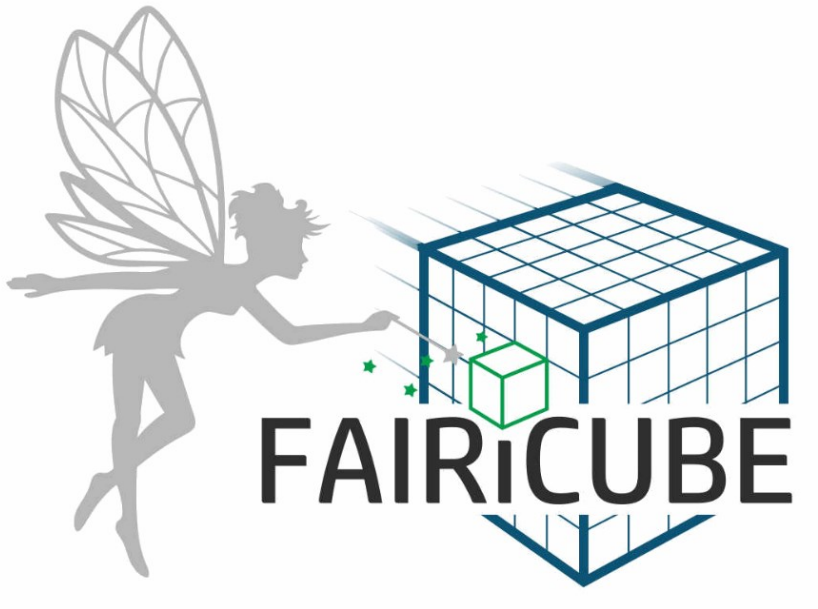


FAIRICUBE – F.A.I.R. INFORMATION CUBES

WP5 Ingest

D5.2 Description of the datacube ingestion pipelines

Deliverable Lead: Constructor (formerly: Jacobs) University

Deliverable due date: 31/12/2024

Version: 3.3

20/12/2024

Document Control Page

Document Control Page	
Title	D5.2 Description of the datacube ingestion pipelines
Creator	JUB (Jacobs University, aka Constructor University)
Description	This Deliverable describes how new data sets can be incorporated into the FAIRiCUBE Hub.
Publisher	"FAIRiCUBE – F.A.I.R. information cubes" Consortium
Contributors	EOX, EPSIT, S4E
Date of delivery	31/12/2024
Type	Text
Language	EN-GB
Rights	Copyright: authors
Audience	☑ Public ☐ Confidential ☐ Classified
Status	☐ In Progress ☐ For Review ☒ For Approval ☐ Approved

Revision History			
Version	**Date**	**Modified by**	**Comments**
0.0	01-02-2023	Peter Baumann, JUB	created
0.1	20-02-2023	Peter Baumann, JUB	updated
0.2	25-02-2023	Peter Baumann, JUB	updated
0.3	26-02-2023	Stephan Meißl, EOX	Re-adding lost content
0.4	27-02-2023	Peter Baumann, JUB	updated
	01-03-2023	Jaume Targa	Review
1.0	11-03-2023	Kathi Schleidt	Finalized, final version for sub-mission
2.1	08-11-2023	Mohit Basak, Peter Baumann	Updated
2.2	15-11-2023	Mohit Basak, Peter Baumann	Updated, and resolved all comments
2.3	17-11-2023	Schiller Christian	updated Data Ingestion Request Chapter 3 and Chapter 5.1
2.4	21-11-2023	Stefan Jetschny	Review and check of formatting
2.5	20-12-2023	Peter Baumann	Final formatting, comment resolution
		Jaume Targa	Review
2.6	12-01-2024	Peter Baumann	Incorporated review comments
2.7	16-01-2024	Jaume Targa	Final review
3.1	26-11-2024	Stefan Jetschny	Preparation of final delivery, update of document control pages
3.2	29-11-2024	Dimitar Misev	Updates for final delivery
3.2	29-11-2024	Schiller Christian	Updates/replacing Figures 3 & 4
3.3	19-12-2024	Jaume Targa, Stefan Jetschny	Final review

Disclaimer

This document is issued within the frame and for the purpose of the FAIRICUBE project. This project has received funding from the Horizon Europe research and innovation programme under grant agreement No. 101059238. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

Table of Contents

List of Figures

List of Tables

1 Introduction

The FAIRiCUBE integrated datacube platform consists of the two independent pillars EOxHub and Rasdaman, provided under the lead of partners EOX and JUB, respectively. Figure 1 shows the general architecture (see Deliverable D4.1 for details on it) of the FAIRiCUBE platform.

Before any data resources can be utilised, these must run through the ingestion process. This will differ depending on the nature of the original data. Some data sources will be provided read-only and as-is (such as the multi-Petabyte rasdaman Sentinel datacubes) whereas FAIRiCUBE-specific data will be ingested into rasdaman datacubes or EOX store, respectively. In addition, all datasets available at the Euro Data Cube (EDC) platform (https://collections.eurodatacube.com/) are available also to FAIRiCUBE users. This also includes the *EDC Sentinel Hub Batch Processing API* for asynchronous mass processing. Consequently, this Deliverable describes both the rasdaman ingestion pipeline as provided by JUB (Section 4) and the EOX equivalent (Section 5).

