A Quick Guide for Databus and MOSS

A STEP BY STEP GUIDE HOW TO REGISTER DATA TO DATABUS AND MOSS

July 19, 2026

A step by step guide how to register data to Databus and MOSS

Carsten Hoyer-Klick [1], Jan Forberg [2], and Sebastian Hellmann [2]

[1]German Aerospace Center, Institute of Networked Energy Systems, ROR,
Curiestr. 4, 70563 Stuttgart
[2]InfAI, Institut für Angewandte Informatik (InfAI) e. V., Leipzig UniversityROR
Goerdelerring 9, 04109 Leipzig

Published by

German Aerospace Center, Institute of Networked Energy Systems, ROR,
Curiestr. 4, 70563 Stuttgart

Acknowledgements

The authors of this article have used various preparatory works from the NFDI4Energy to create this portrait, and references have been made where possible. Thanks to all those who are not named. The authors would like to thank the German Federal Government, the German State Governments, and the Joint Science Conference (GWK) for their funding and support as part of the NFDI4Energy consortium. The work was funded by the German Research Foundation (DFG) – 501865131 within the German National Research Data Infrastructure (NFDI, `www.nfdi.de`).

License

Table of Contents

General Information

Summary

This document gives a short introduction into the tools Databus and MOSS and how they can be used in mulit-domain research data management.

Deliverable within NFDI4Energy

This document is part of task area 8 ”Data and Core Services”. The tools Databus and MOSS have been enhanced within the flex funds project ”SOMMER - Search over Multilayer Meta Data in Energy Research”. It is a user guide which helps users to use the tools for meta data publication and data discovery.

1 Introduction

Databus and MOSS (Metadata Overlay Search System) provide an advanced, flexible cataloging system designed to meet FAIR (Findable, Accessible, Interoperable, and Reusable) [1] data management standards across domains. By creating a unified metadata registry, Databus and MOSS enable researchers to add, refine, and search rich metadata records. This ensures data accessibility and interoperability across diverse repositories and research fields.

1.1 System Capabilities and Concept

The system enables the deployment and population of an interdisciplinary, multi-domain catalog (using Databus)[2] across multiple dataset repositories relevant to a research field. This catalog, enriched with a metadata annotation system and a search interface called MOSS (Metadata Overlay Search System)[3], allows researchers to

1. flexibly add comprehensive metadata records and
2. search and discover data relevant to their field and their current research.

Both Databus and MOSS are conceptually and technically mature tools that have been in active use for several years (in particular in energy system research and the Open Energy Platform[4]). In the following, we describe the primary architecture and the four key concepts that form the foundation of this system. Figure 1 shows the main architecture (using example repositories and metadata schema from energy system research). The data itself stays in various types of data repositories (Virtual Bus). They each follow their own standards. Their data sets are registered on the Databus to create an ID and therefore a root of a metadata graph. The MOSS is used to attach multiple metadata subgraphs to each of the IDs. Essential metadata is available as SPARQL and additionally each subgraph is indexed in a search index within the MOSS. The user mainly interacts with the MOSS. When searching for data, the user will get the available metadata and a download link to the data in the respective repository of the data.

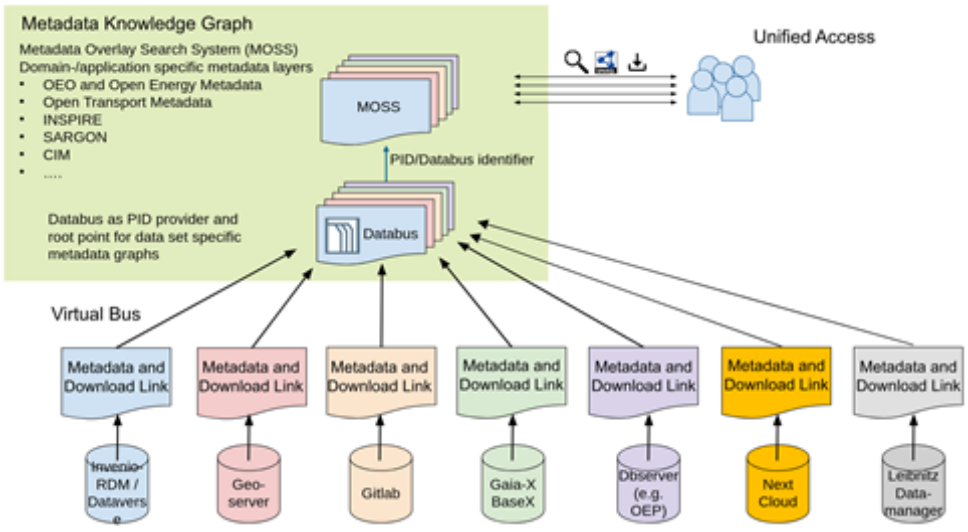

Figure 1: Architecture of Databus and MOSS.

1.2 Multi-Repository Catalog (Databus)

Currently, research data is already published across a wide range of online portals, repositories, platforms and websites. One of the main goals of research data management is to adequately catalog this existing, distributed data in a „registry for metadata for digital objects (DO)“. We have identified a number of typical repositories and repository software which are currently used in energy system research:

1. Zenodo as a publicly available platform,
2. Repositories based on InvenioRDM, the open-source software used in Zenodo, that can be deployed at institutions (e.g. it is used by DLR) or other tools as Harvard Dataverse,
3. Git-based repositories such as the public GitHub platform by Microsoft or the self-deployed GitLab,
4. Geoserver, a custom landing page for energy-related geo data,
5. Open Energy Platform and its databases,
6. Cloud-based repositories such as NextCloud and Google Drive,
7. CKAN-based repositories such as the Leibniz Data Manager (LDM).

