

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: 30/06/2024

Version: 2.9

05/08/2024

Document Control Page

Document Control Page	
Title	D5.2 Description of the datacube ingestion pipelines
Creator	CU (Constructor – formerly: Jacobs - 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	2024-06-30
Type	Text
Language	EN-GB
Rights	Copyright: authors
Audience	<input checked="" type="checkbox"/> Public <input type="checkbox"/> Confidential <input type="checkbox"/> Classified
Status	<input type="checkbox"/> In Progress <input type="checkbox"/> For Review <input checked="" type="checkbox"/> For Approval <input type="checkbox"/> 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
0.5	01-03-2023	Jaume Targa	Review
1.0	11-03-2023	Kathi Schleidt	Finalized, final version for submission
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
2.8	25-05-2024	Peter Baumann, Christian Schiller	Update following reviewer request
2.9	25-06-2024	Kathi Schleidt, Stefan Jetschny	Internal review, fixed some typos, added comments, format checking



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.

This document and its content are the property of the FAIRiCUBE Consortium. All rights relevant to this document are determined by the applicable laws. Access to this document does not grant any right or license on the document or its contents. This document or its contents are not to be used or treated in any manner inconsistent with the rights or interests of the FAIRiCUBE Consortium or the Partners' detriment and are not to be disclosed externally without prior written consent from the FAIRiCUBE Partners. Each FAIRiCUBE Partner may use this document in conformity with the FAIRiCUBE Consortium Grant Agreement provisions.



Table of Contents

Document Control Page	2
Disclaimer	3
Table of Contents	4
List of Figures	5
1 Introduction	6
2 Ingestion Pipeline Requirements	6
3 Data Request & Ingestion Procedure	7
3.1 General Ingestion Procedure	7
3.2 Geo Data Types	11
4 rasdaman Ingestion Pipeline	12
4.1 Service Overview	12
4.2 General Documentation and Support	12
4.3 Data Ingest Logistics	13
4.4 Datacubes Currently Available	13
4.5 Datacube Ingestion: Technical Details	14
4.6 Problems Encountered	15
4.6.1 Data Problems	15
4.6.2 Conceptual Problems	17
4.6.2.1 Temporal Semantics	17
4.6.2.2 Multiple Datacube Views	18
4.6.2.3 Datacube Catalog	18
5 EOX Ingestion Pipeline	20
5.1 Data Ingest Logistics	20
5.2 Datacubes Currently Available	22
6 Summary	23



List of Figures

Figure 1: FAIRiCUBE Platform Architecture - Datasets & Models	6
Figure 2: Data Inventory Sheet (vertically: dataset candidates, horizontally: metadata required) – note that the bottom-right cell has address BY64	8
Figure 3: Data Ingestion Request Procedure	9
Figure 4: Data Ingestion Request Web GUI – Landing page	10
Figure 5: Data Ingestion Request Web GUI – First part of the data entry form	11
Figure 6: Footprints of rasdaman datacubes on the FAIRiCUBE server., including federated datasets	14
Figure 7: Sentinel-1 scene delivered by ESA with different processing parameters applied	17
Figure 8: rasdaman catalog, integrated with dashboard	19
Table 1: rasdaman resources	13



1 Introduction

The FAIRiCUBE integrated datacube platform consists of the two independent pillars EOxHub and Rasdaman, provided under the lead of partners EOx and CU, 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 or the resources made available via Sentinel Hub) whereas FAIRiCUBE-specific data will be ingested into rasdaman datacubes or EOx store, respectively. In addition, all datasets available at CoperniCUBE (<https://copernicube.eu>) and 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.

This Deliverable describes both the rasdaman ingestion pipeline as provided by CU (Section 4) and the EOx equivalent (Section 5).

