

LOD-GEOSS Deliverable D3

# Distributed Data Infrastructure

Johannes Frey, Carsten Hoyer-Klick,  
Christoph Muschner, Ludwig Hülk,  
Denis Streitmatter, Sebastian Hellmann



Supported by:



on the basis of a decision  
by the German Bundestag

03EI1005A-G



## Document Properties

Titel	Distributed Data Infrastructure
Subject	Final report of work package 3
Institute	Institute for Applied Informatics, Leipzig University, DLR Institute of Networked Energy Systems, Reiner Lemoine Institute
Compiled by	Johannes Frey, Carsten Hoyer-Klick, Christoph Muschner, Ludwig Hülk, Denis Streitmatter, Sebastian Hellmann
Date	July 27, 2023
Version	1.0
DOI	10.5281/zenodo.8119055
License	CC-BY 4.0



# Contents

<b>1</b>	<b>Introduction and Goals</b>	<b>1</b>
1.1	Challenges . . . . .	1
1.2	Learning from Earth Observation . . . . .	1
1.3	The DBpedia Databus . . . . .	2
<b>2</b>	<b>Overview of the DBpedia Databus</b>	<b>3</b>
2.1	DBpedia Databus: A Platform for Agile Data Integration . . . . .	3
2.1.1	Promoting Open Data Collaboration . . . . .	3
2.1.2	Efficiency in Data Engineering . . . . .	3
2.1.3	Integration of Databus . . . . .	4
2.1.4	Deployment Levels . . . . .	4
2.2	The Databus metadata model . . . . .	5
2.2.1	Databus approach . . . . .	5
2.2.2	Databus Metadata Example . . . . .	6
2.2.3	Querying . . . . .	9
2.3	Data improvements, extensions, and contributions . . . . .	10
<b>3</b>	<b>MOSS: Metadata Overlay Search System</b>	<b>13</b>
3.1	RDF annotation of datasets . . . . .	13
3.2	Semantic search of Databus Identifiers . . . . .	14
<b>4</b>	<b>Data Provenance</b>	<b>19</b>
4.1	The PROV metadata standard . . . . .	19
4.2	ProvStore . . . . .	20
<b>5</b>	<b>Use Cases for the Energy Databus – Usage of the Databus in energy system analysis community to form a decentralized data(base) architecture</b>	<b>21</b>
5.1	An example data pipeline on the Energy Databus . . . . .	21
5.2	Use Case 1: Sharing and publishing of static data . . . . .	22
5.3	Use Case 2: Recording data provenance and attach descriptive metadata . . . . .	23
5.3.1	Structure of provenance data . . . . .	23
5.3.2	Storage of provenance data in ProvStore . . . . .	24
5.3.3	Provenance graph example . . . . .	24
5.4	Use Case 3: Finding and citing sources for static data . . . . .	25
5.5	Use Case 4: Integration of data updates . . . . .	27
5.6	Use Case 5: Transformation and modification of data . . . . .	27
5.7	Use Case 6: Handling of scenarios . . . . .	28



---

<b>6</b>	<b>Extension of the Open Energy Metadata String</b>	<b>29</b>
6.1	Transforming JSON to Linked Data (JSON-LD) . . . . .	29
6.2	The JSON-LD Context for the OEMetadata . . . . .	29
	<b>Bibliography</b>	<b>33</b>

# 1 Introduction and Goals

## 1.1 Challenges

Energy systems analysis is driven by models and data. The goal is to analyze existing and future energy systems based on numerical models. Modelling in energy systems analysis needs data from a variety of different sources from different domains:

- Meteorological time series for renewable power generation
- Electricity demand for different sectors
- Technology data
- Socio economic data
- Geographical land use data

These data usually come from different sources and one of the first steps in energy systems analysis is data research and processing. The input data is compiled and evaluated. In practice, each source has different access mechanisms and additional information. Each dataset uses its own definitions. This type of data research is very often repeated as the results of this background work usually not shared openly.

One of the main ideas of the LOD-GEOSS project is to ease data discovery and sharing and to reduce the amount of this rather unproductive research and to increase reproducibility and collaboration.

The project therefore contributes to the implementation of the Fair Data Principles (Wilkinson et al. 2016) to make data findable, accessible, interoperable and reusable.

## 1.2 Learning from Earth Observation

Similar problems arose in earth observation about two decades ago. The answer was the development of a distributed data infrastructure for earth observation data, the Global Earth Observation Systems of Systems [GEOSS]<sup>1</sup>. Data could be found through data catalogs, where providers registered their metadata and ways of data access could be discovered through standardized interface descriptions as WSDL (Web Service Description Language) [WSDL]<sup>2</sup>.

<sup>1</sup><https://www.earthobservations.org/geoss.php>

<sup>2</sup><https://www.w3.org/TR/wsdl20/>

As the data stays within the hosting institutions, these institutions stay responsible hosting and maintenance, which eases legal questions and updates of the data. The idea of this current work is the development of a networked database concept based on the ideas of GEOSS and Linked Open Data (LOD) and the semantic web for input and output data of models in energy systems analysis.