Over the last 8 years, starting with the work on DataID [5], we investigated the capabilities of these repositories and many others in order to create a unified approach to catalog the contained data under FAIR aspects. Development on the Databus catalog system started around 2018. Although all the above repositories share the same goal of providing online access and metadata, heterogeneity is extremely high in the details. Each of the systems implements IDs, data lifecycle, metadata, API access and low-level data handling in a different way. In previous projects, we have successfully shown that the Databus Ontology is able to provide a minimal, homogeneous model to accurately capture and integrate the various models and create a consistent, interoperable Metadata Knowledge Graph. Two examples of such heterogeneity are:

1. versioning is done in Zenodo by assigning new IDs to each version, while git-based solutions use commit hashes, but provide stable IDs pointing to the latest version. Other solutions like NextCloud work in overwrite mode, where the retrieved DOs for a ID change.
2. other incompatibilities are found on a lower level such as metadata describing the format with e.g. “csv.gz” could be described as “csv” or “gz” file, or even differences in measuring filesize, in date formats and in hashsums.

To clarify the benefits of Databus for research data management:

1. Databus has an established track record as a metadata registry, supporting platforms like Zenodo/InvenioRDM, HuggingFace, GitHub/GitLab, NextCloud/GoogleDrive, FTP file servers, and notably, the Open Energy Platform (see [6]). As the engineering for these integrations is implementation-specific, the work would have to be repeated (mostly from scratch) within research data management.
2. The Databus (as well as MOSS) is committed to the same open standards as the tooling of many research data management approaches: Knowledge Graphs, SPARQL, Linked Data, DCAT & DCAT-AP, SHACL, JSON-LD, RDF. Thus, it is highly interoperable and complementary to the already developed services that rely on the same standard as namely LDM, ORKG, TIB Terminology and ID services.

1.3 Metadata Annotation for Energy Systems Research (MOSS)

Energy system research is rooted in many different scientific domains and the data that is used in energy systems modeling comes from very different domains such as engineering, remote sensing, meteorology, economics, social sciences, geography among others. Documenting data is difficult as each of the above-mentioned domains use different standards and different vocabularies. Building a metadata schema which covers all the different aspects may prove to be difficult and may not always be compatible with the different domain standards. We propose an easier way to handle multi-domain metadata by attaching multiple metadata schemata of different domains to a single data set. Our approach is

consistent with the construction of specific, minimal metadata "modules" for specific purposes. This eases the discovery of the data while searching for data with different domain perspectives. The MOSS (Metadata Overlay Search System), which has been developed during the LOD-GEOSS project [6], is an addition to the Databus catalog system [7]. The Databus provides an identifier (ID) with basic metadata and this ID serves as a root to a metadata graph for each of the registered data sets. Different "modules" of metadata information can be configured in the MOSS. Each module contains a domain-specific subgraph that can be added to describe the data resource from a multi-domain perspective. The MOSS supports the use of ontologies for a unified data vocabulary within the research domains, e.g. the Open Energy Ontology [8]. The modules are curated by the administrator of the MOSS, therefore the adherence to domain standards can be enforced in opposition to user extensions on the catalog which may vary according to the users using it. MOSS is a central entry point for users managing their data residing in different repositories. Rich metadata such as the Open Energy Metadata Schema 2.0 (JSON-LD) can be used to register data, as it contains all information for a registration in the MOSS and Databus. The MOSS and Databus therefore serve as a flexible, multi-domain and multi-repository metadata catalog and an entry point for data users where they can start to catalog and describe their data. In this manner, Databus and MOSS can bridge data repositories on the data and metadata level, and enable the development of uniform data catalogs. This can consolidate all the metadata from the different data silos and data domains by making them searchable on a single platform.
To clarify the benefits of MOSS for research data management:

1. MOSS allows users to annotate data sets from many repositories in a central place that can later be part of a larger platform.

2. Initial metadata modules will focus on existing standards such as OEO and Open Energy Metdata 2.0, new modules can be added over time including an ORKG layer or others.

3. Subgraphs are versioned in Git allowing traceability and versioning of metadata edits following a Wiki workflow.

4. Adherence to RDF and SPARQL standard makes the collected metadata in MOSS interchangeable with LDM, federated search and other Knowledge Graph approaches.

1.4 Search over Multi-Layer Graphs (MOSS)

A good search is always highly contextual and geared towards the needs of the targeted audience. With MOSS, we provide a practical workaround to the No Free Lunch in Search Theorems (cf. [9]) by enabling each user community to define and deploy its own metadata standards in modules and then configure via SPARQL queries which parts from these metadata subgraphs are indexed in the search and exploited in the user interface via facets, plaintext/keyword search, ontology-based search with autocompletion or hierarchical browsing of ontologies and other search paradigms. This modular approach enables MOSS to avoid the inefficiencies of a one-size-fits-all solution by aligning the search index directly with the unique structure and semantic needs of each community's metadata and thus deliver optimized search quality and efficiency within their context — effectively addressing the challenge posed by the No Free Lunch theorem.
To clarify the benefits of the search in MOSS for research data management:

1. the successful development of a user-accepted search over multi-domain, multi-repository datasets can increase visibility and acceptance of research data management within the research community, instead of perceiving it as a burden.

2. the employed methods and tools aim to have an impact on the whole research data management as a search solution that can be adopted and re-used across different domains.

2 Creating users

While searching and retrieving data from the Databus and MOSS, one does not need a user account. However, publishing requires a user account on either platform. On the Databus, the ID (Persistent Identifier) is always connected to a user account (see section 3 on the structure of the IDs). On the MOSS side the user account is used for granting write access and to track changes in the database to specific users to record the provenance of the data.
Databus and MOSS distinguish between login credentials and accounts which are used for changes or publishing. Therefore, one login may be tied to multiple accounts on the Databus. A single login can be connected to multiple accounts on the Databus. The different accounts are an element to organize the data sets on the Databus. Accounts, groups and artifacts can be used to group and organize data and the IDs which are generated by registering data on the Databus (see section 3).

