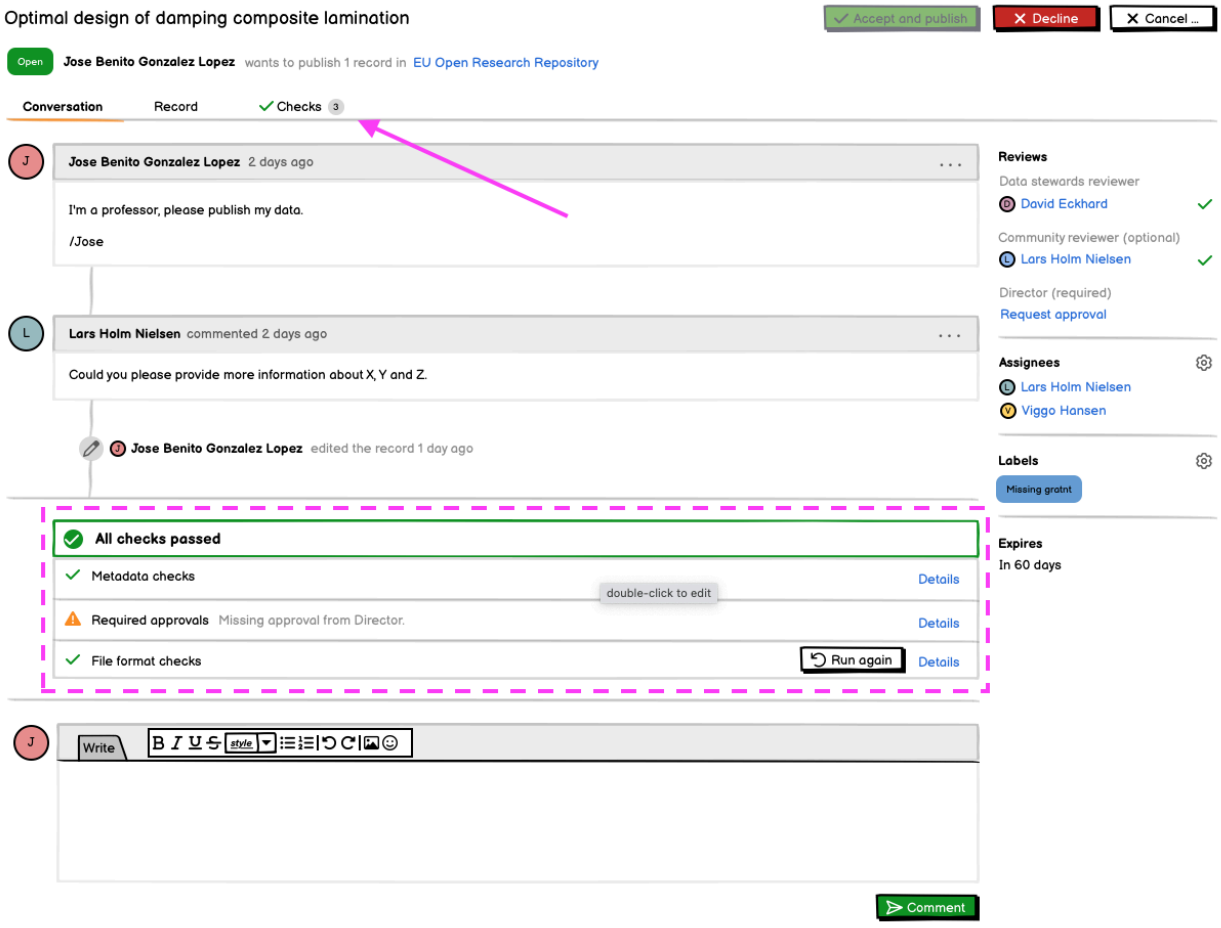

Figure 3: Proposed extension to the review request (new parts marked with pink)