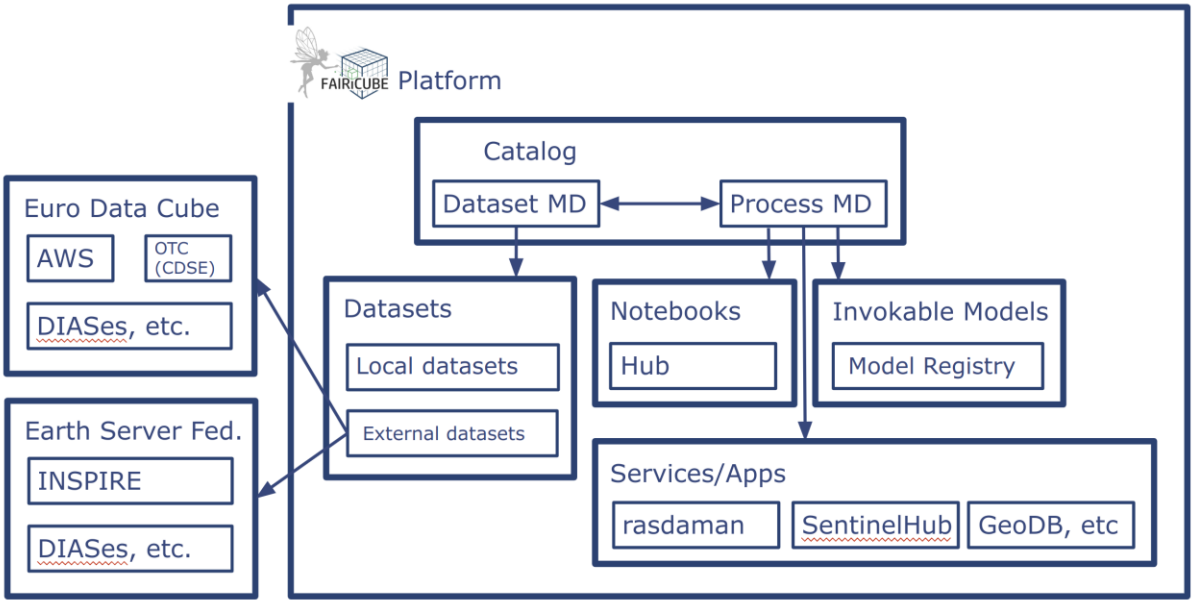


Figure 1: FAIRiCUBE Platform Architecture - Datasets & Models

2 Ingestion Pipeline Requirements

To perform the ingestion of data, the following elements are required:

- **Raster and non-raster data.** Focus has been on the "Big Data" parts, that is: gridded data. Non-gridded sources like vector data are considered according to Use Case partners' requirements and joint prioritization.
- **Metadata** play an important role. In the project, WP5 ingests "the pixels" whereas WP4 administrates the metadata. Therefore, ingestion normally involves two steps, data and metadata ingestion, whereby these parallel resources maintain tight links between them.
- **Access control** is an important facet – not all data are open, and some data owners impose particular regulations on the use of their data. Therefore, individual, fine-grain access control has been established. Both project partners, as well as external entities such as the sister project AD4GD or students in hackatons organized by FAIRiCUBE, get access via their GitHub accounts or via username/password credentials.
- **Defined ingestion process.** The process of suggesting, selecting, ingesting, and communicating the availability of data needs to be well-defined and easy to follow, both for regular users as well as for non-experts. An additional challenge is the fact that two quite different datacube services have been included in the project, rasdaman and EOxHub, both offering EO (Earth Observation) data. One ingestion request procedure has been defined for both rasdaman and EOX, as the nature of the data to be ingested is the same in both cases. On a technical level, as rasdaman and EOX rely on two very different and independently developed software stacks, details vary.

3 Data Request & Ingestion Procedure

In this section, the high-level ingestion process is documented emphasizing the user perspective. While it has been clear from the onset that the goal will be an ingestion process as automatic as possible within FAIRiCUBE, it also became apparent that the Use Cases will need to commence work on specifying required data before this ingestion process has been finalized. In the following the agreed procedures are described for data (WP5). Note that metadata are handled in WP4, therefore see Deliverable D4.3 for resource metadata ingestion and codelist change requests.

In the sequel, the data ingestion process is described: If some additional dataset is requested by a Use Case via a Catalog Editor at https://catalog-editor.eoxhub.fairicube.eu/, which presents a form where the requester can provide the necessary information. The provided information is validated for completeness, and internally a machine-readable request is generated initiating the process. Technically, the Catalog Editor is a frontend to a GitHub repo https://github.com/FAIRiCUBE/data-requests; from the form submission, a GitHub Pull Request is issued as a new branch which the user is automatically watching and thus receiving notifications of updates depending on their GitHub notifications configuration. Any progress, problems, discussions, etc. are documented in GitHub comments associated with the respective Pull Request, so that everybody interested can follow the progress and provide additional feedback or information as necessary.

This is the final approach adopted. Historically, an initial proposal of a Datacube Management Plan was abandoned because it relied on text documents which left too much of a degree of freedom for filling in. Subsequently, an agreement was reached to first collect information on the requested data sets in an "inventory" Excel table shared with all partners. This allowed for discussions on data requirements within the project team while also detecting overlapping data needs across Use Cases (reported in deliverable D2.1). This was understood as an interim solution until a more convenient, but also more controlled data request generation would become available. The schematic procedure for a data request is shown in Figure 2.