### 1.3 The DBpedia Databus

We are developing a distributed database architecture and use it in energy systems analysis. The infrastructure is based on the DBpedia Databus<sup>34</sup>, which was developed in the DBpedia, LOD-GEOSS and PLASS projects. All data providers (this includes users which performed a series of transformation of existing data, but also publishers in a conventional sense) can publish standardized metadata on the Databus (which is based on a metadata catalog in its core). While data providers maintain sovereignty of hosting the data itself, the Databus can be utilized to announce, discover, retrieve, and contribute (to) various data sources in a standardized and interoperable way. The vision of the LOD-GEOSS project is to create an ecosystem between data life cycles of the energy domain itself, the Energy Databus, as part of the larger Databus Network (cf. 1.1).

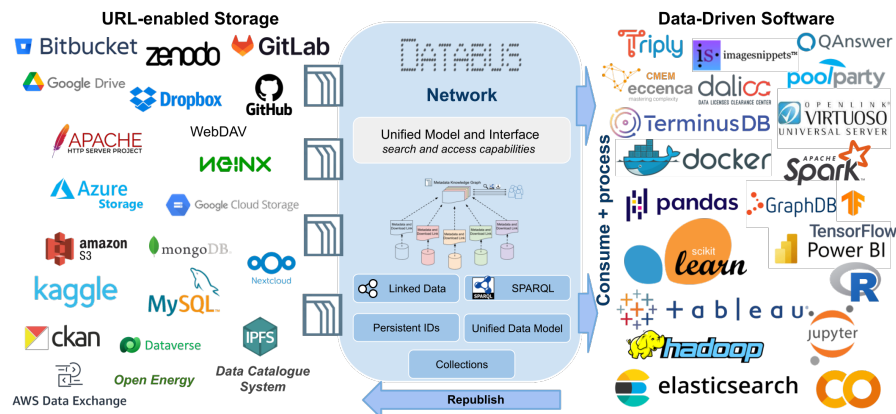


Figure 1.1: An overview of the Databus Network, created by interoperable search and access interface over individually stored data.

<sup>3</sup>Download and code: <https://github.com/dbpedia/databus>

<sup>4</sup>The DBpedia Databus online reference project deployed for the DBpedia community: <https://databus.dbpedia.org/>



## 2 Overview of the DBpedia Databus

### 2.1 DBpedia Databus: A Platform for Agile Data Integration

DBpedia Databus ([Frey et al. 2021](#)) acts as a platform that facilitates agile data integration, collaboration, and automation through a structured meta-data Knowledge Graph. By incorporating concepts from Software Engineering like agile rapid prototyping, build automation, and test-driven development, the Databus enhances Data Engineering capabilities. It enables the interconnection of data through a loosely-coupled bus system and a standardized, extensible metadata format, exploiting state-of-the-art semantic technologies such as ontologies, SPARQL ([Prud'hommeaux & Seaborne 2008](#)), SHACL ([Knublauch & Kontokostas 2017](#)), and Linked Data<sup>1</sup>.

#### 2.1.1 Promoting Open Data Collaboration

The DBpedia Databus empowers communities, public organizations, researchers, and data enthusiasts to deploy their version of the DBpedia Databus, enhancing collaboration on Open Data. This community-driven model enables accelerated innovation and transparency, promoting democratized knowledge and governance. Unlike traditional publisher-centric data platforms, the Databus network emphasizes data consumer needs, improving data discovery, findability, and accessibility.

#### 2.1.2 Efficiency in Data Engineering

The DBpedia Databus addresses the critical issue of pre- and post-processing in data-intensive projects. DBpedia's tech stack includes a comprehensive extraction software, an online database, and several web services. The Databus has fully automated the pipeline and application deployment, saving substantial amounts of time while significantly improving productivity in the DBpedia project by reducing the release cycle duration and improving data validation over billions of facts. The Databus provides a powerful solution to common issues in Data Engineering, namely efficiency, automation, scalability, and data quality. Databus provides an efficient environment for initial identification and acquisition from existing Databuses to low-level tasks such as conversion, normalization to data-quality control and debugging pipelines to loading the data into the final application.

---

<sup>1</sup><https://www.w3.org/DesignIssues/LinkedData.html>

### 2.1.3 Integration of Databus

The DBpedia Databus is designed as an agile solution that can be integrated into existing environments in multiple ways.

- **Comprehensive Integration:** Within the DBpedia Project, Databus has been fully utilized to manage 5000 release files monthly, triple this amount in input and intermediary files, as well as additional dataset contributions from the community.
- **Semantic Layer Addition:** During the LOD-GEOSS project the DBpedia Databus was integrated into the Open Energy Family<sup>2</sup>. Here, the Databus has been deployed as an additional semantic layer over various existing databases to augment the underlying inflexible metadata schemas.
- **Plugin Functionality:** In eccenca's product, the Corporate Memory or CMEM<sup>3</sup>, Databus operates as a plugin. It manages the export, versioning, and archival of subgraphs, which can be shared with other CMEM instances through the semantic catalogue metadata in the Databus Knowledge Graph.