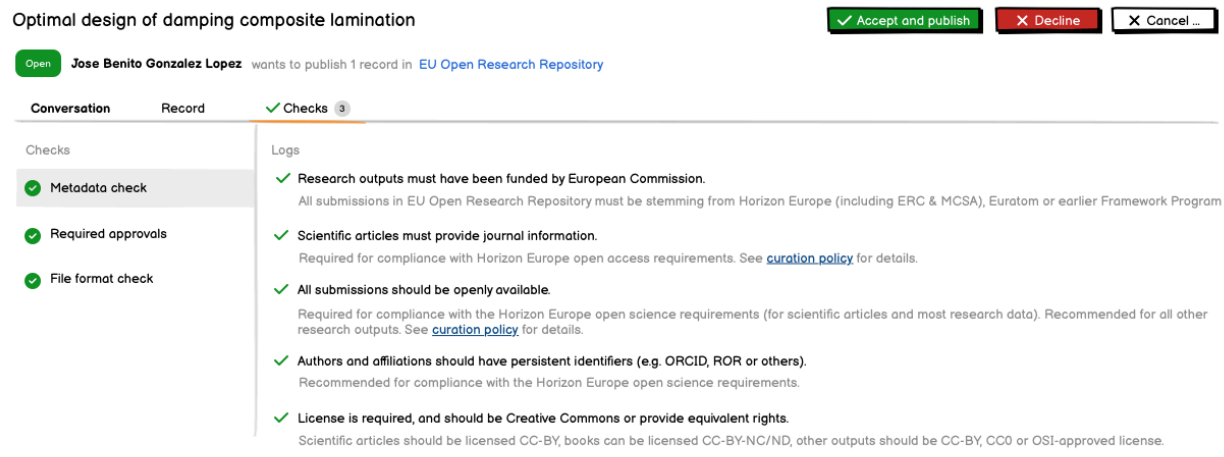

Figure 4: Proposed new tab "Checks" in the review request

Future extensions

Several extensions can be envisioned to this interface to allow EU Project Communities to build their own curation rules in particular for disciplinary-specific metadata.