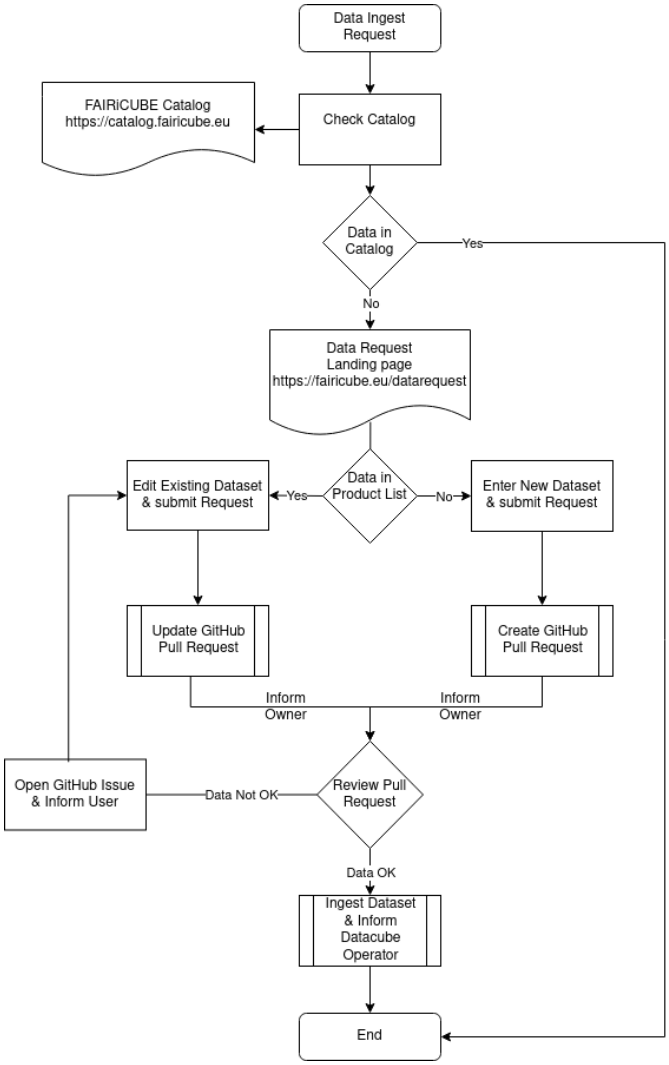


Figure 2: Data Ingestion Request Procedure

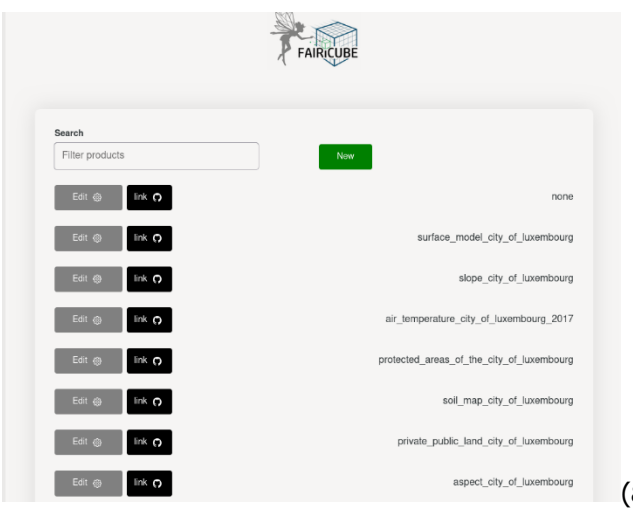
Once all metadata and data requirements are fulfilled and confirmed by the data requester, the ingestion handling partners will perform the merge, and the issue will be closed. Any discussion remains available. In order to enable users to more easily handle the challenge of providing the relevant information to the FAIRiCUBE catalog a "Catalog Editor" Interface has been developed. This Catalog Editor supplies a web-based GUI which provides a multitude of fields to be filled out. by the user. These fields render helpful information concurrently with plausibility checks while the information is entered by the user.

Figure 3(a) shows the landing page of the Catalog Editor enabling the users to supply metadata information in an orderly manner. The landing page lists the available datasets to be edited, each with an Edit Button associated to proceed further. Figures 3(a)-3(l) show an example of a datasets with all the fields provided by the Catalog Editor.

When a user enters a name in the *Search field* and presses the *Add* button on the Landing page, then the data entry form will be displayed (Figures 3(a)-3(l)). If a user chooses an already existing dataset and uses the *Edit button* the same entry form will be shown with the values available already filled in(Figures 3(a)-3(l)). The Catalog Editor is accessible for FAIRiCUBE users at https://catalog-editor.eoxhub.faricube.eu.

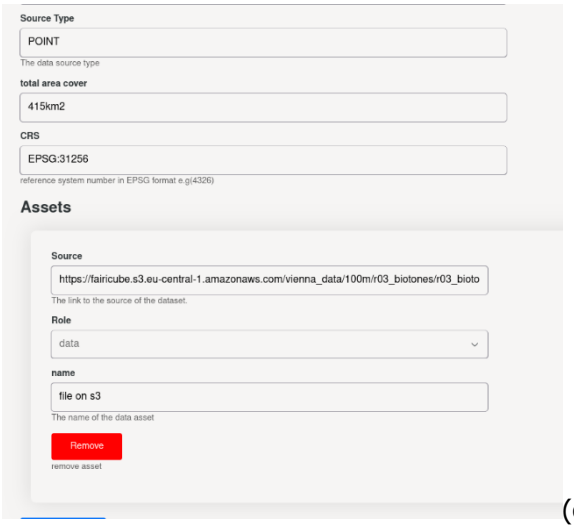
The submitted metadata entries are stored for further processing or editing until the datasets is deemed ready and gets submitted to a GitHub issue. There a last manual check by the "ingestion handling partners" will be performed prior to the final merge and closing of the respective issue. All comments which might have been associated with the merge respective GitHub issue will remain available.

When the merge is done the newly submitted data is available in the STAC Browser deployed at https://catalog.fairicube.eu. The STAC Browser provides additional features like searching, which are not available in the static STAC catalog. Sample screenshots of the STAC Browser are provided in Section 5.1.

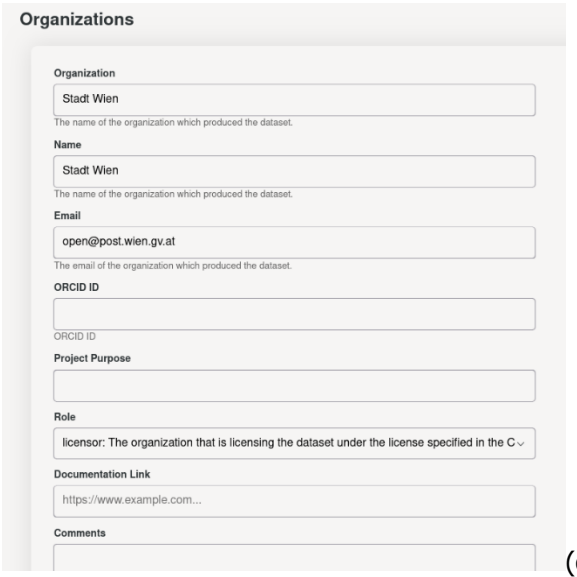


(a)

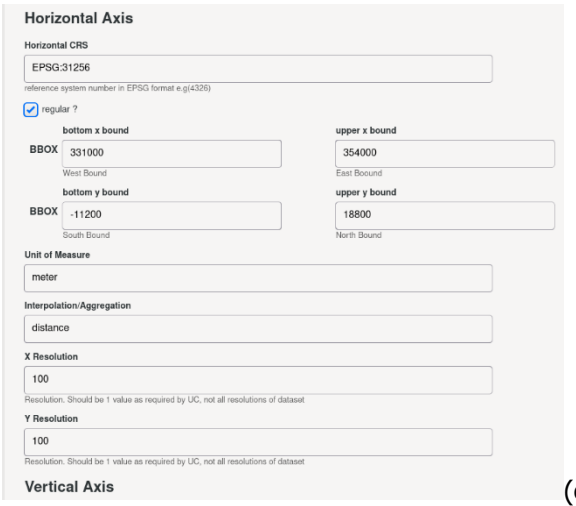
(b)



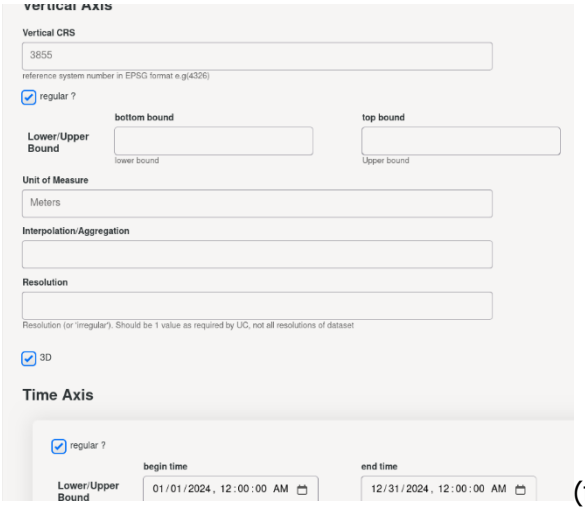
(c)