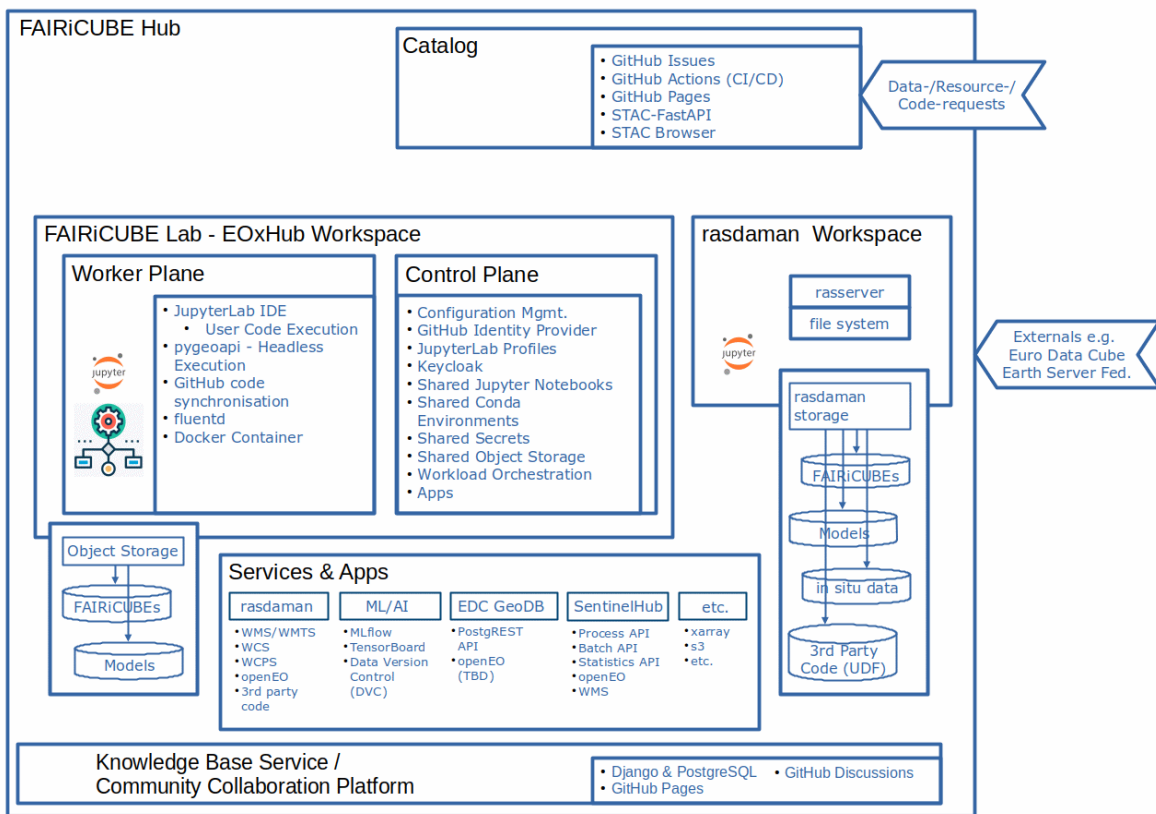


Figure 1: FAIRiCUBE Platform Architecture - Datasets & Models

2 Ingestion Pipeline Requirements

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

- **Raster and non-raster data.** Focus on this initial phase has been on the “Big Data” parts, that is: gridded data. Non-gridded sources like vector data will be considered in the next project phase, driven by 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 needs to be established. In this initial stage where access is restricted to the consortium, such access control is not yet in place, it will be addressed as a next step once a comprehensive project data access governance has been set up. For example, currently, partners get access to the rasdaman datacubes provided by FAIRiCUBE partner CU, but not external entities.
- **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 meta data ingestion and codelist change requests.

3.1 General Ingestion Procedure

The data ingestion process is as follows, identical for both rasdaman and EOx pillars. If some additional dataset is requested by a Use Case then a new record is generated on <https://github.com/FAIRiCUBE/data-requests> via a new GitHub issue. This has been made more convenient through <https://data-request.fairicube.eu> based on a Web GUI with a form where the requester can provide the necessary information. From this information, internally a machine-readable request is generated initiating the process. Technically, the Web Gui is a frontend to a GitHub repo; 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. shall be documented in a GitHub issue 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. Instead, the WP2 coordination decided for an “inventory” Excel table for collecting information on the requested data sets as an initial stop-gap until tool support would become available. 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 abovementioned GitHub Web GUI).

The Inventory Sheet required some minimum specification of the data to enable WP5 locating and ingesting them. In order to keep the Use Case effort low, WP5 kept this tentatively minimal. However, at some time catalog information and other potentially useful information was required, ultimately growing the Inventory Sheet to about 75 items to be entered by Use Cases before an entry should be accepted for ingestion. In fact, this never happened – at the Fall 2023 project meeting the WP5 lead

conducted a straw vote among Use Case partners which revealed that most were willing to provide about 20 entries, few up to 40 entries, and nobody beyond. Please note that other similar projects with comparable systems are said to foresee up to 400 individual metadata concepts, 150 of these mandatory; it remains open, though, how these will be added manually each time. Notably, the straw vote mentioned was purely based on the number of metadata entries, not the concepts; the scope of required metadata concepts has remained unchanged.

Therefore, at some time WP5 started ingestion even when dataset request entries were incomplete. Further, Inventory Sheet v2 was declared frozen in its metadata requirements to achieve a stable basis for continuation. Figure 2 gives a visual impression of the size of the sheet.

Ultimately, entries (rows) in the Inventory Sheet were replaced by GitHub issues with fixed forms for the metadata entries, eventually made more user friendly with Web forms. Hence, today the Inventory Sheet is deprecated and not used any longer.

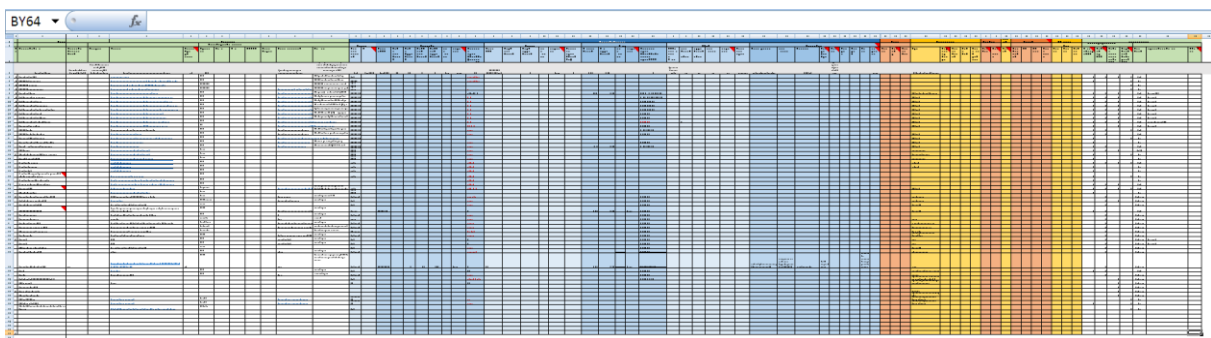
A screenshot of a Microsoft Excel spreadsheet titled 'Data Inventory Sheet'. The spreadsheet is very large, with many rows and columns. The columns are color-coded: blue for the first few columns, yellow for the next few, and orange for the last few. The rows contain text and numbers, representing dataset candidates and their metadata requirements. The bottom-right cell of the spreadsheet is labeled 'BY64'.