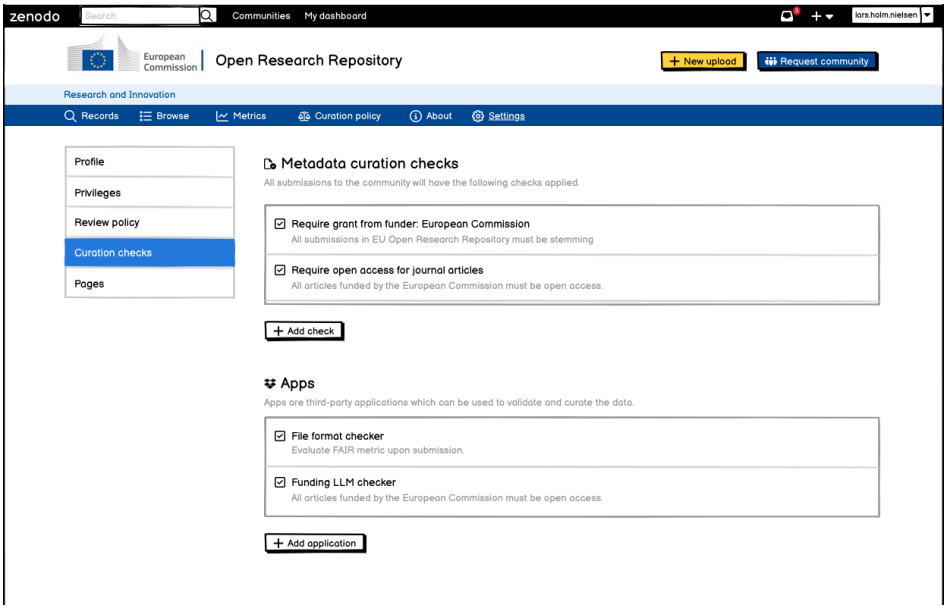

Figure 5: Unplanned interface for communities to edit their review rules

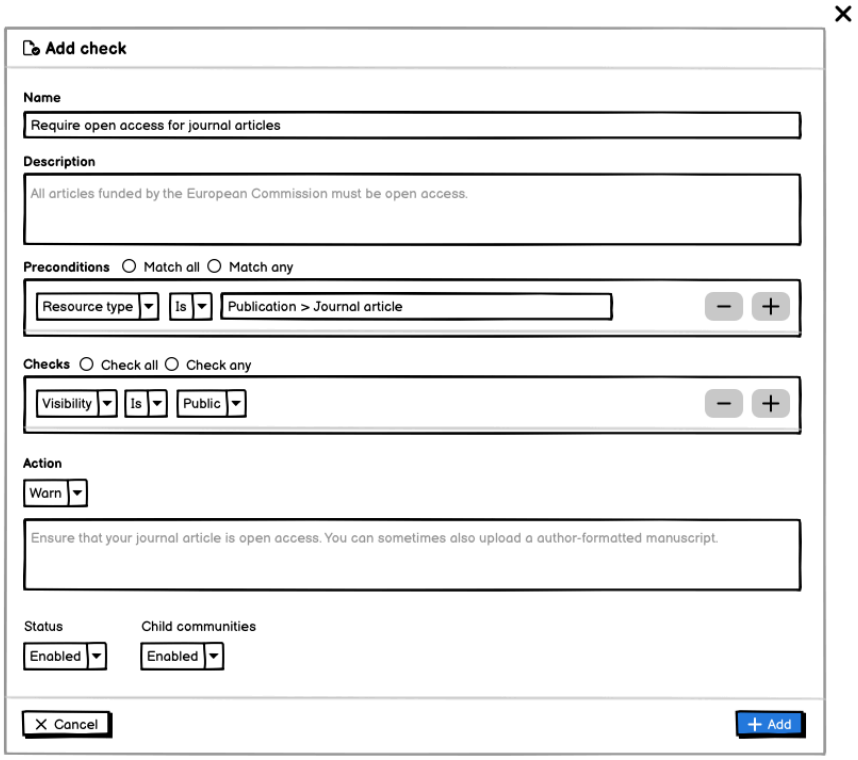

Figure 6: Unplanned interface for communities to create new curation checks

Subject information for existing records

Currently only a very limited number of records in the EU Open Research Repository have any subject information. The subject information is important for later to be able to run statistics and provide tailored recommendations for disciplinary-specific metadata. We plan to add subject information to existing records by:

- Project communities under the EU Open Research Repository will have their respective project's EuroSciVoc vocabulary subjects from the CORDIS database assigned to them.
- Records that are part of a Project community will have the community's subjects propagated to them.

This will allow:

- search and aggregation on the subjects of records under the EU Open Research Repository
- display of the subjects on the record page
- inclusion of the subjects in metadata exports (e.g. DataCite XML, Dublin Core, etc.)

The subjects will also be used to generate the EU Open Research Repository subject statistics. The statistics will show the distribution of subjects across all curated outputs that are part of the EU Open Research Repository and will be displayed on the EU Open Research Repository landing page.

Support for disciplinary metadata

Zenodo already supports disciplinary metadata for biodiversity (based on Darwin Core). We will extend this support in two different ways for two different domains:

- Life sciences, bioinformatics and biomedical science (BY-COVID, GDI):
    - Import of the Medical Subject Headings (MeSH) vocabulary allow
- Aquatic science (iMagine):
    - Definition of custom fields and vocabularies for aquatic sciences.

MeSH vocabulary support

The Medical Subject Headings (MeSH) vocabulary, maintained by the National Library of Medicine (NLM), is a comprehensive controlled vocabulary for the purpose of indexing journal articles and books in the life sciences. It serves as a thesaurus that facilitates searching and organizing information in biomedical and health-related fields.