2.1 Databus

The first step for publishing data is to create a user account. This can be done be clicking on the login button and then creating a user account (see figure 2). The open energy Databus is connected to the NDFI authentication service, therefore existing institutional accounts can be reused.

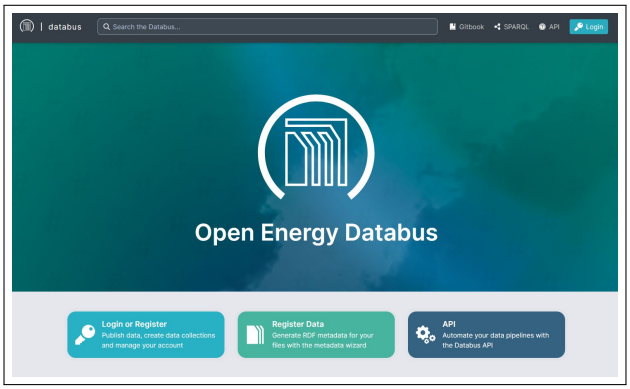

Figure 2: Login page of the Databus. On the top is the search bar. The login button is at the top right corner.

2.1.1 Personal account

Once the login has been created, the next step is to create a personal user account. By clicking the account name on the top right and on ”My Accounts” an empty ”User Settings” page will appear. On the right hand side there is an ”Add account button” which needs to be clicked as can be seen in figure 3.

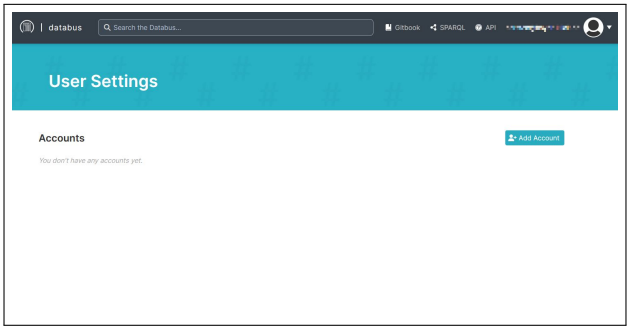

Figure 3: Initial user settings page with no accounts created.

Once the button is clicked, please enter your name and a label. The ”Name” is the string which will later be used as an account name in the IDs. The name should only contain alphanumeric characters and should be between 3 and 15 characters long. The ”Label” is a human readable explanation. You can add your real name with whitespaces. An example can be seen in figure 4.

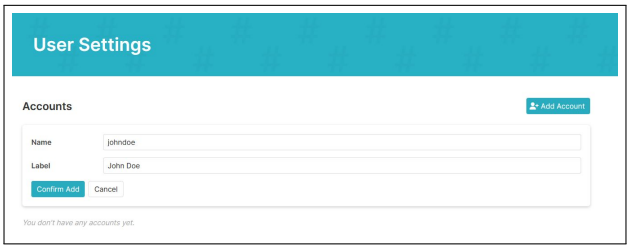

Figure 4: Creation of a personal user account on the Databus.

Finally click on ”Confirm Add” to create the personal user account.

2.1.2 Virtual accounts

In addition to your personal user account, multiple virtual accounts can be created. These can be used to organize the data IDs e.g. to your organizational structure. As an example, we create an additional NFDI4Energy account. Multiple secretaries can be added to the virtual accounts, to be able to manage them from different users in an organization.
Virtual accounts can be created from the ”User Settings” page, which is accessible from the ”My Accounts” menu at the user icon on the top right. Just click another time on ”Add Account” to create an additional virtual account. It needs to be filled with a name and a label as the personal account. An example can be seen in figure 5.

Figure 5: Creation of a virtual account e.g. for the NFDI4Energy organization.

Click on ”Confirm Add” to finally create the virtual account.

2.2 Getting an API key

If you want to use the API to register data to the Databus, you need to create an API key first. The API keys can be created and accessed via the ”User Settings” page. Again go to ”My Accounts” and find all of your accounts. On the right hand side of your accounts you will find the gear wheel to access the account settings as it can bee seen in figure 6.

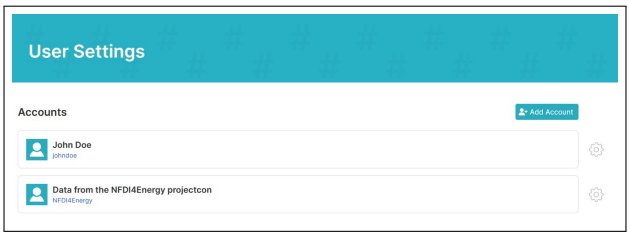

Figure 6: Accessing user settings of the different accounts.

Clicking on the gear wheel takes you to the account settings. The first block is about the public profile, which can be editied as needed. The second block is about the API keys. To create an API key, a name for the key has to be entered in the ”Create API Key” section and then press ”Create”. Figure 7 shows an example to create an API key named ”NFDI4Energy” for the NFDI4Energy virtual account. The API keys can be used in scripts as shown in the curl example on top of the API Key section.

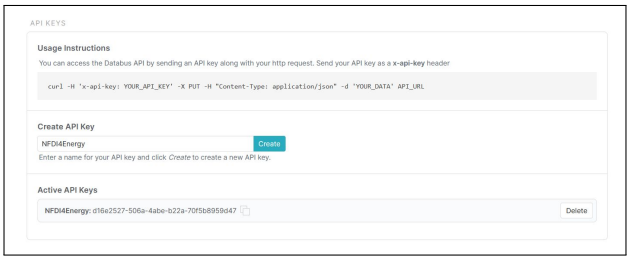

Figure 7: Creation of API Keys on the Databus.

2.3 MOSS

Similar to Databus, MOSS also needs a user account for data posting. The user account is mainly used to prevent unauthorized postings and to track changes to the stored data on MOSS, as the data sets are editable. Changes to the database are tracked in a git repository.