(d)



(e)



(f)

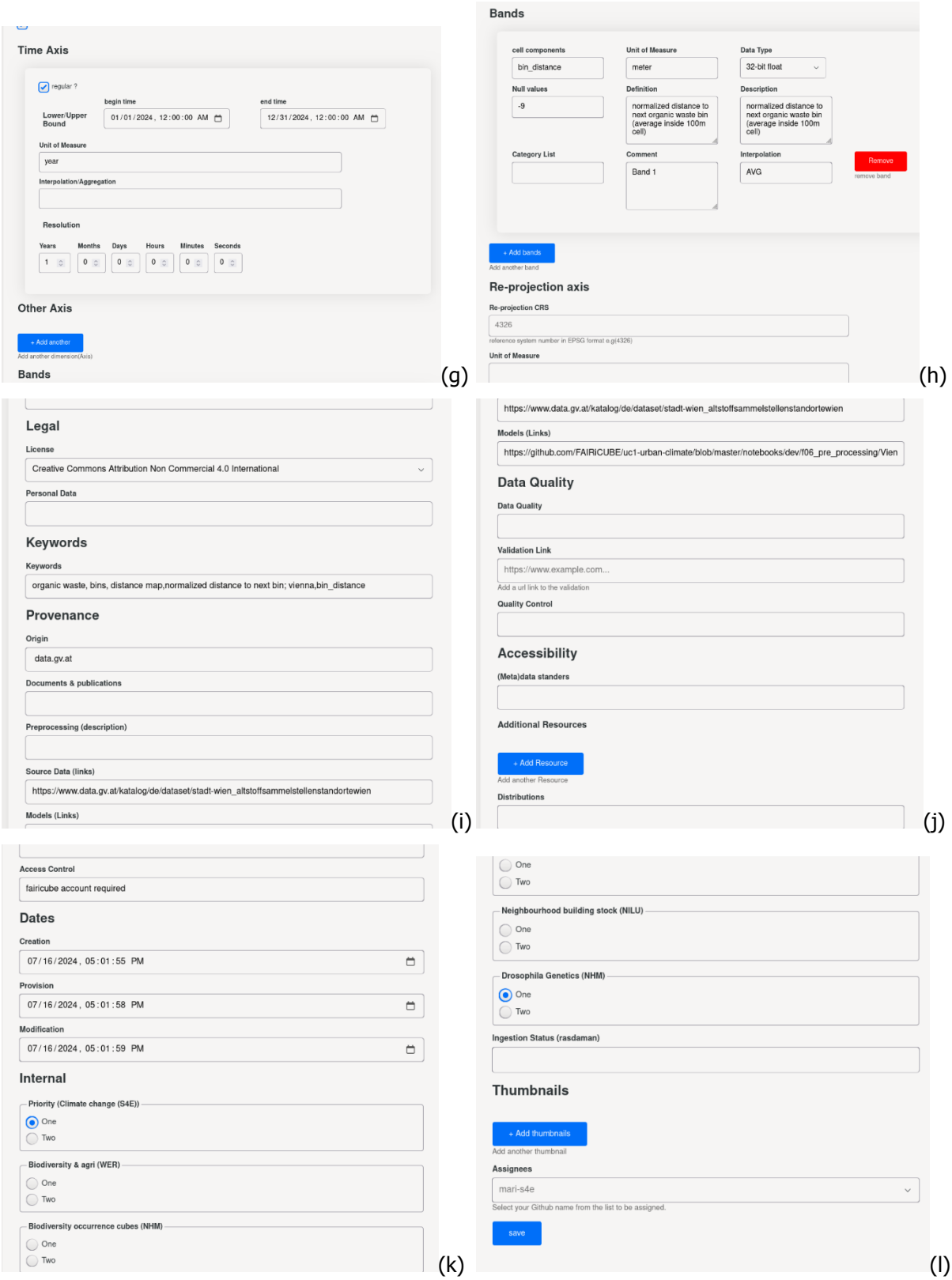


Figure 3: (a) showing the landing page of the Catalog Editor. (b-l) showing the Catalog Editor in editing mode with all the fields of metadata information available for of a dataset

4 rasdaman Ingestion Pipeline

4.1 Service Overview

Technically, the rasdaman deployment is split into two VMs (Virtual Machines), both allocated in the EU:

- one VM offers several Petabytes of data from ds.earthserver.xyz; this VM is for read access via federation from the FAIRiCUBE VM (see below) only.
- one VM contains the FAIRiCUBE playground where partners have logins to perform any action desired using WMS, WMTS, WCS, and WCPS (several partners additionally have operating systems login to this machine for experimental purposes).

These two VMs have been set up so that the data from ds.earthserver.xyz (where build-up and maintenance of datacubes typically are expensive) cannot be compromised by FAIRiCUBE experiments. This VM is federated with the FAIRiCUBE VM to provide read-only access to institutional users of data.

4.2 General Documentation and Support

The complete rasdaman documentation, containing, in particular, the ingest documentation, is available online, plus a series of tutorials with interactive sandboxes, YouTube videos on this channel, etc. All these are documented in the FAIRiCUBE rasdaman notebook in the project workspace, accessible to the partners, as well as provided below in Table 1. Furthermore, on-demand tutorials and continuous support are provided by JUB. An integration of respective links from the FAIRiCUBE Knowledge Base will ensure user-friendly access to these resources.

Resource	Link
rasdaman documentation	https://doc.rasdaman.org/
Ingest documentation	https://doc.rasdaman.org/05_geo-services-guide.html#data-import
Tutorials	https://earthserver.eu/wcs
Sandbox	https://standards.rasdaman.com/
YouTube videos	https://www.youtube.com/@PeterBaumannRasdaman/featured

Table 1: Additional rasdaman resources

4.3 Data Ingest Logistics

The rasdaman team, as part of its user support, performs ingestion for the Use Case partners based on their requests.