### 2.1.4 Deployment Levels

During many discussion within the LOD-GEOSS, we have identified several deployment levels:

- **Open community:** Develop a data space in the Databus Network and manage it with community contributions spanning across multiple organizations.
- **Organization:** Implement and enterprise's data strategy and optimise efficiency, integration of external data and re-use; manage research data university-wide for scientific sustainability and FAIR. Databus hooks into single sign-on authentication like Siemens ID, Helmholtz AAI or Active Directories of organisations.
- **Department, group or team:** Systematize data workflows internally, record scientific results transparently.
- **Collaborative projects:** Efficiently coordinate data with partners in large projects or multi-project environments.
- **Application, Product or Pipeline:** Streamline and automate data flow within a target application, product or pipeline. The Databus is particularly effective for agile and data-driven decision making and is adept at managing input/output for data-intensive applications such as Search, AI, Deep Learning, Natural Language Processing (NLP), Knowledge Graph

<sup>2</sup><https://openenergy-platform.org/>

<sup>3</sup><https://eccenca.com/products/enterprise-knowledge-graph-platform-corporate-memory>

Construction and Evolution, Databases, Continuous Integration and Microservice Orchestration.

## 2.2 The Databus metadata model

The Databus<sup>4</sup> is a platform built with a domain-independent, minimal core metadata model (the Databus Ontology<sup>5</sup>), that can be extended by user communities via additional custom metadata (ACM) graphs. The core model is designed to publish metadata in a structured, sustainable and automatized fashion and allows federated querying over all Databuses, i.e. the Databus Network. Basic guiding principle for the development of the Databus is to achieve a high degree of interoperability and automatization between:

- Access to data
- Discovery and analysis of data
- Recombination, transformation, and reuse of data
- Contribution of data modification like add-ins and addon (e.g. summary, error report, translation)
- Flexible import of data and deployment of software and services along with data such that users and communities can use the platform to coordinate and manage their data life cycles to establish a platform economy and benefit from network effects of it.

### 2.2.1 Databus approach

The DBpedia Databus is a platform for agile data integration, collaboration, and automation via a structured metadata Knowledge Graph. Data scientists that invest daily efforts in cleaning and integrating data and improve data quality (fitness for use) are able to contribute improvements back to the data source and others. It allows exchanging, curating, and accessing data between multiple stakeholders. Any data entering the bus will be versioned, cleaned, mapped, linked and its licenses and provenance tracked. Hosting in multiple formats is provided to access the data either as dump download or as API. In computer architecture, a *bus*<sup>6</sup> is a communication system that transfers data between components inside a computer, or between computers. The DBpedia Databus elevates this principle to the Web in the following manner:

- Databus is **intended for data analysts**, who can mirror the original data and their extracted or cleaned derivatives to facilitate easier reuse, deployment, and further derivatives. Original data publishers can have a benefit by re-integrating derived value like validations and additions into their provided work. The Databus becomes a community and network.

<sup>4</sup>[https://downloads.dbpedia.org/repo/lts/publication/strategy/2019.09.09/strategy-databus\\_initiative.pdf](https://downloads.dbpedia.org/repo/lts/publication/strategy/2019.09.09/strategy-databus_initiative.pdf)

<sup>5</sup><https://dataid.dbpedia.org/databus>

<sup>6</sup>[http://dbpedia.org/resource/Bus\\_\(computing\)](http://dbpedia.org/resource/Bus_(computing))

- **Data providers** can also create a space on the Databus and add links to accessible files on their servers. It is made by using the publish-wizard or by posting a metadata JSON-LD file with a parameter (*dcat:downloadURL*) to the Databus. The JSON-LD is based on the Databus Ontology, Dublin Core and the Data Catalog Vocabulary [DCAT]<sup>7</sup>. Data providers retain full authority over storage on their servers. This is useful for open data with open licenses and direct access and at the same time the coordination of internal projects with sensitive data.
- **Data user** can access the metadata search using a SPARQL query<sup>8</sup> for discovery and network coordination.

## 2.2.2 Databus Metadata Example

The example shown below shows metadata in JSON-LD and Turtle for a wind power plant list of the German market core data register (MaStR), that has been converted to .nt (RDF) from the original .csv (Table) .

```
1 {
2   "@context" : "https://downloads.dbpedia.org/databus/context.
      jsonld",
3   "@id" : "https://databus.dbpedia.org/jj-author/mastr/bnetza-
      mastr/2021.05.04",
4   "@type" : "Version",
5   "description" : "Data extracted from MaStr by RLI, see [Github]
      (https://github.com/OpenEnergyPlatform/open-MaStR), this is
      a re-publication of open data for demo purposes, please
      attribute the original source",
6   "license" : "https://www.govdata.de/dl-de/by-2-0",
7   "publisher" : "https://jj-author.github.io/webid.ttl#this",
8   "title" : "MarktStammDatenRegister (MaStR) Processing",
9   "account" : "https://databus.dbpedia.org/jj-author",
10  "group" : "https://databus.dbpedia.org/jj-author/mastr",
11  "artifact" : "https://databus.dbpedia.org/jj-author/mastr/
      bnetza-mastr",
12  "hasVersion" : "2021.05.04",
13  "issued" : "2021-05-04T14:43:24Z",
14  "distribution" : [
15    {
16      "@type" : "Part",
17      "dataid-cv:type" : "wind",
18      "dataid-cv:origin" : "rli",
19      "file" : "https://databus.dbpedia.org/jj-author/mastr/
          bnetza-mastr/01.04.01/bnetza-mastr_rli_type=wind.nt.gz",
20      "formatExtension" : "nt",
```