Figure 2: Data Inventory Sheet (vertically: dataset candidates, horizontally: metadata required)
– note that the bottom-right cell has address BY64

The schematic procedure for a data request is shown in Figure 3.

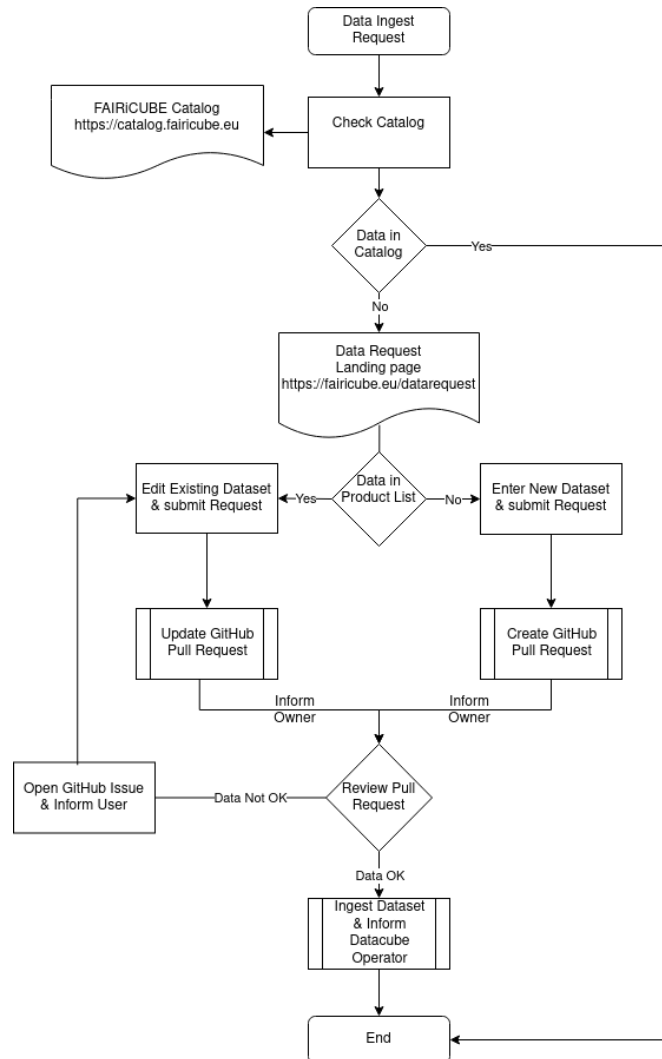


Figure 3: Data Ingestion Request Procedure

Once all metadata and data requirements are fulfilled, confirmed, and validated by the data requester, the ingestion handling partners will perform the merge and the request will be closed. Any discussion remains available.

Figure 4 shows the landing page of the WebGUI which can also be used to create new or update existing datasets. In Figure 4 a sample list of available datasets is shown, each with an Edit Button associated. Additionally, a *Link Button* is provided which directly links to the pull request in the respective GitHub repository, which also provides access to the *.json files and allows reviewing any changes.

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 (Figure 5), allowing the user to create a new record. 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.

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.