Importing the MeSH vocabulary will enable the subject field in Zenodo to suggest terms from the vocabulary, in a similar way to how you can today suggest terms from EuroSciVoc (see Figure 7).

Funded by the European Union

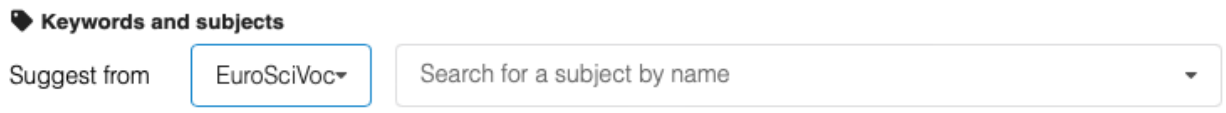

Figure 7: Subjects field in the deposit form

Overall, this will increase the browsing and discoverability of records for life sciences in Zenodo, e.g. for researchers searching for specific datasets.

Aquatic sciences metadata support

The iMagine project (#101058625) which is an early adopter of HORIZON-ZEN wants to share marine training datasets consisting of thousands of labeled images on Zenodo. The project has already shared 4 such datasets on Zenodo, and is interested in adding support for metadata specific to the datasets. While there's not a formally defined metadata standard which can be adopted, there's already been work performed on the iFDO (image FAIR digital object)[1] and in vocabulary[2].

The work in defining the needed custom fields as well as the vocabularies to import did not yet complete at the time of writing this deliverable. Through the project phase, we'll work with the project to define needed custom fields which should be added to Zenodo.

[1] https://marine-imaging.com/fair/
[2] https://vocab.nerc.ac.uk/

Data curation

Data curation is integrated in the user interface in a similar way as the metadata curation checks. This section mainly deals with third-party party tools as part of the curation checks. This will enable e.g. a EU Project Community to provide a disciplinary-specific tool which can be used to validate metadata and data submitted to the community.

Third-party tools integration

Figure 8 below provides an overview over how the third-party tools are integrated into the deposit workflow. The integration is achieved through notification sent from Zenodo to the third-party tools. After receiving the notification, the third-party tool takes over can do the following:

- Update the status of the request on Zenodo (e.g. in progres/done as well as descriptive information to the end-user)
- Retrieve information about the unpublished record (metadata and data) in order to perform the checks.