<sup>7</sup><https://www.w3.org/TR/vocab-dcat-2/>

<sup>8</sup><https://www.w3.org/TR/rdf-sparql-query/>

```

21     "compression": "gz",
22     "modified": "2021-05-04T14:36:54Z",
23     "downloadURL": "http://dbpedia-mappings.tib.eu/databus-
        repo/jj-author/mastr/bnetza-mastr/01.04.01/bnetza-
        mastr_rli_type=wind.nt.gz",
24     "byteSize": 3189271,
25     "sha256sum": "05af23c7102c89b7fbf9806be331eb47203de41ba7fd
        3d4b90caf052e3285596",
26     "signature": "ew4Ccw2M04CnBXKSt6nOKJLsF6a0B0zU3SB5GMwrhHFc
        919z3C ..."
27
28   }
29 ]
30 }

```

Listing 2.1: Databus Metadata example in JSON

```

1  @prefix databus: <https://dataid.dbpedia.org/databus#> .
2  @prefix dataid: <http://dataid.dbpedia.org/ns/core#> .
3  @prefix dcv: <http://dataid.dbpedia.org/ns/cv#> .
4  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
5  @prefix dct: <http://purl.org/dc/terms/> .
6  @prefix dcat: <http://www.w3.org/ns/dcat#> .
7  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
8  @prefix cert: <http://www.w3.org/ns/auth/cert#> .
9  @prefix dbo: <http://dbpedia.org/ontology/> .
10 @prefix foaf: <http://xmlns.com/foaf/0.1/> .
11 @prefix prov: <http://www.w3.org/ns/prov-o#> .
12 @prefix sec: <https://w3id.org/security#> .
13
14 <https://databus.dbpedia.org/jj-author/mastr/bnetza-mastr/2021.05
    .04>
15   a databus:Version ;
16   dct:description "Data extracted from MaStr by RLI, see [
        Github](https://github.com/OpenEnergyPlatform/open-MaStr),
        this is a re-publication of open data for demo purposes,
        please attribute the original source" ;
17   dct:license <https://www.govdata.de/dl-de/by-2-0> ;
18   dct:publisher <https://jj-author.github.io/webid.ttl#this> ;
19   dct:title "MarktStammDatenRegister (MaStr) Processing" ;
20   databus:account <https://databus.dbpedia.org/jj-author> ;
21   databus:group <https://databus.dbpedia.org/jj-author/mastr> ;
22   databus:artifact <https://databus.dbpedia.org/jj-author/mastr
        /bnetza-mastr> ;
23   dct:hasVersion "2021.05.04" ;
24   dct:issued "2021-05-04T14:43:24Z"^^xsd:dateTime ;
25   dcat:distribution [

```

```
26     a databus:Part ;
27     dcv:type "wind" ;
28     dcv:origin "rli" ;
29     databus:file <https://databus.dbpedia.org/jj-author/mastr/
        bnetza-mastr/01.04.01/bnetza-mastr_rli_type=wind.nt.gz
        > ;
30     databus:formatExtension "nt" ;
31     databus:compression "gz" ;
32     dct:modified "2021-05-04T14:36:54Z"^^xsd:dateTime ;
33     dcat:downloadURL <http://dbpedia-mappings.tib.eu/databus-
        repo/jj-author/mastr/bnetza-mastr/01.04.01/bnetza-
        mastr_rli_type=wind.nt.gz> ;
34     dcat:byteSize "3189271"^^xsd:decimal ;
35     databus:sha256sum "05af23c7102c89b7fbf9806be331eb47203de41
        ba7fd3d4b90caf052e3285596" ;
36     sec:signature "ew4Ccw2M04CnBXKSt6nOKJLsF6a0B0zU3SB5
        GMwrhHFc9l9z3C ..."
37 ] .
```

Listing 2.2: Databus Metadata example in turtle

The Databus Ontology is a detailed framework for describing classes and properties related to the management and distribution of datasets in a version-centric model, specifically within the DBpedia Databus platform. The main classes are:

1. **databus:Group**: Represents a collection of Databus artifacts (metadata) owned by a Databus user.
2. **databus:Artifact**: A logical dataset in the DBpedia Databus platform, analogous to Maven artifacts in software libraries. The artifact maintains a stable reference across different versions and variants of the dataset, facilitating tracking and retrieval.
3. **databus:Version**: Represents a particular version of an artifact. The URL <https://databus.dbpedia.org/jj-author/mastr/bnetza-mastr/2021.05.04> is of the type **databus:Version**.
4. **databus:Part**: Represents a file (or distribution) associated with a version. A version can contain several parts, each contributing to the dataset version.