Any potentially open issues get resolved in direct discussion between data requesters and the support team. Likewise, after completion of the ingestion, a joint inspection takes place for data validation. Finally, the GitHub data request is merged closed. In brief, the workflow is:

- Use case partners create a data-request issue indicating key parameters, such as source/origin of data, description, etc. through the WebGUI as specified in chapter 3.2;
- Rasdaman's team validates and asks back (via issue communication) where necessary until enough information is provided for creating the datacube;
- The ingest process is created and executed (technical details are provided in chapter 4.5);
- After ingestion, the rasdaman team performs an internal quality control;
- Finally, the datacube is then validated by the use case partners, typically in a virtual meeting.

In addition, those FAIRiCUBE use case partners who are interested in performing ingestion themselves can add their own datacubes directly using the automated ETL ingestion suite of rasdaman, which is based on the OGC WCS-T standard.

4.4 Datacubes Currently Available

Following the described procedure above, a number of datacubes have established based on the respective Use Case requests, communicated via the data inventory sheet in combination with GitHub requests. Most of the datacubes resemble timeseries, with a temporal resolution between a few days and several years. The authoritative list of datacubes is provided with Deliverable D5.1 (List of Datacube Resources Made Available). For the reader's convenience, here is a snapshot of the datacubes available at the time of submission of this deliverable:

- Dominant Leaf Type
- Grassland Status
- Forest Type
- Imperviousness
- Tree Cover Density
- Water and wetness
- Small Woody Features
- LGN
- Near Surface Air Temperature
- Corine land cover
- European Settlement Map
- EU demography
- Agrodatacube-derived datacubes
- Temperature and Precipitation in the Netherlands
- Global Pesticide Grids
- Sentinel-2 cloud-free over multiple months/years
- GBIF species occurrence
- ERA5-Land monthly averages 1950 – present

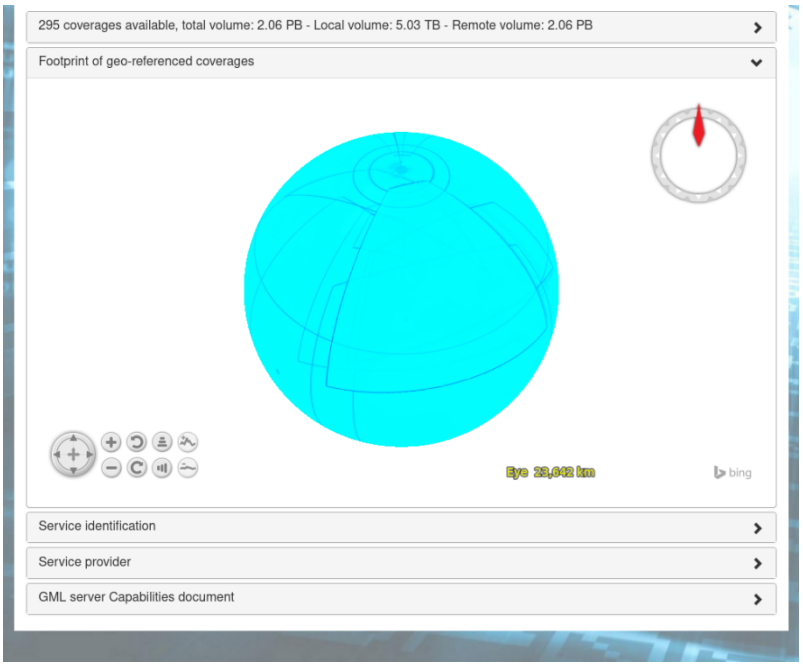


Figure 4: Footprints of rasdaman datacubes on the FAIRiCUBE server.

In addition, about 220 Copernicus datacubes (Sentinels, C3S, EU_DEM, etc.) are available via the EarthServer federation, including (but not limited to):

- Sentinel-1 GRDH (all polarizations)
- Sentinel-2 L2A (data bands as well derived products like TCI, SCL and cloud/snow masks)
- Sentinel-3 OLCI (all products L2)
- Sentinel-5p (all products, L2).

4.5 Datacube Ingestion: Technical Details

Creation and maintenance of datacubes in rasdaman are mostly automated. A single short configuration file per datacube regulates the complete process, from input file origin over intermediate processing steps up to placing incoming pixels into the right space and time position in the cube. Also, image pyramids, necessary for fast map zoom support, are created and maintained automatically.

Internally, data ingest relies on the OGC WCS-T standard. However, as this is very low-level and operates on single files, an intelligent ETL (Extract, Transform, Load) suite has been built on top of WCS-T in rasdaman which automates most of the tasks. The definition of each datacube is governed by a recipe (such as for Sentinel-1 radar, Sentinel-2 optical, regular vs irregular grids, etc.), which is configured by the provision of an ingredient file crafted specifically for the import task. Ingredients include required information like:

- datacube name, CRS, null values
- the input data file paths;
- metadata not available in the input files (whereby common situations, such as TIFF / TFW file pairs, are detected automatically);
- whether WMS support is requested (in which case rasdaman builds and maintains pyramids automatically) and registers WMS layers;
- WMS styles, whether in the form of a color ramp/palette table, or WCPS query fragments that are applied before rendering the map;
- specification on how to determine the datetime coordinates of input files which are most often in a 2D spatial data format such as TIFF, e.g. by extracting it from the file name, path, or metadata; additionally areas of validity may be specified which set the footprint of each point on an *irregular* time axis;
- information about the data bands: the type (quantity or categorial data), name, label and description, unit of measure (UoM) for quantity data or codespace for categorial data, null values;
- whether data import should be done by copying into the database ("ingest") or just registering the input files for further query processing directly on these files ("in-situ");
- physical optimization directives, such as the tiling, compression, or indexing method to use for the ingested data.

Also, "virtual datacubes" can be built from existing, even heterogeneous datacubes, such as the integration of Sentinel-2 data in different UTM zone grid tiles into a single Sentinel-2 datacube. This ETL suite detects common situations automatically and is additionally highly configurable for all sorts of data import situations, including customized pre-processing of input data as necessary. Simple pre-processing steps can be captured in the resource metadata associated to the ingested data set; more comprehensive pre-processing is prepared with Shell or Python scripts. This way, the import action is automated and fully reproducible as long as the source data is available.