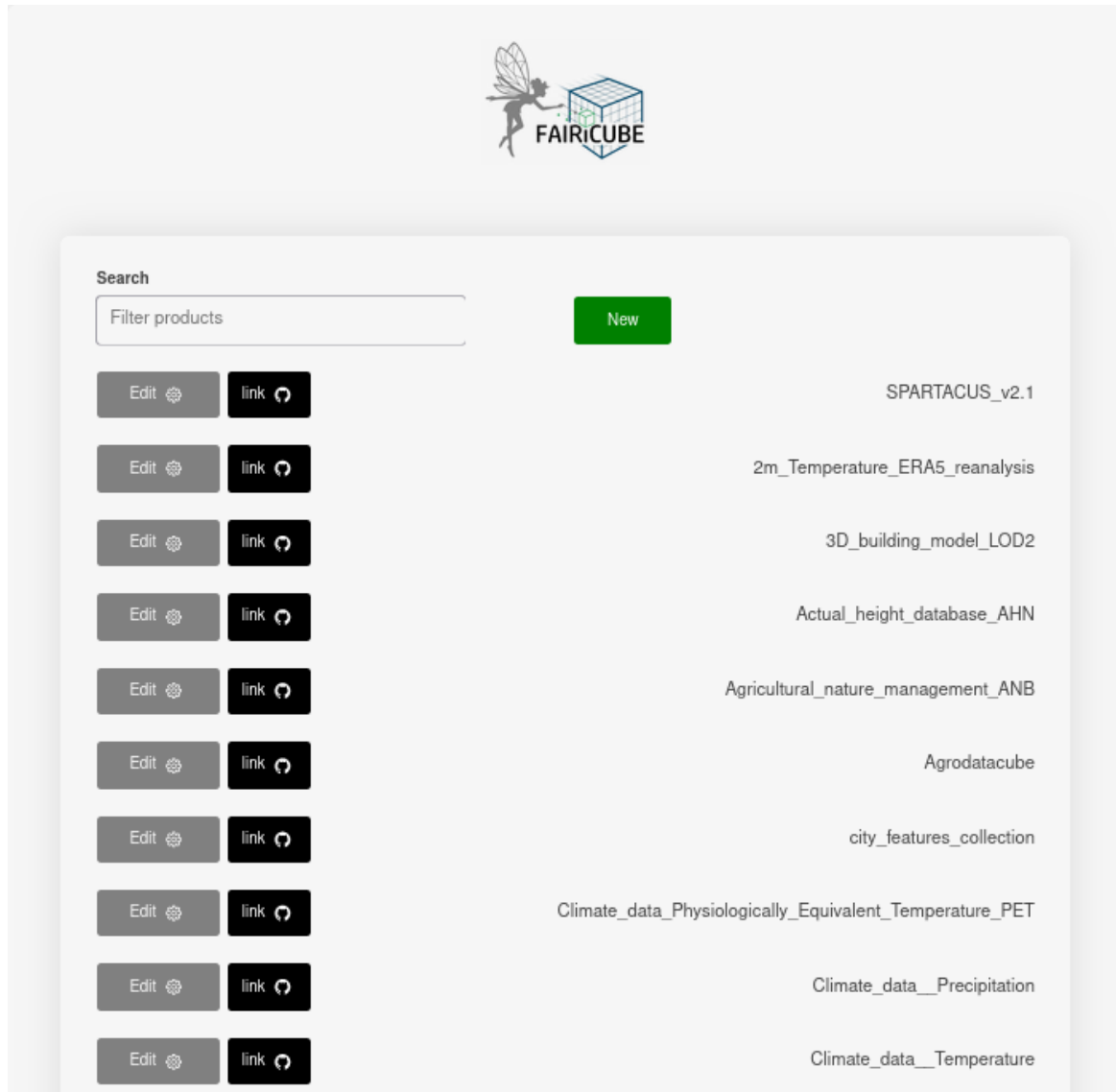


Figure 4: Data Ingestion Request Web GUI – Landing page



Figure 5: Data Ingestion Request Web GUI – First part of the data entry form

3.2 Geo Data Types

In FAIRiCUBE, three types of data are considered (aside from the metadata feeding the catalog, see WP4): gridded (i.e., raster) data, vector data, and point clouds. Potentially relevant information exists in all of these categories. A basic decision of the project is to map all such data to rasters, requiring a transformation of point clouds and vectors prior to ingestion, together with a decision on the desired target resolution. In some cases, the transformation can easily be done at ingestion time by adding some tool like GDAL into the pipeline. Figure 6 shows a schematic overview emphasizing the data flow perspective.

Rasterized vector layers have the nice property that they compress well, shrinking volume to a few percent of a comparable raster image with same resolution.

Point clouds get aligned to a grid by “snapping” points to the grid, which has to have a suitable resolution. In practice, this does not cause significant relocation errors as point cloud sources like LIDAR and SAR use sensor arrays which deliver relatively regular data point sets already. This task often is accomplished already with prefabricated products, for example with DTMs, so it is expected that in many cases gridded versions are readily available for download and ingestion.

As FAIRiCUBE partners do not work with point clouds, additional methods for aggregating point data to GeoDataCubes is required. An example of such datasets are the species Occurrence Cubes being created jointly with the B-Cubed project and their partner GBIF. As this exciting new biodiversity format requires integration of additional metrics when generating Occurrence Cubes, the existing modalities foreseen for LIDAR and SAR data do not apply. Custom generation routines are being implemented by the UC partners.

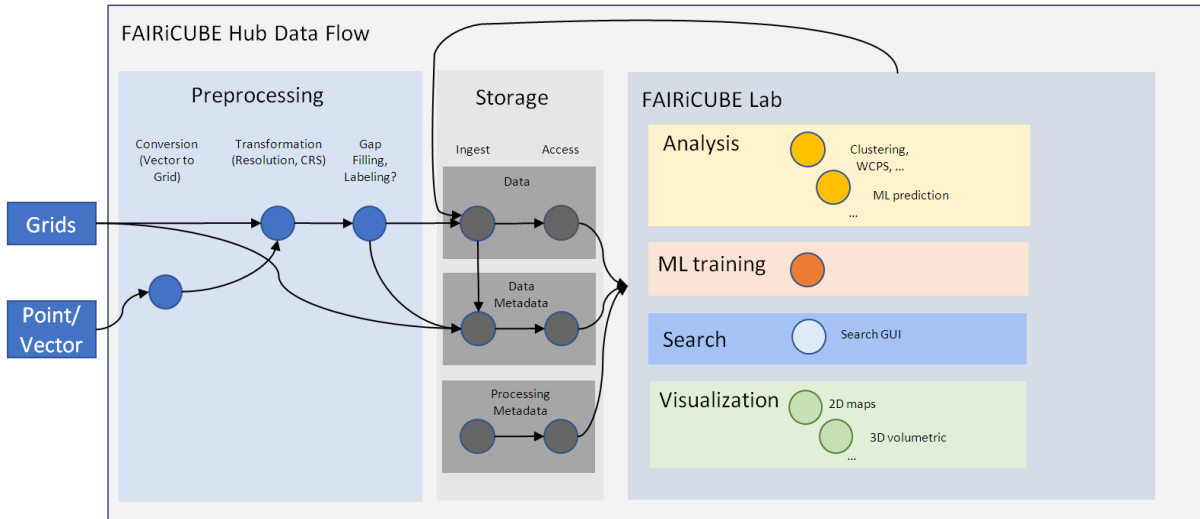


Figure 6: Schematic data flow from ingestion to use (possibly storing derivatives again)

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
FAIRiCUBE datacube documentation for rasdaman	https://github.com/FAIRiCUBE/data-requests (preliminary location, containing ingestion and more information in a simple-to-update wiki, in future to be integrated into KB as central documentation turnpike)
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: 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. In parallel, work is going on to automate ingestion from GitHub issues, earlier through a GitHub form and since recently through a Web GUI on top of GitHub.