The main properties are:

1. **dct:description**: Provides a description of the version. It is detailing the source and purpose of the data.
2. **dct:license**: Specifies the license under which the data is released. The data is licensed under the "Datenlizenz Deutschland – Namensnennung – Version 2.0" license.

3. `dct:publisher`: Indicates who published this version. The publisher is referenced by the URL <https://jj-author.github.io/webid.ttl#this>.
4. `dct:title`: Provides the title of the data, which is "MarktStammDaten-Register (MaStR) Processing".
5. `databus:account`: Specifies the account that provided the metadata, which is <https://databus.dbpedia.org/jj-author>.
6. `databus:group`: The group associated with this version is <https://databus.dbpedia.org/jj-author/mastr>.
7. `databus:artifact`: Specifies the artifact, which is <https://databus.dbpedia.org/jj-author/mastr/bnetza-mastr>.
8. `dct:hasVersion`: The version of this dataset is "2021.05.04".
9. `dct:issued`: The date and time at which this version was issued.
10. `databus:part`: Link to the part of the dataset, which then has the following properties:
  - `dcv:type`: Specifies the type of the part, which is "wind".
  - `dcv:origin`: Specifies the origin of the part, which is "rli".
  - `databus:file`: Provides the URL of the file associated with the part. It is a canonical and persistent file url assigned by the Databus. Accessing it redirects to the `dcat:downloadURL`.
  - `databus:formatExtension`: Specifies the file format extension, which is "nt".
  - `databus:compression`: Specifies the compression method used for the file, which is "gz".
  - `dct:modified`: Specifies when this part was last modified.
  - `dcat:downloadURL`: Provides the URL where the part can be downloaded.
  - `dcat:byteSize`: Provides the size of the file in bytes.
  - `databus:sha256sum`: Provides the SHA-256 hash of the file, which can be used for integrity checks.
  - `sec:signature`: A digital signature for ensuring the authenticity of the metadata.

### 2.2.3 Querying

Any user can query the Databus with a semantic query language for databases called SPARQL to identify their own datasets alongside datasets of other consumers and download them via retrieving the `dcat:downloadURL` or the Databus Client into their local machines and applications. The subsequent query fetches the download URLs of all CSV files from the MaStR dataset in version '2021.05.04'

```
1 PREFIX databus: <http://dataid.dbpedia.org/databus#>
2 PREFIX dct: <http://purl.org/dc/terms/>
3 PREFIX dcat: <http://www.w3.org/ns/dcat#>
4 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
5
6 # Query 1: list all available versions
7 SELECT * WHERE {
8   ?dataset databus:artifact <https://databus.dbpedia.org/jj-author/
9     mastr/bnetza-mastr> .
10  ?dataset dct:hasVersion ?version
11 } Order by ?version
12
13 # Result
14 01.04.00
15 01.04.01
16 2021.05.03
17 2021.05.04
18
19 # Query 2: list all download URLs for version 2021.05.04
20 SELECT ?file WHERE {
21   ?dataset databus:artifact <https://databus.dbpedia.org/jj-author/
22     mastr/bnetza-mastr> .
23   ?dataset dct:hasVersion '2021.05.04' ^^xsd:string .
24   ?dataset dcat:distribution ?distributions .
25   ?distributions dcat:downloadURL ?file .
26 }
27
28 # Result
29 # Note that the Dataset is actually comprised of three files or
30 # parts, we omitted two of the in the above example, but show
31 # them here
32 http://dbpedia-mappings.tib.eu/databus-repo/jj-author/mastr/
33   bnetza-mastr/2021.05.04/bnetza-mastr_type=biomass_cleaned.nt.
34   gz
35 http://dbpedia-mappings.tib.eu/databus-repo/jj-author/mastr/
36   bnetza-mastr/2021.05.04/bnetza-mastr_type=hydro_cleaned.nt.gz
37 http://dbpedia-mappings.tib.eu/databus-repo/jj-author/mastr/
38   bnetza-mastr/2021.05.04/bnetza-mastr_type=wind_cleaned.nt.gz
```

Listing 2.3: SPARQL query examples for the databus

## 2.3 Data improvements, extensions, and contributions

If users either spot quality issues (such as parse errors, wrong data, missing coverage) or would like extensions, they are able to add value to the data in the following manner:



- **Derivatives:** Managing data quality is pareto-efficient, 20% of the initial effort make up 80% of the result. Then effort increases steeply, in fact so steep that the original publisher will struggle to maintain the data. Commercial data providers will weigh effort against income and therefore triage what to improve. Open data providers without an income model normally do not have the resources to accommodate all user needs and focus on own needs. The Databus allows users to re-publish patched data or community extensions, which can be picked up by publishers from the bus to complement their data. These “derive” processes are fully automatable and keep the original source in the parameter *prov:wasDerivedFrom*, therefore all processing can be traced back to the original dataset and all modifications which have been done since then. New versions can be detected by querying the Databus, and processors can run and publish derivatives in third-party storage. Different from pipelines, derive operations form a data network where effort and value are discoverable and can flow freely through the network up to the sources in a trickle-up manner.
- **Mods** are routines for statistical and semantic analysis, continuous integration (CI) testing and enrichment, or any other tasks. Mods are an effective way to provide value for the whole Databus. Only one developer and one server (not necessarily provided by the developer) is required to add additional (meta)information to the datasets, producing a consistent layer of annotations to all data. In other projects this was done by user tagging, which quickly becomes confusing (and often messy). Mods can also be completely selfish. For the DBpedia knowledge graph itself, DBpedia will implement a statistic collection mod, allowing us to better understand how the knowledge graph evolves. Others need to deploy and run it for themselves. Consumers can write mods that check new data before they ingest it to avoid application breaks on data updates. There is an implemented demo mod<sup>9</sup> that measures uptime of all available links of all storage. It is a *prov:Activity* (see chapter 2.3 below) and adds “onlinerate” and “weeklyonlinerate”, and links to detailed reports to the SPARQL-accessible database.
- **User feedback:** Alongside the metadata, users can include feedback links to forums, issue-trackers, or directly to the code (e.g. on GitHub) where other users can specify the issue, fix the software, or provide test cases. This feedback is a valuable input for initial derivative, as it can improve data quality and speed up innovation. It is the technical basis for building a user community.

<sup>9</sup><https://github.com/dbpedia/databus-mods/>

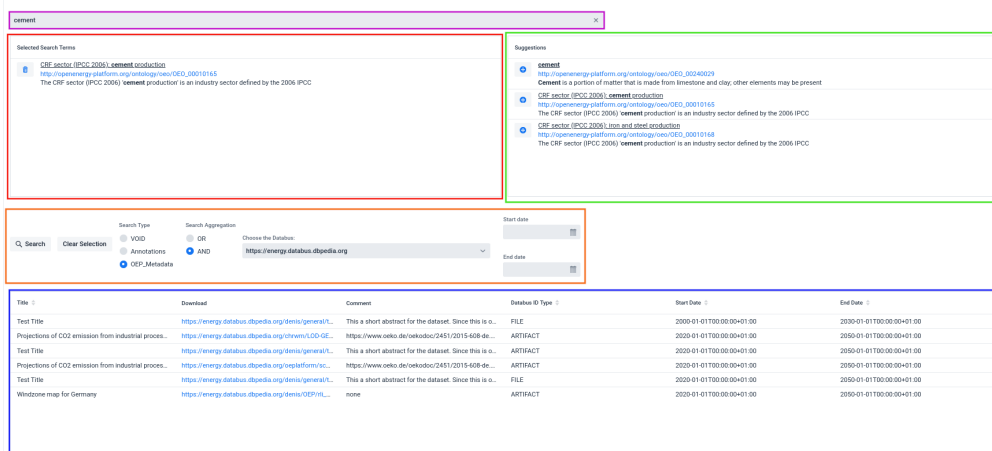


## 3 MOSS: Metadata Overlay Search System

The MOSS (short for Metadata Overlay Search System) is an approach for retrieving Databus Identifiers based on the Databus Mod metadata the identifier is annotated with. Currently there is a vertical prototype implemented<sup>1</sup>, which serves three main features:

1. Annotation of Databus Identifiers with simple RDF Identifiers (URIs)
2. Annotation of Databus Identifiers with complete RDF metadata graphs
3. Retrieval of Databus Identifiers based on RDF Identifiers

Databus Metadata Overlay Search System



Title	Download	Comment	Databus ID	Type	Start Date	End Date
Test Title	<a href="https://energy.databus.dbpedia.org/denis/genera/...">https://energy.databus.dbpedia.org/denis/genera/...</a>	This is a short abstract for the dataset. Since this is o...		FILE	2020-01-01T00:00:00+01:00	2050-01-01T00:00:00+01:00
Projections of CO2 emission from industrial proces...	<a href="https://energy.databus.dbpedia.org/denis/CO2-GE...">https://energy.databus.dbpedia.org/denis/CO2-GE...</a>	<a href="https://www.oeko.de/oekodoc/2451/2015-608-de-...">https://www.oeko.de/oekodoc/2451/2015-608-de-...</a>		ARTIFACT	2020-01-01T00:00:00+01:00	2050-01-01T00:00:00+01:00
Test Title	<a href="https://energy.databus.dbpedia.org/denis/genera/...">https://energy.databus.dbpedia.org/denis/genera/...</a>	This is a short abstract for the dataset. Since this is o...		ARTIFACT	2020-01-01T00:00:00+01:00	2050-01-01T00:00:00+01:00
Projections of CO2 emission from industrial proces...	<a href="https://energy.databus.dbpedia.org/denis/genera/...">https://energy.databus.dbpedia.org/denis/genera/...</a>	<a href="https://www.oeko.de/oekodoc/2451/2015-608-de-...">https://www.oeko.de/oekodoc/2451/2015-608-de-...</a>		ARTIFACT	2020-01-01T00:00:00+01:00	2050-01-01T00:00:00+01:00
Test Title	<a href="https://energy.databus.dbpedia.org/denis/genera/...">https://energy.databus.dbpedia.org/denis/genera/...</a>	This is a short abstract for the dataset. Since this is o...		FILE	2020-01-01T00:00:00+01:00	2050-01-01T00:00:00+01:00
Windzone map for Germany	<a href="https://energy.databus.dbpedia.org/denis/DEP/RL...">https://energy.databus.dbpedia.org/denis/DEP/RL...</a>	none		ARTIFACT	2020-01-01T00:00:00+01:00	2050-01-01T00:00:00+01:00

Figure 3.1: UI window of the metadata search. Marked areas: The search term(s) (purple), the already selected identifiers (red), the suggested identifiers based on the search term (green), the search configurations (orange) and the resulting dataset IDs (blue).