The initial data requirements in FAIRiCUBE are covered by the existing predefined recipe selection but the definition of custom recipes is also quite straightforward. A sample ingredient file for a Sentinel-2 timeseries is shown in the paragraph below. Notice the "automated" parameter set to "true" for continuous ingestion of incoming data (see next sections for details).

```
{
        "config": {
                "service_url": "http://localhost:8080/rasdaman/ows",
                "automated": true
        },
        "input": {
                "coverage_id": "S2_${crsCode} ${resolution} ${level}",
                "paths": [ "S2*.zip" ],
                "resolutions": ["10m", "20m", "60m", "TCI"],
                "levels": ["L1C", "L2A"],
                "crss": ["32757"]
        },
        "recipe": {
                "name": "sentinel2",
                "options": {
                        "coverage": {
                                "metadata": {
                                        "type": "xml",
                                        "global": { "Title":"Sentinel-2 data served by rasdaman""
                                }
                        }
                },
                "tiling": "ALIGNED [0:0, 0:1999, 0:1999] TILE SIZE 32000000",
                "wms_import": true
        }
}
```

The FAIRiCUBE Catalog contains a link to the datacubes provided. Conversely, each datacube contains a metadata slot with a link to the corresponding STAC entry. This way, a mutual linkage is provided. Establishing this mutual linking is done automatically via an API call provided by the catalog developers.

4.6 Data Problems Encountered

Problems encountered with the data include:

- Sometimes a link provided leads only to metadata, but these do not contain data links; this requires extra search to spot where and how data can be downloaded.

  Solution: Remains manual work.

- Some data require a signed compliance statement, to be provided by the Use Case partner requesting the data.

  Solution: Remains manual work.

- Issues with varying data quality in dataset series; for example, some Copernicus data sets change spatial resolution over time.

  Solution: For each resolution occurring all corresponding data have been put into a single datacube each so that each such datacube contains a homogeneous resolution. For example, Dominant Leaf Type data has been split into a dominant_leaf_type_20m datacube 2012 and 2015, and a dominant_leaf_type_10m for the year 2018 (which is the last year currently available).

- In some datasets the data semantics change over time. For example, the Small Woody Features dataset for the year 2018 has different categorial values than for the year 2015. Hence both time slices cannot be ingested into a single datacube.

  Solution: These datasets have been split into multiple datacubes, usually suffixed by the years they correspond to.

- Categories sometimes have been extended with additional classes over time, in the same dataset, e.g. the LGN dataset of land use in the Netherlands.

  Solution: Requires manual intervention and disclosing inhomogeneity to users.

- Generally, in some cases, legends have been observed to vary over the years.

  Solution: Alerted by this as well as the small woody features issue mentioned above it was decided to investigate individually for each dataset whether this is just a change in visualization or – deeper – in data semantics. Remains manual work.

- Time specifications in the repositories that had to be harvested were sometimes inexact and incomplete. While discussing how to get to an accurate temporal description it turned out that temporal axis functionality was required going beyond what today typically is offered by rasdaman and other tools.

  Solution: A detailed, advanced temporal selection capability has been developed, offering substantially more powerful and flexible calendar functionality.

- Copernicus data undergo evolution, documented in version numbers (e.g., at the time of this writing a new Sentinel-2 baseline has been issued with productBaseline = 5.09). There is some documentation, but checking whether new data are comparable with older timeslices or instead data properties have changed remains a manual task. Information like this might be insufficient as it does not mention changes in processing parameters etc., but only the renaming of products. Even more substantially, ESA sometimes performs a reprocessing of Sentinel-2 data retrospectively based on new findings, such as an improved DEM used for correction. This retroactively changes the radiometry of Sentinel-2 data offered. While accuracy improvement on principle is something to applaud, we expect negative, yet difficult-to-detect implications, such as on models trained on older versions of the pixels.

  As an illustration below a Sentinel-1 scene is shown, left with an acquisition (and, hence, ESA processing) time between June 2017 and May 2018, right with a Sentinel-1 patch from 2023 extracted from the ESA Copernicus archive. Obviously, not only changes in tools but even variations in the processing parameters can have a significant impact[1]. Experience shows that neural networks, for example, are highly sensitive to such changes.

  Solution: Manual inspection upon every product update. Upon retroactive reprocessing pyramids may need to be recomputed, which constitutes a significant effort.

[1] N. Djamai, R. Fernandes: Comparison of SNAP-Derived Sentinel-2A L2A Product to ESA Product over Europe. Remote Sensing 2018, 10, 926, doi: 10.3390/rs10060926

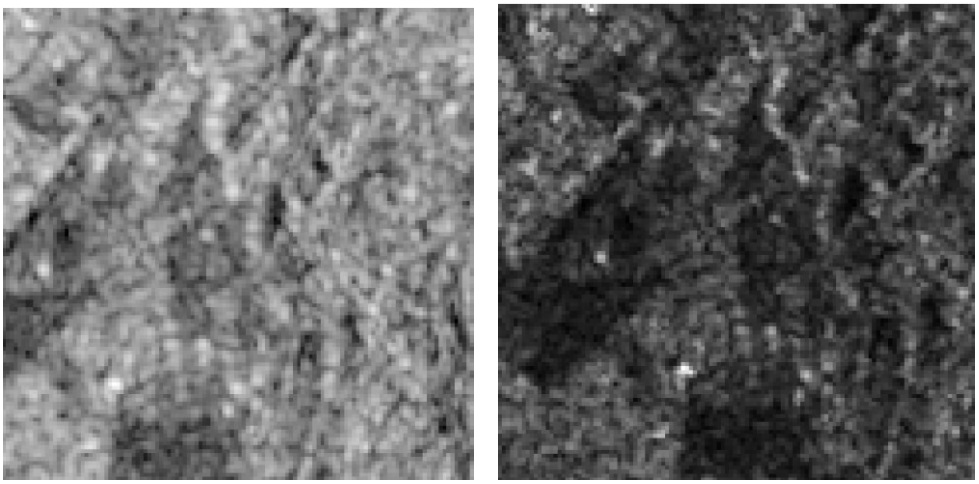


Figure 5: Sentinel-1 scenes delivered by ESA with different processing parameters applied.

5 EOX Ingestion Pipeline

5.1 Data Ingest Logistics

Technically, the pipeline provided on the EOX deployment is more a matter of registration than of ingestion, as data uploaded to object storage only needs to be registered in services as required and not ingested, i.e., copied. The EOX deployment provides a bucket on object storage for each use case team to store relevant datasets. The credentials to access objects stored on object storage are transparently injected in the user's EOxHub workspace as well as any integrated external services and apps such as Sentinel Hub. Figure 6 shows the workflow on how to upload data to the FAIRiCUBE Hub using the EOX platform.