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 issue is closed. In brief, the workflow is:

- Use case partners create a data-request indicating key parameters, such as source/origin of data, etc. through the WebGUI as specified in chapter 3.1;
- Rasdaman's team validates and asks back (via Pull Request communication) where necessary until enough information is provided for creating the datacube;
- The ingest process is created and executed;
- 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 procedure described, 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
- European Settlement Map
- Grassland Status
- Forest Type
- Imperviousness
- Tree Cover Density
- Water and wetness
- LGN
- Near Surface Air Temperature
- Corine land cover

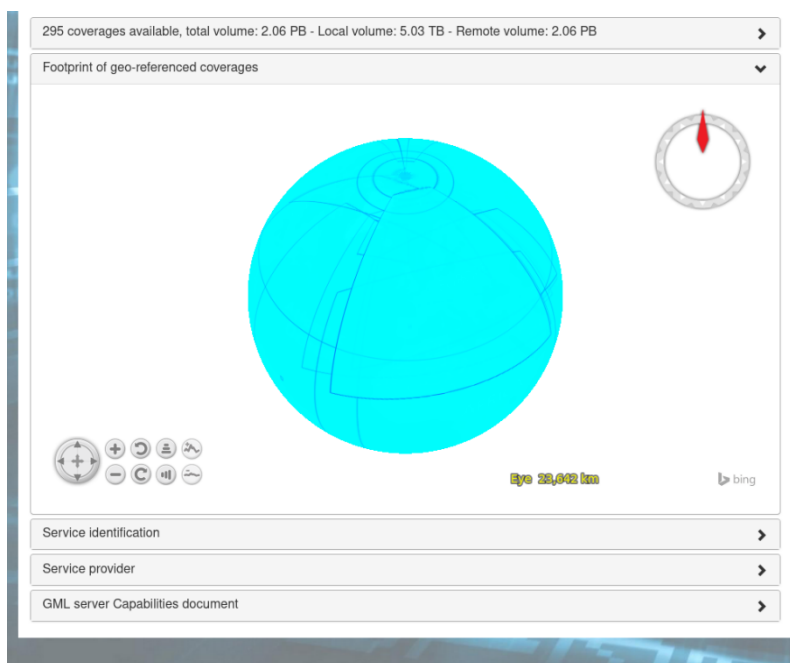


Figure 6: Footprints of rasdaman datacubes on the FAIRiCUBE server., including federated datasets

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

- Sentinel-1 GRDH (polarisations DV-VV and DV-VH)
- Sentinel-2 L2A (including derived products, like TCI)
- Sentinel-3 OLCI (all products L2)
- Sentinel-5p (all products, L2)

To ensure early availability in several datacubes, only an initial set of layers had been imported while some discussions on the interpretation of the timestamps were still going on (some metadata, such as multi-annual data from EEA, were not always unambiguously clear). The goal was to give the Use Case partners an early opportunity to test their algorithms. In the meantime, this discussion has converged, and a clear concept has been found (cf. [Temporal Semantics](#)).

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 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:

- the input data directory;
- 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



- 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”).

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 will be documented as a comment to the data request form. This way, the import action can be fully automated once a datacube has been defined.

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", "LA" ],
    "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 that way. Conversely, each datacube will contain a metadata element with a link to the corresponding STAC entry. This way, a mutual linkage is provided. Establishing this mutual linking will be done automatically via an API call provided by the catalog developers.

4.6 Problems Encountered

4.6.1 Data Problems

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.

- Information supplied for the datasets requested was not always sufficient for datacube creation, and the request forms practically never get filled in completely. Own investigation mostly helped, but not always. In any case, this caused significant extra effort.

Solution: it proved helpful to have a data kickoff and a data handover meeting where concrete information can be exchanged. Altogether, this remains extra work for WP5.

- 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 making the timeseries inhomogeneous; further, due to evolution of the sen2cor tool used by ESA, with processing parameters changing, some characteristics can change which are known, e.g., to affect the accuracy of previously trained models.

Solution: The solution chosen was a 2-stage approach: first, for each resolution occurring all data have been put into a single datacube each so that each such datacube contains a homogeneous resolution. Now users can choose to access the convenient virtual datacube which performs automatic resolution adjustment or - if interpolation is a concern - to access the exact, non-interpolated sub-datacube containing the region of interest; where resolution has been enhanced it has been done using the nearest-neighbour method. On top of that, a common virtual datacube was built containing all these datacubes offering them in a single resolution.

- In one dataset, the data semantics change over time: the imperviousness dataset for the year 2018 has percentage values from 0 - 100% whereas for 2006 it has numbers indicating 0-20%, 20-40 %, etc. Hence both time slices cannot be compared directly.

Solution: requires manual intervention and disclosing inhomogeneity to users.

- Categories sometimes have been extended with additional classes over time, in the same dataset.

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 imperviousness 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 access functionality was required going beyond what today typically is offered by rasdaman and other tools.