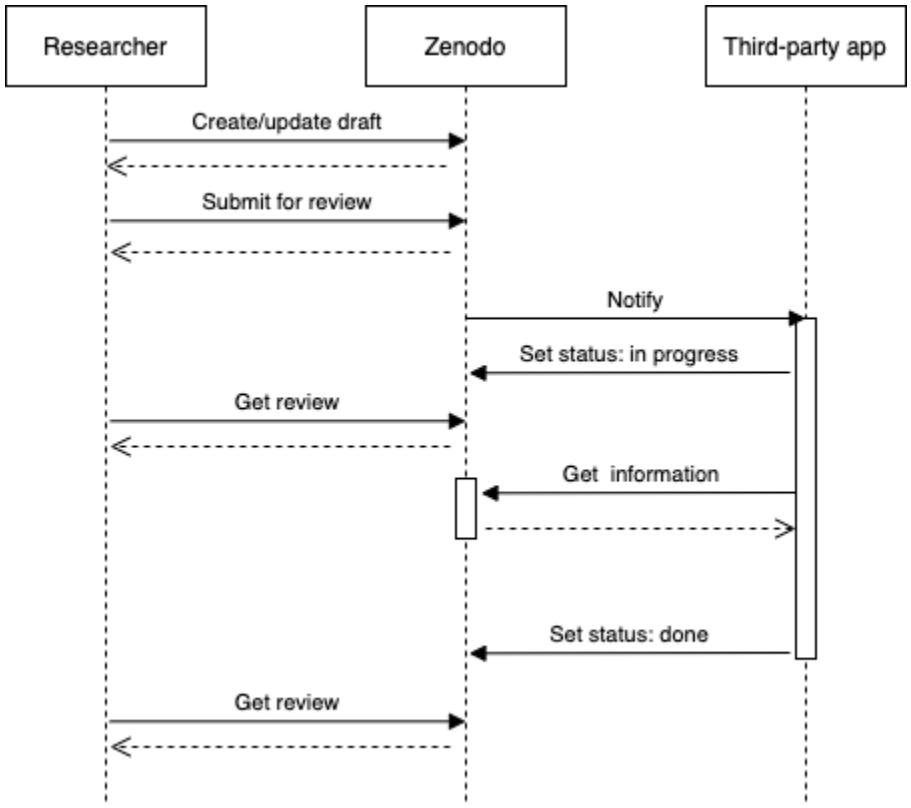

Figure 8: Integration of third-party tools.

Authentication

In order to ensure the third-party tools are as easy to implement as possible, it should be as stateless as possible to avoid the need for extra data storage or similar. Because the record is not yet published, the third-party app will need to authenticate to retrieve any information about the record and set the status. Similarly, Zenodo needs to authenticate against the third-party application so that the application can trust that a given event was received from Zenodo.

Third-party application
Zenodo will send an event to a preconfigured HTTPS endpoint and will sign the payload using a shared secret key which can be verified by the third-party application.

Zenodo
The third-party application will receive a short-lived access token from Zenodo and must similarly provide it either as URL parameter or in an HTTP header. The access token will be limited in scope to the specific draft record. Because the access token is included in the event sent to the third-party application, it must only be provided over HTTPS to ensure a secure exchange of the access token.

Application template

We will develop a simple application template based on Python/Flask/Celery which will demonstrate how a third-party application can be developed. The application will provide:

- Authentication
- Notification endpoint
- Background processing

Authentication
The application will provide a simple mechanism to sign and verify an access token based on an internal secret key. This ensures that the application is stateless and does not need to manage users.

Notification endpoint
The notification endpoint only needs to validate the request payload, and put a job into a background queue for processing.

Background processing
The background processing of a job ensures that the HTTP request for the notification does not get stuck/connection dropped in case it is a long running job. The background job will provide status updates to Zenodo throughout the processing.

Reliability

The primary concern by integrating third-party tools is providing information to end-users in Zenodo in case of unexpected events. Following failure scenarios should be handled:

- Notification could not be sent - e.g. the third-party application is not reachable or times out.
- Third-party application does not return - e.g. due to an error in the third-party application or network issues.

Errors
If Zenodo during event notification cannot notify the third-party application the user will be notified immediately in the interface.

Timeouts
If the third-party application does not provide a status update within the time limit of the access token, it will be considered as having errored out and the user will be notified accordingly.

Application registration

Third-party applications are integrated directly into Zenodo and are granted access to restricted content. For this matter, we will require applications to be registered in Zenodo in order to be able to receive event notifications. This will allow Zenodo to keep control over which applications are receiving events, and make it easier for communities to enable third-party applications.

Policy need
For now, we will not allow users to register new applications, but we can, on a case-by-case basis, enable applications for selected communities. Enabling users to register their applications requires developing a policy for which applications are allowed, associated requirements on the third-party application as well as to adapt existing policies such as our privacy policy.

File format check

To demonstrate the third-party tool integration, we will develop an application which can perform the file format check of deposited files. The check itself will be extracted from F-UJI source code which provides a list of open file formats, long-term file formats and scientific file formats.