2.3.1 Personal account

If a user is logged into MOSS for the first time, the user is only named ”User” on the top of the page, as it can be seen in figure 8

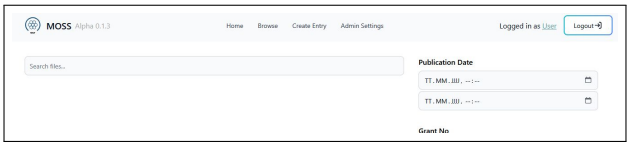

Figure 8: Starting Page of MOSS with no user initialization.

By klicking on the ”User”, the user settings page appears and a user name can be set. It is saved by clicking ”Save Profile”, see figure 9.

2.3.2 Getting an API key

An API key can be created by clicking on the ”Keys” tab on the left hand side. Just enter a name for the API key and click ”New API key”. The key itself will not be stored on MOSS and cannot be retrieved later. Therefore, it should be stored in the safe place immediately. An example can be seen in figure 10

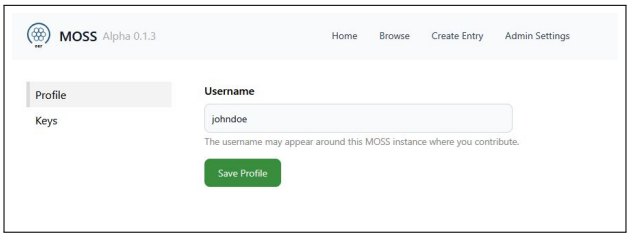

Figure 9: Setting the account name in MOSS.

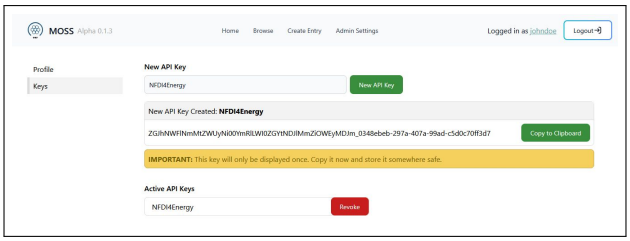

Figure 10: Generation of a new API Key in MOSS.

3 The Structure of the Databus IDs

The aim of registering data to the data bus is to generate an identifier which can later be used as the root of a meta data graph. Once the data is registered it gets a Databus identifier which has a fixed syntax:

```
https://domain.tld/user/group/artifact/version/
```

The meaning of the single elements are:

- **user:** this is the registered user name of the user at the Databus. Therefore all identifiers on the database are referenced with a distinct user. Once a personal user account is created, the user can create additional accounts which may be associated to organizational structures of the organization or projects.
- **group:** groups can be used to structure different datasets. The group may be a department of an institution, or a project or a model or an experiment the data belongs to.
- **artifact:** this is the actual dataset. It should be short descriptive name for the data. A dataset can consist of several files which belong together. Artifacts are usually something which has the same data life cycle, e.g. all the files in a data set should be produced by the same model run or measurement campaign.
- **version:** datasets can have different versions, as they e.g. get updated. Using the date of publication in the form YYYY-MM-DD is common way for naming versions of dataset.

Data publishers can freely group their data into users, groups and artifacts. Any kind of structure can be generated which matches the way of working in different organizations.

4 Creating new Modules at MOSS (admin)

The creation of new metadata modules at MOSS needs administrative privileges at MOSS. The ensures that the metadata modules which can be used to annotate data are curated. New metadata modules can be created by going to the ”Admin Settings” on the top of the MOSS page. The page shows the existing modules, which can be edited or new modules can be created (see figure 11).

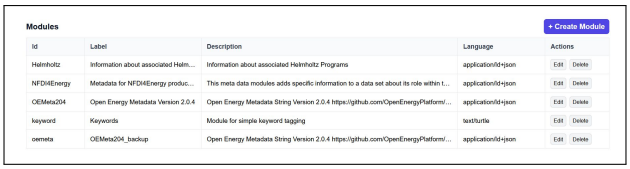

Figure 11: Listing of existing metadata modules and option to create new modules.

To create a new module, click on ”+Create Module”. A module needs a name, which is a charcter string, a human readable label and a description. The language is for the selection of the serialization of the graph data. Figure 12 shows as an example the creation of an NFDI4Energy module.

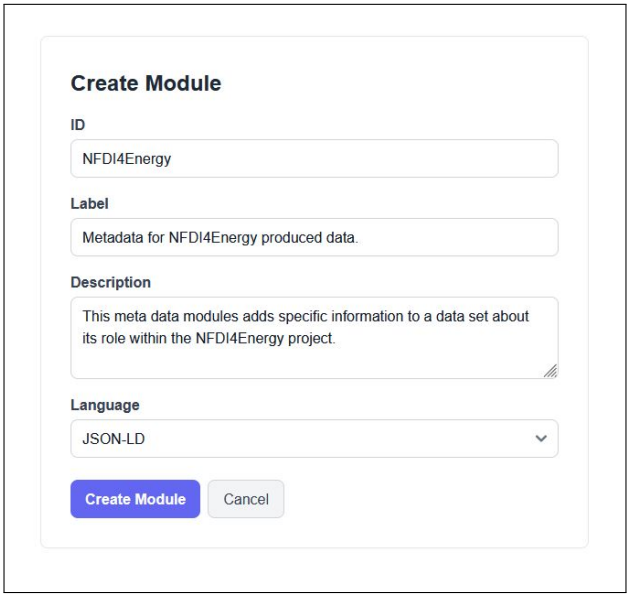

Figure 12: Initialization of a sample NFDI4Energy module.

Once the module has been created it can be edited by clicking on the module in the modules list. The there are five tabs on the left hand side (for a JSON-LD module) to configure a module. In the following example we will look at a JSON-LD example of a meta data module. The ”General” tab is to edit the information which has been entered at the creation of the module. The ”Indexer” tab configures SPA-QL queries which are used to fill the tables of the search indexer. In the ”SHACL” tab, SHACL constraints for the validation of the entered data can be added. The ”JSON-LD” tab can be used to host `@context` information to be used in the JSON-LD Metadata directly at the MOSS web page. MOSS will ensure that it is delivered as ”application/LD+JSON” MIME type. The ”Template” tab is to host an empty meta data template for this module, if the meta data will be added manually to MOSS. When editiing the information, the appropriate files may have to be created by clicking the ”Create... ” button.