Solution: A detailed, advanced temporal selection capability is being developed currently offering substantially increased 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 this¹ 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 impact². 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.

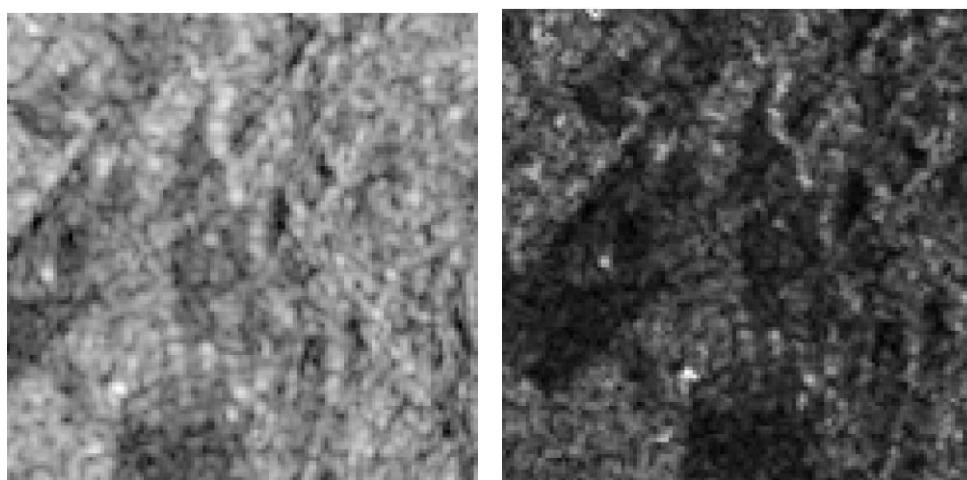


Figure 7: Sentinel-1 scene delivered by ESA with different processing parameters applied

4.6.2 Conceptual Problems

Occasionally problems surfaced which have not yet been solved in science nor standardization. In the attempt to advance the state of the art, CU addressed these. Another class of challenges can be summarized under the common "Analysis-Ready Data" topic of making access and processing of data easier by pushing technicalities behind the curtain; to this end, research was done to enhance the machine-readable semantics of OGC coverages aka datacubes.

4.6.2.1 Temporal Semantics

For the timeseries a special modelling approach was taken. Normally in such cases datacubes simply receive a time axis allowing for selection of timeslices. For example, CORINE land cover data are issued about every six years with validities 1986-1998, 2000 +/- 1 year, 2006 +/- 1 year, 2011-2012, 2017-

¹

https://cloud.code-de.org:8080/swift/v1/AUTH_279dbc97d5b5434fa8aeacf09c08c520/portal_prod/media/filer_public/a8/72/a872bbde-859a-44fe-a31d-3a33b8653471/20231025_reference_table_for_product_type_attribute_changes_in_the_new_catalogue_api_for_code-de_and_eo-lab.pdf

² 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



2018 ([source](#)). This results in “holes” of validity where no data should be delivered, according to the CORINE specification. In summary, a timeslice is not an atomic point in time (such as 2018-01-01T00:00) but it likewise does not stretch until, halfway up to the next timeslice. Rather, for capturing the full temporal semantics it is necessary to capture such individual validity periods. CU realised that advanced calendar functionality is needed for the temporal representation and retrieval of the datasets. This sparked research on refining the calendar date/time modelling functionality.

As an interim workaround until calendar queries became available, the datacubes were built with an integer (in OGC called “index”) axis allowing addressing by year (below visible by the name, such as “imperviousness_2006_index”). After completion (after the reporting period but demonstrated in the review) the datacubes were replaced with a genuine time axis and the new behaviour. In our example, this final object is named “imperviousness_2006”.

CU finally created a new concept called “period of validity” of pixels in the vicinity of their actual coordinate, combined with “query granularity”, and implemented it. After that, the rasdaman datacubes were updated and a webinar was given to the consortium. Implementation was finished after the reporting period of this deliverable, but a prototype was demonstrated in the February 2024 review. A journal paper has been accepted by ACM Transactions on GIS.

4.6.2.2 Multiple Datacube Views

In design discussions it was requested to access timeslices of the same object in different ways:

- as individual 2D objects, one for each time point, selectable without indicating a time slicing operation;
- As 3D x/y/t datacubes where the timeslice is obtained, as usual in a datacube, by indicating a time coordinate as the selection (i.e., slice) point.

This extra complexity was resolved by establishing 2D x/y objects, with the year in the name, and combining these into a virtual coverage offering the 3D x/y/t view including temporal subsetting. This is reflected by the multiple objects (in OGC speak: “coverage”) per datacube listed in Deliverable D5.1. By using the virtual coverage for combining these 2D layers into a single 3D object, data duplication could be avoided.

4.6.2.3 Datacube Catalog

CU provided a condensed, datacube-tuned catalog integrated with the rasdaman dashboard to give an overview of data holdings until the STAC integration could be accomplished (done at the time of this writing). This catalog allows for hierarchically structured datacube assets, displayed in a Web browser with folding and unfolding capabilities, as well as drilling down into datacube details. Figure 8 shows the catalog; note the listing of both Sentinel and FAIRiCUBE data, reflecting the datacube federation.

The catalog information can be retrieved in a structured, machine-readable manner and therefore enables an automatic feed of the central FAIRiCUBE STAC catalog.