Funded by the European Union

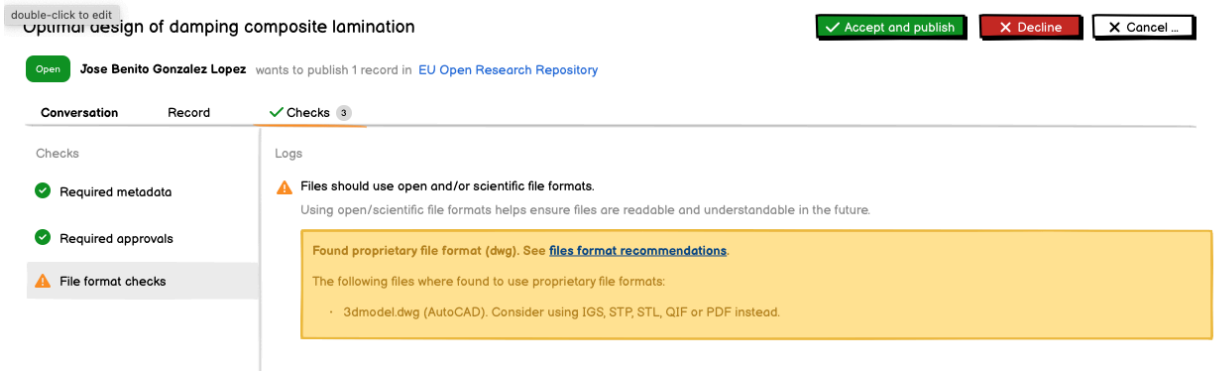

Figure 9: Integration of third-party tools.

Implementation

The design described in the preceding sections is implemented through an interface which can easily be used in the future to implement further automated curation capabilities. The key ingredients are:

- Request checks - used to display information on the request page to users about checks performed on a given record being submitted to a community.
- Webhooks - used to notify third-party applications about events in Zenodo, which can then use the request checks to provide information to users.
- Metadata rules - which implement the rules to be checked on a given community.

Request checks

Request checks are used to display the status of the metadata and file format checks to an end-user on the request page. It provides a generic interface by which other types of checks can be integrated in the future.

A community defines a check suite. Once a draft review request is submitted to the community, the community's check suite is used to initialize individual request checks on the request (e.g. metadata check and file format check). A request check can have multiple check runs which are updated accordingly. A check run stores the state and output from a given run and is used to render the concrete information to the user shown in e.g. figure 4.

The check run for the file format check is updated by the third-party application via the REST API (see figure 8), while the check run for the metadata check is updated internally.

The community check suite is constructed based on enabled internal checks as well as the installed applications.

Metadata rules

The metadata check is implemented through metadata rules. A metadata rule besides descriptive information contains the following properties:

- Condition - a complex object used to describe any preconditions required to pass for the rule to be checked.
- Test - a complex object describing the test to perform on the metadata.
- Action - a complex object describing the action to be performed (currently it will only report).

Metadata rules are associated with a community, and may be set to apply to child communities or not.

Webhooks

The third-party application is notified about events in Zenodo using webhooks - i.e. an HTTP request sent from Zenodo to a third-party on specific events.

A community may register one or more applications. Each application has an associated webhook which defines the URL to be notified (and other properties. A webhook registers which events it would like to receive.

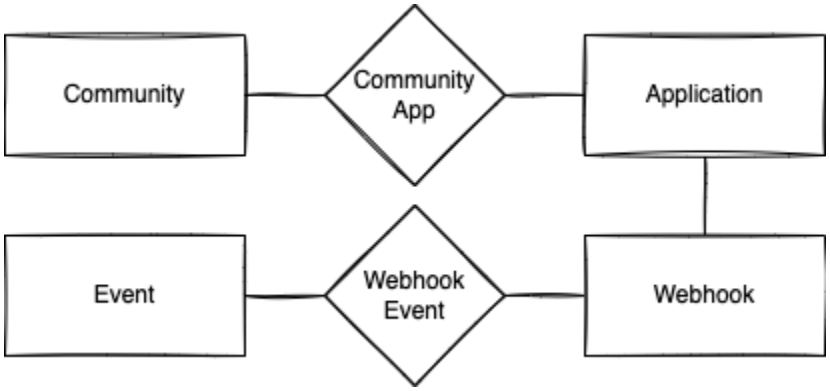

Figure 10: Entity relationship diagram of webhooks.