4.1 Create a Template

As an example we take a very simple metadata module, which adds the information to which task area of NDFI4Energy a data set belongs and which participating institutions were involved. The template could look like this:

```
{
    "@context": "https://moss.openenergyplatform.org/api/v1/
       NFDI4Energy/context.jsonld",
    "taskArea" : "",
    "NFDI4EnergyPartner": []
}
```

Listing 1: A simple metadata module to annotate project information

It is very simple, it just links the JSON-LD context via the ”@context” key (see also section 4.4) in line 2 and adds two keys for the task area and the NDFI4Energy Partners in lines 3 and 4. The latter can be a list, if multiple partners were involved.

4.2 Create an Indexer

This feature is paused and will likely be deprecated.
With an indexer you can specifiy how entries for this module will be written to a Lucene index. The Lucene indexer used to be the main driver of the search-strategy over MOSS instances and has been as such replaced by search facets paired with a SPARQL query generator.
You can however still use this feature to create a powerful search over the contents of the MOSS instance e.g. for use in other applications. The search indexer runs on the DBpedia Lookup project, which is documented here [TODO: Enter link]. Its core idea is to create key-value pairs for an inverse index by running specific SPARQL queries on RDF data.

4.3 Create SHACL Constraints

While the template helps to give a clearer understanding how a MOSS entry should look like, it is usually best to validate inputs to ensure that the information provided is complete. With the metadata being in an RDF format, our go-to validation language is SHACL. SHACL constraints (also called SHACL shapes) are a way of defining a more formal schema that data inputs can be validated against.
If you add SHACL shapes to your module, all entries created for this module will be validated against the shapes. Metadata can only be saved when conforming to the specified shapes.
[TODO: ADD EXAMPLE]

4.4 Create a JSON-LD Context

When creating a module in the JSON-LD syntax, it can be useful to create a JSON-LD context document tailored to the fields of your specific module to make the JSON-LD metadata more lean and readable. JSON-LD context documents act as a mapping between plain JSON and RDF. The JSON-LD context could be based on schema.org:

```
{
  "@context": {
    "schema": "http://schema.org/",
    "taskArea": "schema:Action",
    "NFDI4EnergyPartner": "schema:Organization"
  }
}
```

Listing 2: JSON-LD context for the simple meta data module

5 Registration of data

In general there are two different ways to register data to Databus and MOSS. It can be manually entered in the forms of Databus and MOSS or the data can be sent via the API as JSON-LD or turtle. The registration of data is done in two major steps:

1. construction of the ID on the Databus. The construction of the ID consists of up to three substeps (depending on which elements are already available):
   1.1 registration of a group,
   1.2 registration of an artifact,
   1.3 registration of a version,
2. registration of the enhanced rich meta data to MOSS.

The construction of the ID on the data bus just needs minimal meta data as names, human readable titles, abstracts, descriptions, licenses and the download links to the actual data. Usually this information can be copied from the rich meta data which is used for the catalog in MOSS in the second step.

5.1 Manual registration of data

The manual registration of data can be started the top user menu, by clicking on publish data (see figure 13

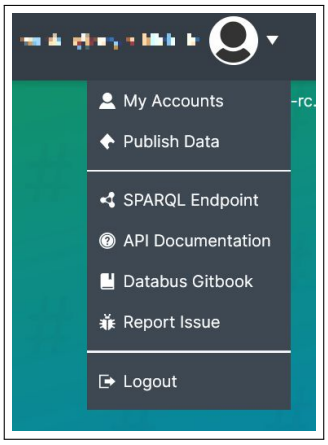

Figure 13: Pubish data on the Databus menu.

From there, groups, artifacts and versions can be created by clicking on one of the menu items as showsn in figure 14.

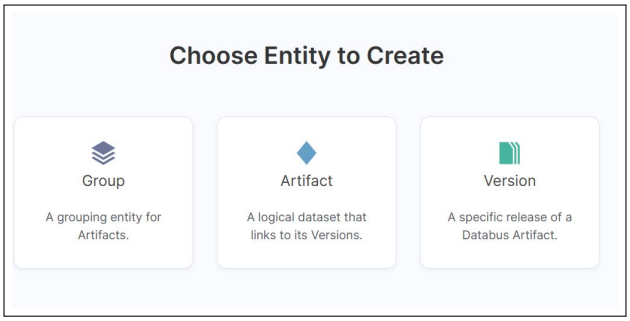

Figure 14: Selection of entry types on the Databus.

5.1.1 Creating a Group

A group can be created in the group section by the publish data wizard. An example is shown in figure 15.

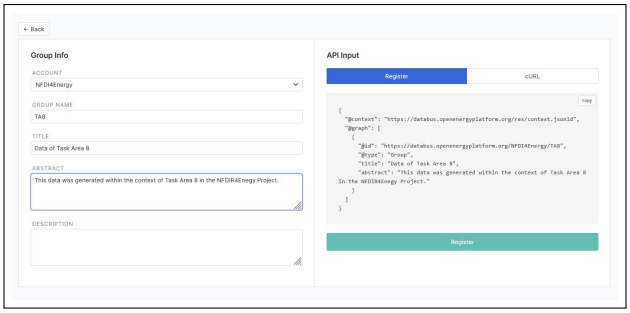

Figure 15: Creation of a group on the Databus.

The group needs to be connected to an account. In this example we choose the NFDI4Energy virtual account. The group also needs a name, we choose ”TA8” for task area 8. Finally it needs an abstract, which also can be entered. A more detailed description is optional.
The right hand side shows the meta data in JSON-LD as it is created by the form. There is also the option to show a `curl` command which can be used to register the data. Here you need to select an API key for the command.