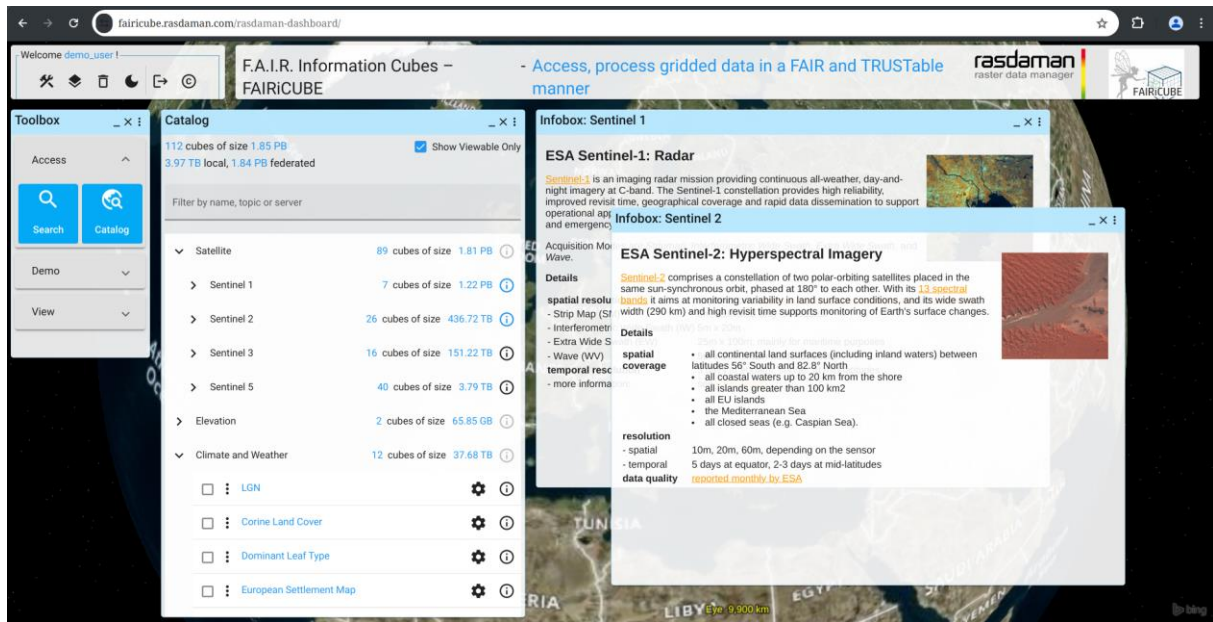


Figure 8: rasdaman catalog, integrated with dashboard



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 9 shows the workflow on how to upload data to the FAIRiCUBE Hub using the EOX platform.