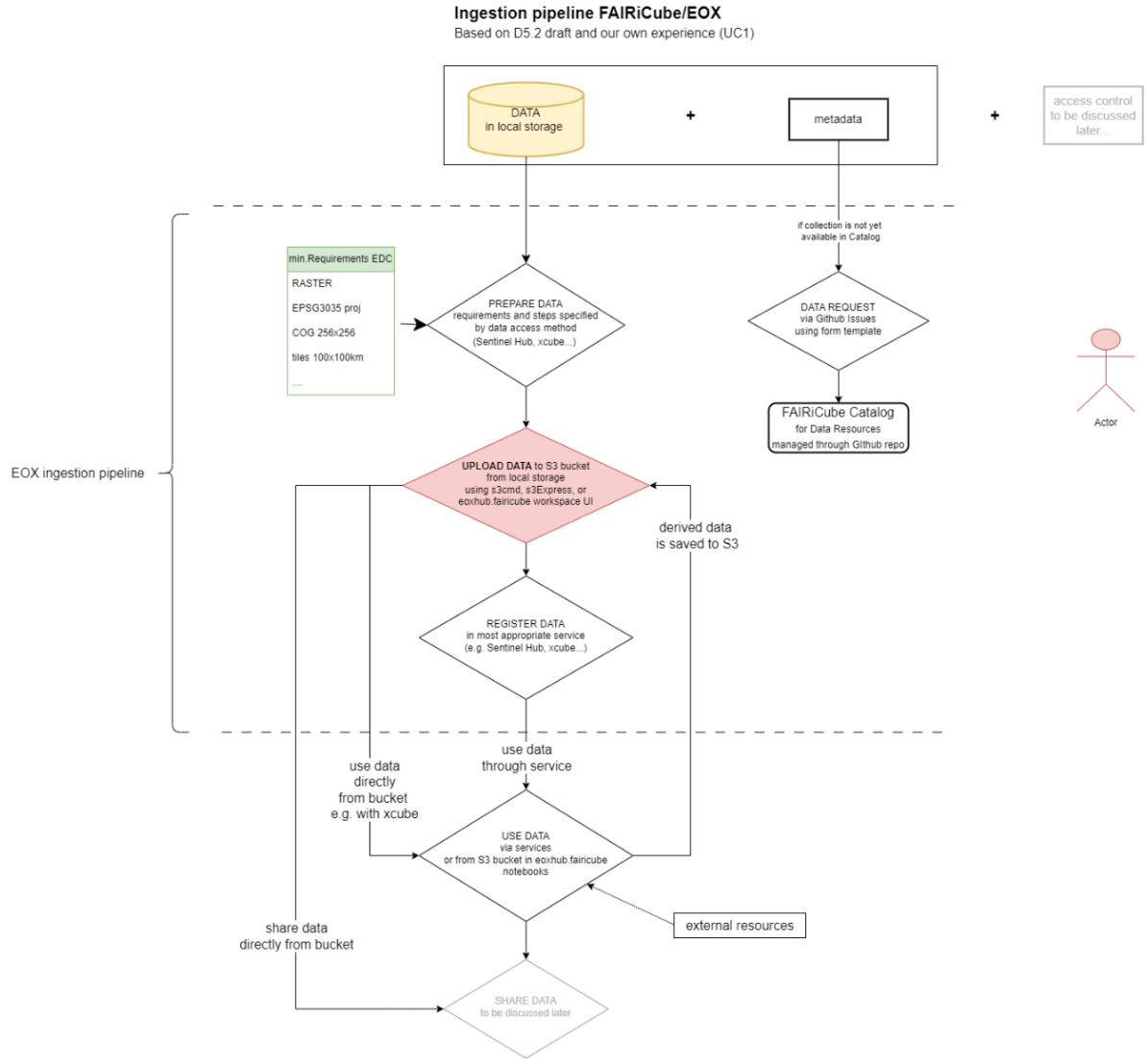


Figure 6: Data ingestion pipeline workflow using EOX-platform.

As a first step, the use case representative requesting new data together with the ingestion handling partner EOX reviews and decide which services are required to be available for the new data. The most interesting services are either direct access to object storage via S3 protocol or the API suite provided. by Sentinel Hub like Process API, Batch API, Statistics API, openEO, WMS, etc. If Sentinel Hub services

are required, the data has to be converted to Cloud Optimized GeoTIFFs[2] or ZARRs[3] following the constraints and settings explained in the given links. It turned out that the direct usage of data provided as CDF and NetCDF files, which is a widely used standard, especially in the Meteorological world, is not very efficient when stored on S3 (or object storage in general). It is therefore recommended to convert datasets to ZARRs or GeoTIFFs before using it (see description for the conversion provided for the Sentinel-Hub above). The two formats are also recommended in case direct s3 access is required. There are two options of how to pre-process or convert own data that a user wants to use in the FAIRiCUBE Hub:

- Upload the selected data without further pre-processing into the s3 bucket. This can be any kind of spatial data set in any projection. If such a raw dataset has been uploaded, further processing must be done directly on s3. For example, a Jupyter Notebook can be created that converts the raw dataset into a projected Cloud Optimised GeoTIFF (COG) with a pixel size of 10m.
- Perform the pre-processing before uploading the dataset to the s3 bucket. If feasible, we would recommend this option to save CPU and memory costs.

Regardless of which of the two options is chosen, in the end, the final data (dimensions) on s3 must fulfil various conditions if to be registered in Sentinel Hub[4].

Own data requirements:
Spatial raster dataset must be stored in ZARR format: □ COG block size: between 256 x 256 and 2048 x 2048.
Projection: □ The projection needs to be one of: WGS84 (EPSG:4326), WebMercator (EPGS:3857), any UTM zone (EPSG:32601-32660, 32701-32760), or Europe LAEA (EPSG:3035).
Max. number of bands: □ 100

Table 2 : Upload data requirements

The next step is to upload the data to the provided bucket on object storage, e.g., 'hub-fairicube0'. The provided buckets are configured with a soft quota, i.e., with an alert at a certain size to make each team aware of costs. The bucket can be mounted via fuse in each team member's workspace, for example under a folder 's3'. All files which are saved in this folder are read- and writeable for all members of the team. Additionally, the files can be made available via s3 protocol directly, for example, to be registered in Sentinel Hub. Moreover, the credentials to access the bucket are injected in the JupyterLab session, meaning they are available as env variables. This means for example that direct s3 access via tools like 's3cmd' works in the workspace out of the box. Example commands describing object storage usage:

```
pip install s3cmd

./local/bin/s3cmd put --access_key=$username --secret_key=$password –region=eu-central-1 --host=s3.amazonaws.com test.txt s3://$endpoint

./local/bin/s3cmd ls --access_key=$username --secret_key=$password –region=eu-central-1 --host=s3.amazonaws.com s3://$endpoint
```

[2] https://docs.sentinel-hub.com/api/latest/api/byoc/#converting-to-cog

[3] https://docs.sentinel-hub.com/api/latest/api/zarr

[4] https://docs.sentinel-hub.com/api/latest/api/byoc/#a-note-about-cog-overviews- used-for-processing

After uploading the data, there are various possibilities to address the data in the Jupyter notebook on the EOX platform:

- Direct reading the data from the s3 bucket
- Create a collection and read the data using the sentinel hub and API.