5.1.2 Creating an Artifact

The creation of an artifact is very similar to the creation of a group. An example can be seen in figure 16.

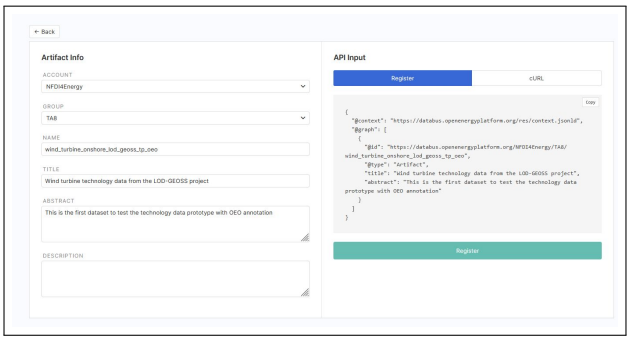

Figure 16: Creation of an artifact on the Databus.

In the beginning the artifact needs to be linked to a group. Here we choose the ”TA8” group we created before.

5.1.3 Creating a Version

Creating a version is a bit more complex, as it needs additional information. In particular license and attribution information. The form therefore has three pages.
First the corresponding group and artifact have to be selected, as it can be seen in figure 17. For the version name we use semantic versioning and take the publication data of the data set, which in this case is June 29th, 2021. Titles, Abstract and Description are copied from the artifact.
Figure 18 shows the second page of the form. Here a license and attribution has to be entered. The ”Derived from” field is optional. Here you can link to data which has been used to generate this data set.

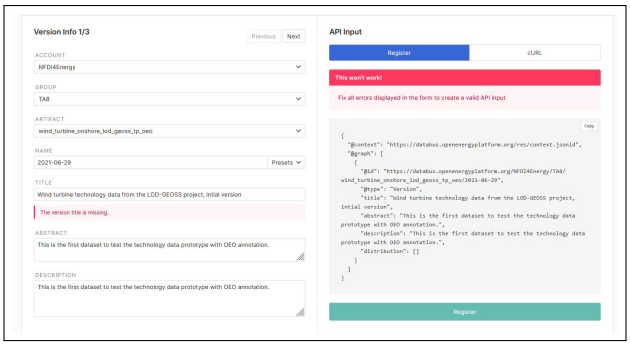

Figure 17: First page of the create version form.

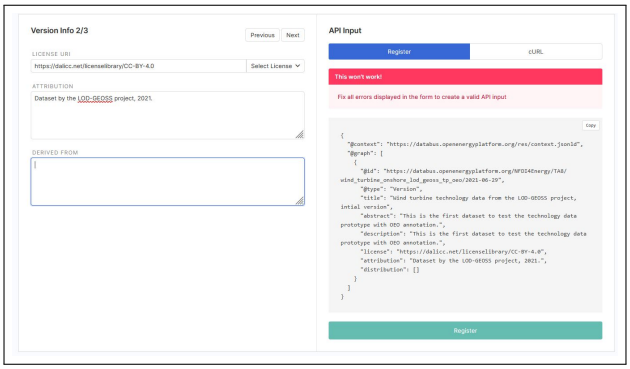

Figure 18: Second page of the create version form.

The final third page is for the URLs to the actual data. Here we choose a dataset from the Open Energy Platform, which has been produced in the LOD-GEOSS project. The dataset can be found at `https://openenergyplatform.org/dataedit/view/model_draft/wind_turbine_onshore_lod_geoss_tp_oeo`.
The URLs to the files can be added by the ”Add File URLs” button. The URLs can be entered in the field below and by clicking on ”Add URLs” the Databus will access the files and calculate their file sizes and a checksum. If successful, they will be added to the list at the bottom. By clicking on ”Register” the registration can be finalized. A success message will appear below the button once the registration has been done. Otherwise an error message will appear.
Now the ID is complete and it can be used for enhanced metadata graphs in MOSS.

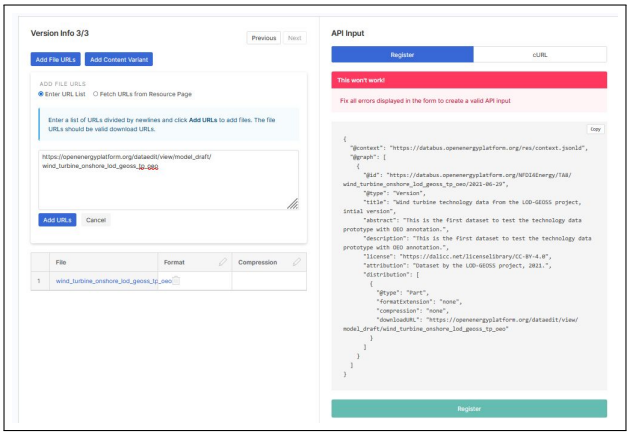

Figure 19: Third page of the create version form.

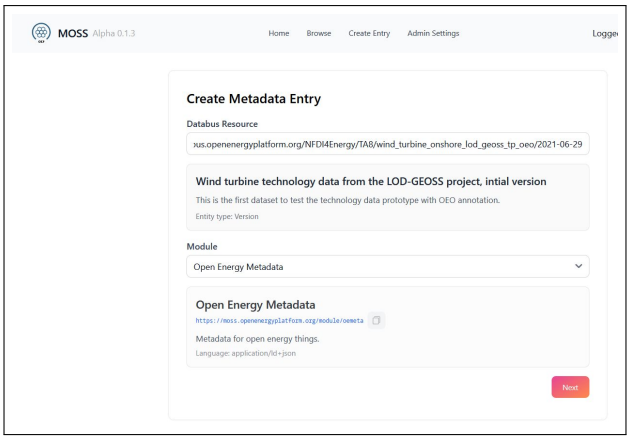

Figure 20: Creation of a rich meta data entry on MOSS.