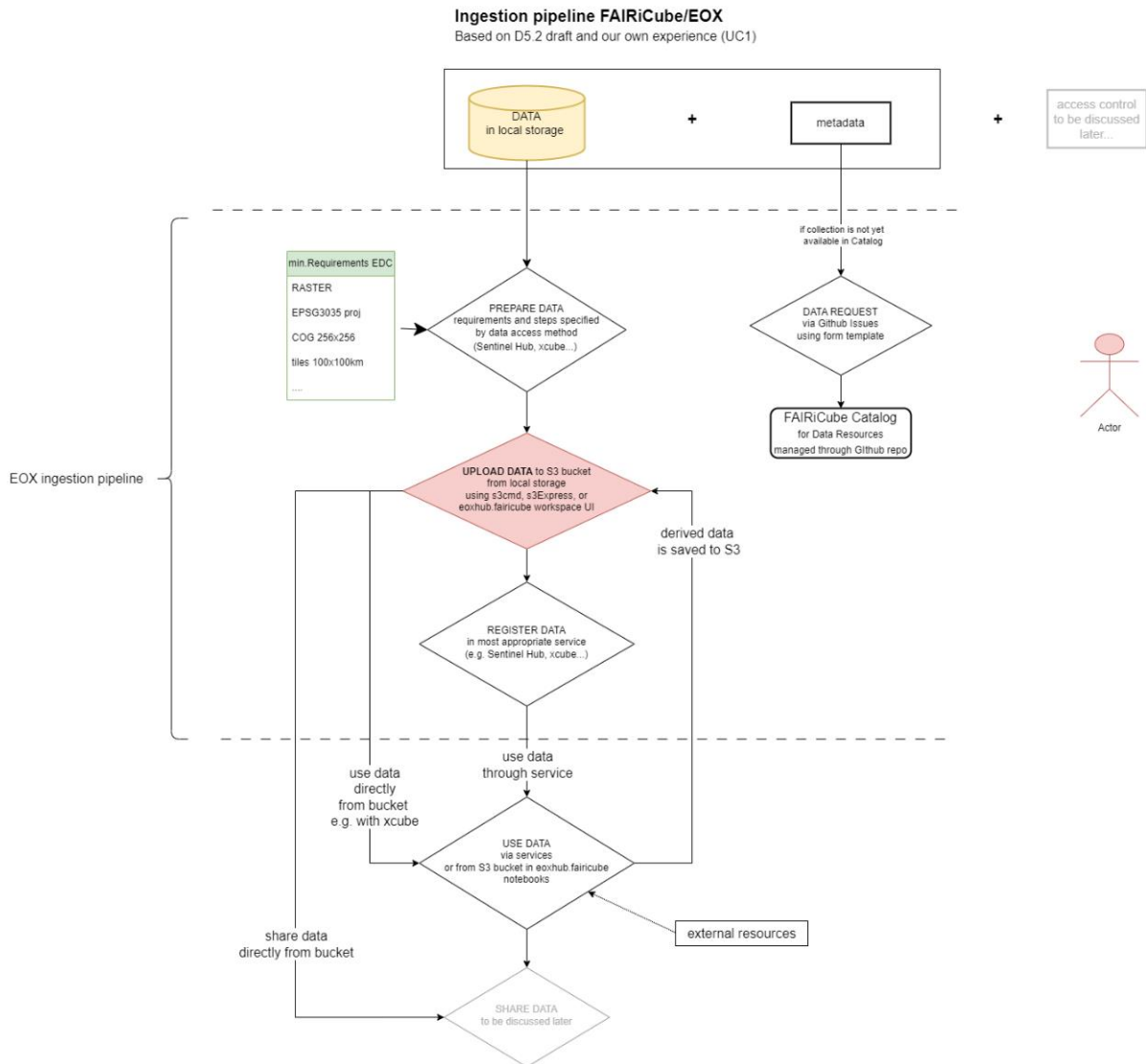


Figure 9: Data ingestion pipeline workflow using EOX-platform.

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 (<https://docs.sentinel->



[hub.com/api/latest/api/byoc/#converting-to-cog](https://docs.sentinel-hub.com/api/latest/api/byoc/#converting-to-cog)) or ZARRs (<https://docs.sentinel-hub.com/api/latest/api/zarr>) following the constraints and settings explained in the given links.

Concurrently with the data ingestion pipeline the metadata registration shall occur (see section 3.2 as well as "D4_2 Public Listing (Catalog) of FAIRiCUBE data resources.docx") for more details.

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 (<https://docs.sentinel-hub.com/api/latest/api/byoc/#a-note-about-cog-overviews-used-for-processing>).

Box 1 Upload data requirements

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 (EPSG:3857), any UTM zone (EPSG:32601-32660, 32701-32760), or Europe LAEA (EPSG:3035).

Max. number of bands:

- 100

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
```



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 has to 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 10 below.



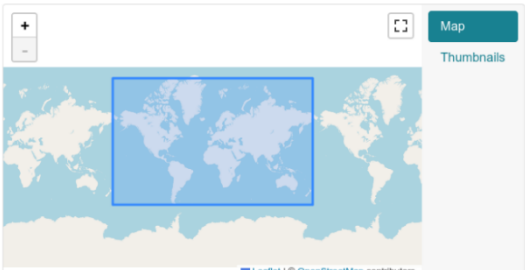


ESA WorldCover

in stac-fastapi

[↑ Up](#)
[Collection](#)
[Browse](#)
[Search](#)

[API](#)
[Source](#)
[Share](#)
Language: English



Description

The European Space Agency (ESA) WorldCover is a global land cover map with 11 different land cover classes produced at 10m resolution based on combination of both Sentinel-1 and Sentinel-2 data. In areas where Sentinel-2 images are covered by clouds for an extended period of time, Sentinel-1 data then provides complimentary information on the structural characteristics of the observed land cover. Therefore, the combination of Sentinel-1 and Sentinel-2 data makes it possible to update the land cover map almost in real time. WorldCover Map was first produced for 2020 using v100 of the algorithm and later for 2021 with v200 algorithm. Due to the different algorithm versions, it should be noted that changes between WorldCover map 2020 and WorldCover map 2021 are a result of both actual changes in land cover and in the used algorithm. WorldCover map is provided as part of the

[Read more](#)

Asset

- Thumbnail THUMBNAIL PNG

Additional Resources

About this resource

- [Website describing the collection](#)
- [Details about running Evalsctrpts](#)
- [Nomenclature mapping - band values WorldCover labels](#)

Processing instructions/code

- [Evalsctrpt to generate ESA WorldCover Map imagery](#)

Collection

[Data Request catalog](#)
A catalog that contains requested datasets and Euro Data Cube datasets.

Provider

Sentinel Hub PROCESSOR

General

License	License
Keywords	<ul style="list-style-type: none"> - land cover - agriculture - biodiversity - worldcover - derived data - open data
Time of Data ends	1/1/2021, 12:00:01 AM UTC
Time of Data begins	1/1/2020, 12:00:00 AM UTC
Datasource Type	byoc-0b940c63-45dd-4e6b-8019-c3660b81b884

Data Cube

Dimensions					
Id	Type	Axis	Extent	Values	
t	temporal	n/a	2020-01-01T00:00:00Z – 2021-01-01T00:00:01Z	n/a	
x	spatial	x	-180 – 180	n/a	
y	spatial	y	-56 – 83	n/a	
bands	bands	n/a	n/a	- Map - dataMask	

Powered by STAC Browser 3.1.0

Figure 10: 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 pull request. 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 Copernicus 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. Therefore, the WP5 partners believe that a first stable state has been achieved. Next steps include:

- Ingest more data, as always driven by Use Case priority;
- Link datacubes with the metadata records in the STAC Catalog (done at the time of the deliverable update);
- Investigate to what extent the two (technologically quite different) stacks of rasdaman and EOX can be combined under a common hood, to enable data sharing;
- Ingestion of point and vector sources, including the provision of reusable algorithms;
- Enhancing rasdaman to provide comprehensive calendar functionality on time-type axes (done at the time of the deliverable update);
- Enhancing rasdaman to store additional information in the range type. During the reporting period this was prototyped with some objects at the time of the deliverable update. At the time of this writing, following thorough verification it will be propagated into all coverages.