Now the data can be registered in services like Sentinel Hub as necessary. Registration in Sentinel Hub can either be performed via API directly or using a Python library or a web dashboard. Now the data requester can start using the new data either via the services the data is registered in or directly from object storage. The final step, however, is to properly share the new data depending on the data license and the willingness to maintain and cover costs incurred. As a minimum, the use case team has to cover the storage costs of s3 object storage. There are two other types of s3 costs namely for bandwidth and requests which can be either covered by the data provider or the requester depending on configuration.

In case the use case team decides to provide direct s3 object storage access, they have to decide if they want to configure the 'requester pays' option. Using this option requires any data user to have a valid AWS account to cover the costs incurred by bandwidth and requests. If this option is not used all costs need to be covered by the use case team. It must be noted that opening a bucket this way can incur significant costs as the use case team has no control over actual usage by external parties.

There are two options for sharing the data via the Sentinel Hub APIs. Either the collection ID is shared publicly which allows other users to use it in their Sentinel Hub subscription and thus not incurring costs for the use case team. Alternatively, the instance ID and potentially layer IDs are shared which means that any usage needs to be covered by the Sentinel Hub subscription of the use case team.

5.2 Datacubes Currently Available

Any dataset or collection listed at https://catalog.fairicube.eu and showing 'Sentinel Hub Resources', 'xcube Resources', or `geoDB Resources' are currently available. For example, the ESA WorldCover dataset is available via Sentinel Hub using the provided collection ID as shown in Figure 7 below.

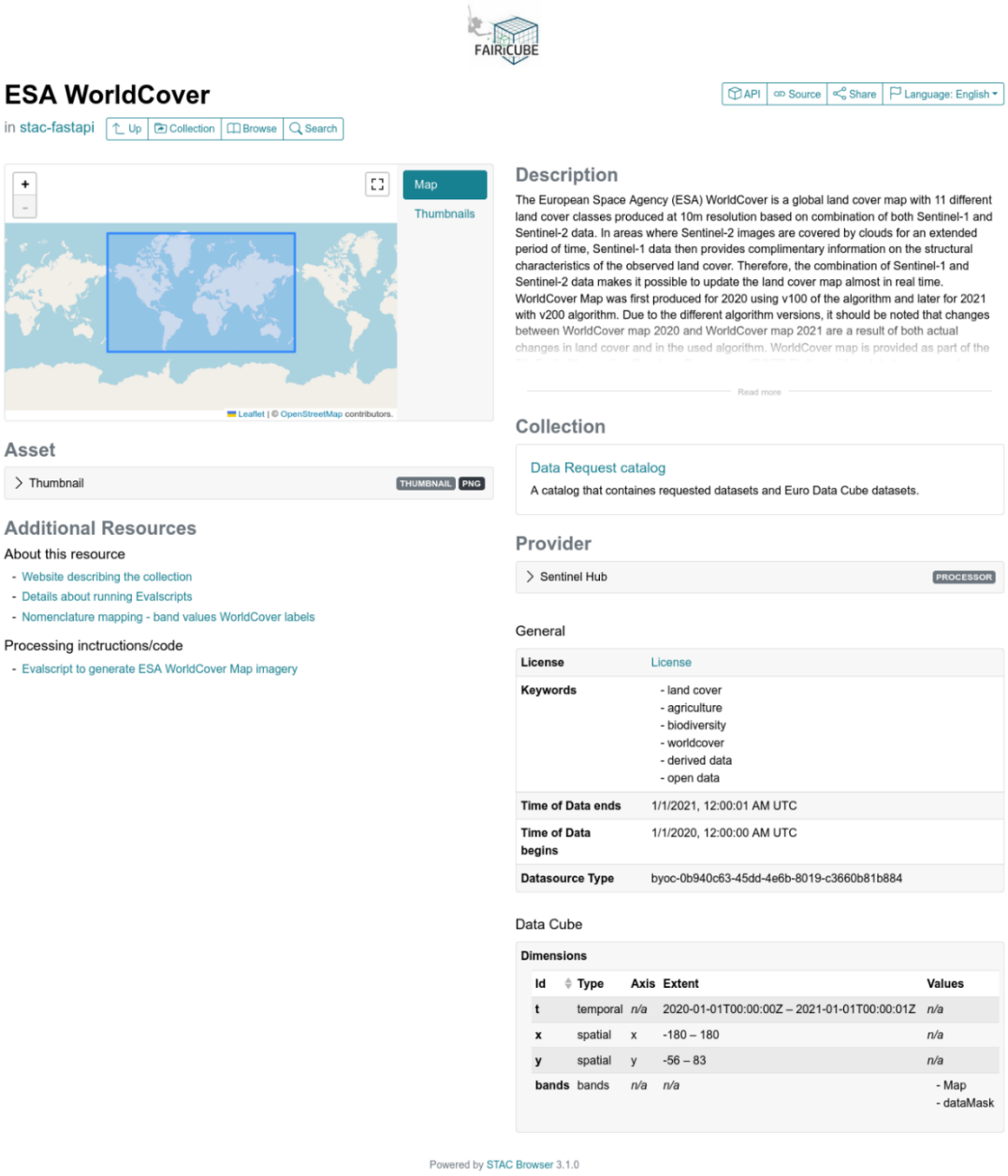


Figure 7: ESA World Cover

6 Summary

The ingest routine has been established and demonstrated through several user-selected data sets, some bringing interesting challenges. Among others, data sets retrieved from their official sources sometimes have been found corrupted (such as 0-length files). Issues are documented routinely, as per the standard ingest workflow; by commenting on the existing GitHub Issue the requesting use case partner can react appropriately (such as pointing to an alternative source where available).

Various project-specific datasets have been ingested through the workflow described. Additionally, in rasdaman, the federated DIAS archive data are available in a location-transparent manner, while EOX provides access to the Sentinel Hub data. All data ingested are readily available for access and processing and linked with the STAC catalogue's metadata records catalog. Therefore, the WP5 partners believe that a stable state has been achieved.