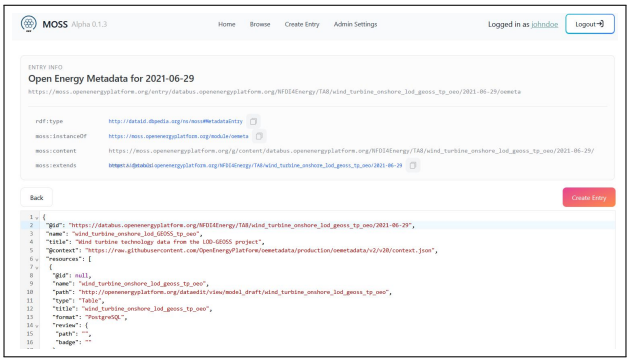

Figure 21: Pasting of meta data information into the MOSS Entry.

5.1.4 Creating an Entry at MOSS

The final step is to create a rich meta data entry on the MOSS. This can be done by going to ”Create Entry” on the MOSS homepage. You need to be logged in, to be able to create entries. The creation needs a valid Databus ID. Once it is entered, some basic information will be retrieved from the Databus and displayed. Next a meta data module needs to be selected. Again some basic information on the meta data module will be displayed. This can be seen in figure 20. By clicking on ”Next” the entry will be created.
The rich meta data information can now be pasted into the meta data filed, as shown in figure 21. The information on top of the page shows some meta information MOSS and Databus use to connect the entries. It cannot be altered.

5.2 Data registration using the API

5.2.1 Creating a Group

To create a group via the API, first a JSON-LD payload containing the group information has to be generated. We take the same group as in figure 15, which is shown in listing 3.

```
1  {
2    "@context": "https://Databus.openenergyplatform.org/res/context.
       jsonld",
3    "@graph": [
4      {
5        "@id": "https://Databus.openenergyplatform.org/NFDI4Energy/TA8
           ",
6        "@type": "Group",
7        "title": "Data of Task Area 8",
8        "abstract": "This data was generated within the context of Task
             Area 8 in the NFDIR4Enegy Project."
9      }
10   ]
11 }
```

Listing 3: JSON-LD payload to create a group on the Databus

The main elements are the ”@context” in line 2 which links to the context information of Databus and the ”@graph” in line 3 which will be an element of the Databus graph. The ”@id” in ine 5 is the identifier of the graph element, which only contains the Databus URL, the account and the group name. ”@type” in line 6 identifies the element as a group, ”title” and ”abstract” are the content information as in the form.
Listing 6 shows a small python script, which generates the JSON-LD step by step and registers it to the Databus API.

```
import json
import requests
# ---------------------------------
# Step 1: Define the @context
# ---------------------------------
context = ”https://Databus.openenergyplatform.org/res/context.jsonld”
# ---------------------------------
# Step 2: Create the @graph entry step by step
# ---------------------------------
# Initialize an empty dictionary for the group
group = {}
group[”@id”] = ”https://Databus.openenergyplatform.org/NFDI4Energy/
   TA8”
group[”@type”] = ”Group”
group[”title”] = ”Data of Task Area 8”
group[”abstract”] = (
    ”This data was generated within the context of Task Area 8 in the
        NFDI4Energy Project.”
)
# ---------------------------------
# Step 3: Combine entries into @graph
# ---------------------------------
graph_entries = [group]
# ---------------------------------
```

```python
# Step 4: Create the final JSON-LD payload
# ---------------------------------
payload = {}
payload["@context"] = context
payload["@graph"] = graph_entries
# ---------------------------------
# Step 5: POST the payload to the Databus API
# ---------------------------------
url = "https://Databus.openenergyplatform.org/api/publish"
API_KEY = "d16e2527-506a-4abe-b22a-70f5b8959d47"
headers = {
        "accept": "application/json",
        "Content-Type": "application/ld+json",
        "x-api-key": f"{API_KEY}",
    }
response = requests.post(
    url,
    headers=headers,
    data=json.dumps(payload)
)
# ---------------------------------
# Step 6: Handle response
# ---------------------------------
if response.ok:
    print("Successfully posted JSON-LD to the Databus!")
    print(response.text)
else:
    print(f"Failed to post (status {response.status_code}):")
    print(response.text)
```

Listing 4: Pyhthon script to generate a group

5.2.2 Creating an Artifact

To create an artifact via the API is very similar, first a JSON-LD payload containing the artifact information has to be generated. We take the same artifact as in figure 16, which is shown in listing 5.

```
{
  "@context": "https://Databus.openenergyplatform.org/res/context.
     jsonld",
  "@graph": [
    {
      "@id": "https://Databus.openenergyplatform.org/NFDI4Energy/TA8/
         wind_turbine_onshore_lod_geoss_tp_oeo",
      "@type": "Artifact",
      "title": "Wind turbine technology data from the LOD-GEOSS
         project",
      "abstract": "This is the first dataset to test the technology
         data prototype with OEO annotation"
    }
  ]
}'
```