- Application: Main entity for registering the metadata associated with an application to be presented to end-users
- Webhook: A specific endpoint that should be notified - separate from application to ensure we can have hooks that are not associated with communities.
- Event: A specific type of event (e.g. new submission).

Deliveries receipts of webhooks (i.e. logging what was sent, when and what was the response) is currently out of scope for the implementation.

Events

The following event types will need to be defined and sent in order for the third-party application to run the file format check and report back on the status.

- `draft-review.submitted` - A new record draft review was submitted to the community.
- `draft-review.record-updated` - The associated record in an existing draft review was updated.
- `inclusion-request.submitted` - A published record was submitted for inclusion in the community.

Payload

The payload sent to the third-party application will provide the following information:

- Id
- Created at
- Type
- Actor
- Request
- RequestStatus

The request status will contain enough information for the third-party application to provide status updates on the request without having to do additional authentication.

Appendix

The following section provides details on the challenges in integrating third-party FAIR evaluation tools into the deposit workflow. **The section in no way meant as a criticism of the selected tool or other FAIR evaluation tools in general.** It's meant to provide concrete examples of the challenges which can serve as a basis for concrete discussion. The tools are not designed for the use case we are trying to use them for, and the tools themselves provide a lot of value for testing the repository software itself.

Test

We choose F-UJI to test a regular Zenodo record and test a draft record. The draft record was shared through a preview link (e.g. `https://zenodo.org/records/1234?preview=1&token=`) which generates the landing page for the draft record. The link includes an access token to grant access to the page.

Results

Differences between draft record and published record

The following tests fails or has a lower score because the DOI is not yet registered:

- FsF-F1-02D (fail): Data is assigned a persistent identifier.
- FsF-F4-01M (lower score): Metadata is registered in major research data registries (DataCite).
- FsF-I1-01M (fail): Parsable, graph data (RDF, JSON-LD) is accessible through content negotiation, typed links or sparql endpoint.

Because the DOI is not yet registered, it cannot be verified to be resolvable, nor can it be found in the DataCite Metadata catalog or use the DataCite content negotiation and thus fail 2 tests and get a lower score on one test.

Differences between records

The following tests will pass for all records in Zenodo given the following 3 metadata fields have been provided: keywords, related identifiers and license.

- FsF-F1-01D: Data is assigned a globally unique identifier.
- FsF-F1-02D: Data is assigned a persistent identifier.
- FsF-F2-01M: Metadata includes descriptive core elements (creator, title, data identifier, publisher, publication date, summary and keywords) to support data findability.
    - *Provided keywords have been provided.*
- FsF-F3-01M: Metadata includes the identifier of the data it describes.
- FsF-F4-01M: Metadata is offered in such a way that it can be retrieved programmatically.

- FsF-I1-01M: Metadata is represented using a formal knowledge representation language.
- FsF-I3-01M: Metadata includes links between the data and its related entities.
    - *Provided related identifiers has been provided.*
- FsF-R1-01MD: Metadata specifies the content of the data.
    - *5 out 6 subtests will give the same result*
- FsF-R1.1-01M: Metadata includes license information under which data can be reused.
    - *Provided the license has been provided.*
- FsF-A1-01M: Metadata contains access level and access conditions of the data.
- FsF-R1.2-01M: Metadata includes provenance information about data creation or generation.
- FsF-R1.3-01M: Metadata follows a standard recommended by the target research community of the data.
- FsF-A1-03D: Data is accessible through a standardized communication protocol.
- FsF-A1-02M: Metadata is accessible through a standardized communication protocol.

The following two tests are the only ones which really depends on metadata/data provided by the researcher on upload:

- FsF-I2-01M: Metadata uses semantic resources
- FsF-R1.3-02D: Data is available in a file format recommended by the target research community.

We also ran tests of a normal Zenodo record on FAIR Evaluator, FAIR Enough and FAIR Checker with a similar result.