Listing 5: JSON-LD payload to create an artifact on the Databus

The main difference to the group is, that the ”@id” in line 5 now contains the group name and the artifact name and that ”@type” in line 6 has been changed to ”Artifact”.
Listing 6 shows a small python script, which generates the JSON-LD step by step and registers it to the Databus API.

```
import json
import requests
# ---------------------------------
# Step 1: Define the @context
# ---------------------------------
context = "https://Databus.openenergyplatform.org/res/context.jsonld"
# ---------------------------------
# Step 2: Create the @graph entry step by step
# ---------------------------------
# Initialize an empty dictionary for the group
artifact = {}
artifact["@id"] = "https://Databus.openenergyplatform.org/NFDI4Energy
    /TA8/wind_turbine_onshore_log_geoss_tp_oeo"
artifact["@type"] = "Artifact"
artifact["title"] = "Wind turbine technology data from the LOD-GEOSS
    project."
artifact["abstract"] = (
    "This is the first dataset to test the technology data prototype
        with OEO annotation."
)
# ---------------------------------
# Step 3: Combine entries into @graph
# ---------------------------------
graph_entries = [artifact]
# ---------------------------------
# Step 4: Create the final JSON-LD payload
# ---------------------------------
payload = {}
payload["@context"] = context
payload["@graph"] = graph_entries
# ---------------------------------
# Step 5: POST the payload to the Databus API
# ---------------------------------
url = "https://Databus.openenergyplatform.org/api/publish"
API_KEY = "d16e2527-506a-4abe-b22a-70f5b8959d47"
headers = {
        "accept": "application/json",
        "Content-Type": "application/ld+json",
        "x-api-key": f"{API_KEY}",
    }
response = requests.post(
    url,
    headers=headers,
    data=json.dumps(payload)
)
# ---------------------------------
# Step 6: Handle response
# ---------------------------------
```

```python
if response.ok:
    print("Successfully posted JSON-LD to the Databus!")
    print(response.text)
else:
    print(f"Failed to post (status {response.status_code}):")
    print(response.text)
```

Listing 6: Pyhthon script to generate a group

5.2.3 Creating a Version

To create a version via the API is very similar, first a JSON-LD payload containing the version information has to be generated. We take the same version as in figure 17, which is shown in listing 7.

```
1  {
2    "@context": "https://Databus.openenergyplatform.org/res/context.
       jsonld",
3    "@graph": [
4      {
5        "@id": "https://Databus.openenergyplatform.org/NFDI4Energy/TA8/
           wind_turbine_onshore_lod_geoss_tp_oeo/2021-06-29",
6        "@type": "Version",
7        "title": "Wind turbine technology data from the LOD-GEOSS
           project, intial version",
8        "abstract": "This is the first dataset to test the technology
           data prototype with OEO annotation.",
9        "description": "This is the first dataset to test the
           technology data prototype with OEO annotation.",
10       "license": "https://dalicc.net/licenselibrary/CC-BY-4.0",
11       "attribution": "Dataset by the LOD-GEOSS project, 2021.",
12       "distribution": [
13         {
14           "@type": "Part",
15           "formatExtension": "none",
16           "compression": "none",
17           "downloadURL": "https://openenergyplatform.org/dataedit/
               view/model_draft/wind_turbine_onshore_lod_geoss_tp_oeo"
18         }
19       ]
20     }
21   ]
22 }
```

Listing 7: JSON-LD payload to create a version on the Databus

The main difference to the group and artifact is, that the "@id" in line 5 now contains the group, artifact and version name a name and that "@type" in line 6 has been changed to "Version". The payload also contains additional information on the license in line 10 and attribution in line 11. A distribution in line 12 is created for each file. The "@type" in line 14 of the distribution is "Part". Each distribution contains information on the format extension in line 15, compression in line 16. The download line to the acutal data is added as "downloadURL" in line 17. As distribution is a list, multiple files can be added.

5.2.4 Creating a MOSS Entry

The creation of a MOSS entry just needs the full metadata information as a payload. In the following listing 8 we will assume that the meta data has already been assigned to the variable payload.

```
headers = {
        "accept": "application/json",
        "Content-Type": "application/ld+json",
        "x-api-key": f"{MOSS_API_KEY}",
    }
r = requests.post(
    f"{MOSS_URI_BASE}/api/save?module=module:oemeta&resource={
        databus_URI_BASE}/{ACCOUNT_NAME}/{GROUP_SLUG}/{ARTIFACT}/{
        VERSION}",
    data=json.dumps(DATEN, indent=4),
    headers=headers,
    )
```

Listing 8: Python request to register a data set to MOSS

The MOSS has a seperate API key, which has been generated in section 2.3.2. The URL needs the information on the selected module and the Databus ID which it will extend as a parameter.

6 Creating Search Facets

[TODO: Describe creation of Search Facets in Admin Interface]

References

[1] M. D. Wilkinson et al., "The FAIR Guiding Principles for scientific data management and stewardship," en, *Scientific Data*, vol. 3, no. 1, p. 160 018, Mar. 2016, Publisher: Nature Publishing Group, ISSN: 2052-4463. DOI: 10.1038/sdata.2016.18. Accessed: Jun. 1, 2024. [Online]. Available: https://www.nature.com/articles/sdata201618.

[2] DBpedia Association. "Databus," Accessed: Oct. 24, 2025. [Online]. Available: https://databus.dbpedia.org.

[3] DBpedia Association. "Databus moss," Accessed: Oct. 24, 2025. [Online]. Available: https://github.com/dbpedia/databus-moss.

[4] Open Enery Familiy. "Open energy platform," Accessed: Oct. 24, 2025. [Online]. Available: https://openenergyplatform.org.

[5] M. Freudenberg et al., "The metadata ecosystem of dataid," in *Metadata and Semantics Research*, E. Garoufallou, I. Subirats Coll, A. Stellato, and J. Greenberg, Eds., Cham: Springer International Publishing, 2016, pp. 317–332, ISBN: 978-3-319-49157-8.

[6] C. Hoyer-Klick et al., "Demonstration and best practices," LOD-GEOSS Project, funded by German Ministry for Economic Affairs and Climate Action, Grand 03EI1005A-G, Tech. Rep., 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8119096.

[7] *Documentation*. [Online]. Available: https://dbpedia.gitbook.io/databus.

[8] M. Booshehri et al., "Introducing the open energy ontology: Enhancing data interpretation and interfacing in energy systems analysis," *Energy and AI*, vol. 5, p. 100 074, 2021, ISSN: 2666-5468. DOI: https://doi.org/10.1016/j.egyai.2021.100074. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666546821000288.

[9] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Search," Santa Fe Institute, Working Papers 95-02-010, Feb. 1995. [Online]. Available: https://ideas.repec.org/p/wop/safiwp/95-02-010.html.