### 3.1 RDF annotation of datasets

MOSS offers two possibilities for annotating any Databus identifier (group, artifact, version, file or collection): simple identifier annotation and full RDF graph annotation. The simple annotation method is accessible through the web user

<sup>1</sup><https://moss.tools.dbpedia.org/>

## Annotate Databus File

Paste Databus file identifier of the file you like to annotate

[https://databus.dbpedia.org/ij-author/mastr/bnetza-mastr/01.04.01/bnetza-mastr\\_rli\\_type=wind.nt.gz](https://databus.dbpedia.org/ij-author/mastr/bnetza-mastr/01.04.01/bnetza-mastr_rli_type=wind.nt.gz)

Annotations

- [http://openenergy-platform.org/ontology/oeo/OEO\\_00000038](http://openenergy-platform.org/ontology/oeo/OEO_00000038)
- <http://dbpedia.org/ontology/WindMotor>

submit

Search term: wind

Suggestions

- [http://openenergy-platform.org/ontology/oeo/OEO\\_00000448](http://openenergy-platform.org/ontology/oeo/OEO_00000448)  
A **wind rotor** (or **wind turbine**) is a turbine that converts the **wind's** kinetic energy into rotational energy.
- [http://openenergy-platform.org/ontology/oeo/OEO\\_00000044](http://openenergy-platform.org/ontology/oeo/OEO_00000044)  
A **wind energy converting unit** is a power generating unit that uses **wind** energy.
- [http://openenergy-platform.org/ontology/oeo/OEO\\_00000043](http://openenergy-platform.org/ontology/oeo/OEO_00000043)  
**Wind** is a process of air naturally moving.
- [http://openenergy-platform.org/ontology/oeo/OEO\\_00000447](http://openenergy-platform.org/ontology/oeo/OEO_00000447)  
A **wind farm** is a power plant that has **wind** energy converting units as parts.
- [http://openenergy-platform.org/ontology/oeo/OEO\\_00000446](http://openenergy-platform.org/ontology/oeo/OEO_00000446)  
**wind energy**

Figure 3.2: UI window of the simple annotation. Marked areas: The annotated dataset identifier (purple), the already existing annotations (red), the search term for new identifiers (blue) and the suggestions for new identifiers (green).

interface (UI) and involves linking URIs via `dc:subject` to any Databus ID for rapid annotation. For this a identifier search functionality is implemented using a DBpedia lookup instance<sup>2</sup>, a highly configurable, Apache Lucene-based search engine for data in RDF graphs. It can work with various graphs, in this case with the Open Energy Ontology, enabling comprehensive and flexible annotation capabilities. By utilizing the UI, users can conveniently track the current status of a Databus identifier annotation. Conversely, the graph annotation method supports the annotation of complex metadata graphs and can handle multiple RDF formats, providing a more extensive and flexible annotation solution. It's model is based on the data model of Databus Mods (see Section 2.3) This module supports both REST API and web UI for graph annotation, accommodating diverse user preferences and workflows. Although any type of RDF graph can be submitted here, the search in the next Section only works for the frictionless-based model of the OEP.

## 3.2 Semantic search of Databus Identifiers

The goal of the semantic search is to turn a term based search (e.g. *cement industry*) into an RDF identifier-based, precise request. Therefor the search consists of two parts: first turning the search term(s) into a set of RDF identifier and then generating a SPARQL query based on these terms, querying the Mods SPARQL endpoint.

The first step is achieved by utilizing the aforementioned DBpedia lookup instance, finding identifiers based on their labels, titles, and descriptions in the (in this case: Open Energy) ontology. The identifiers can be combined across different search terms.

In the second step the identifiers are transformed into a SPARQL query, an example for such a query can be seen in Listing 3.1. This process needs to be

<sup>2</sup><https://github.com/dbpedia/dbpedia-lookup>

configured by the user:

1. **Type of queried metadata:** This defines the type of SPARQL query the search identifiers are used in. Every type of metadata needs a tailored query. Currently there are three different modes implemented:
  - a) VOID<sup>3</sup>: This query type looks for the membership of the identifiers in VOID metadata<sup>4</sup> about datasets deployed to the Databus
  - b) Annotations: This type of query searches for the simple annotations mentioned in Section 3.1
  - c) OEP metadata: This query is based on the frictionless metadata model of the OEP (Hülk et al. 2022) and retrieves datasets where either `dc:subject` or the columns are annotated with the sought-after RDF identifiers.
2. **Databus instance selection:** This selects the Databus instance the metadata of MOSS are joined with (e.g. the Open Energy Databus<sup>5</sup>)
3. **Logical connection of identifiers:** The logical combinations of search terms, either OR or AND for all identifiers.

```

1 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2 PREFIX void: <http://rdfs.org/ns/void#>
3 PREFIX dataid: <http://dataid.dbpedia.org/ns/core#>
4 PREFIX dct: <http://purl.org/dc/terms/>
5 PREFIX dcat: <http://www.w3.org/ns/dcat#>
6 PREFIX db: <https://databus.dbpedia.org/>
7 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
8 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
9 PREFIX mods: <http://mods.tools.dbpedia.org/>
10 PREFIX csvw: <http://www.w3.org/ns/csvw#>
11 PREFIX prov: <http://www.w3.org/ns/prov#>
12 PREFIX time: <http://www.w3.org/2006/time#>
13 PREFIX dbo: <http://dbpedia.org/ontology/>
14 PREFIX saref: <https://saref.etsi.org/core/>
15 PREFIX gr: <http://purl.org/goodrelations/v1#>
16 PREFIX oeo: <http://openenergy-platform.org/ontology/oeo/>
17 PREFIX moddemo: <http://mods.tools.dbpedia.org/ns/demo#>
18
19 SELECT DISTINCT ?type ?title
20             ?comment ?id ?versionURI
21             ?startDateTime ?endDateTime {
22   Graph ?g {
23     ?metadata csvw:table ?table .

```

<sup>3</sup><https://www.w3.org/TR/void/>

<sup>4</sup>Metadata Generated by a Databus Mod

<sup>5</sup><https://energy.databus.dbpedia.org/>

```
24 ?table csvw:tableSchema/csvw:column ?column .
25 ?column gr:valueReference|saref:isAbout oeo:OEO_00010165 .
26 ?activity a moddemo:ApiDemoMod;
27     prov:used ?id .
28 OPTIONAL {
29     ?metadata
30         time:hasTemporalDuration/time:hasDateTimeDescription
31             ?timeDesc .
32     ?timeDesc dbo:startDateTime ?startDateTime;
33         dbo:endDateTime ?endDateTime .
34 }
35 }
36 SERVICE <https://energy.databus.dbpedia.org/sparql> {
37     # Fetches the metadata for each different
38     # type of Databus Identifier
39     {
40         ?dataset a ?type .
41         OPTIONAL {
42             ?dataset dataid:version ?versionURI .
43         }
44         ?dataset dcat:distribution ?distribution .
45         ?distribution dataid:file ?id .
46         ?dataset dct:title ?title .
47         ?dataset dct:abstract|rdfs:comment ?comment .
48     } UNION {
49         VALUES ?type {
50             dataid:Group
51             dataid:Version
52             <https://databus.dbpedia.org/system/voc/Collection>
53             dataid:Collection
54         }
55         ?id a ?type .
56         ?id dct:title ?title .
57         ?id dct:abstract ?comment .
58     } UNION {
59         ?id a ?type .
60         {
61             # Selects the latest version
62             SELECT DISTINCT (MAX(?v) as ?latestVersion) WHERE {
63                 ?dataset dataid:artifact ?id.
64                 ?dataset dcat:distribution ?distribution .
65                 ?dataset dct:hasVersion ?v .
66             }
67         }
68         ?dataset dataid:artifact ?id .
69         ?dataset dct:hasVersion ?latestVersion .
70         OPTIONAL {
```

```

71     ?dataset dataid:version ?versionURI .
72   }
73   ?dataset dcat:distribution ?distribution .
74   ?dataset dct:title ?title .
75   ?dataset dct:abstract|rdfs:comment ?comment .
76 }
77 }
78 }
```

Listing 3.1: An example SPARQL query for joining the metadata on the Mods endpoint with the metadata on the Databus endpoint

With these configurations given the actual search can be executed, consisting of a federated SPARQL query on the Mods SPARQL endpoint (containing the metadata about the *content* of the datasets) and the selected Databus (containing the metadata about the *proper retrieval* of the dataset). Additionally the result can be filtered based on time periods the dataset is about (only possible for the OEP metadata search).





## 4 Data Provenance

In typical analysis workflows, datasets and models from multiple sources are constantly transformed, extended, and merged, which eventually yields a growing number of new entities. In order to maintain reliability and increase trustworthiness, it is inevitable that single processing steps, the origin of data and involved actors can be traced back. This kind of information is referred to as *provenance*. In order to collect provenance information in a consistent way, standards and technologies have been developed. They facilitate comprehensive post-hoc analysis of workflows and backtracking of data products. The following section introduces the established provenance standard and its application to present use cases.

### 4.1 The PROV metadata standard

PROV<sup>1</sup> is a W3C standard defining a set of concepts and relationships to describe provenance of data, which is also adopted in the LOD-GEOSS data infrastructure. The general structure of the PROV ontology (PROV-O) is depicted in figure 4.1. A PROV document consists of a set of relation triples between PROV-O elements with additional meta information.

Elements can be either entities, activities or agents, connected by the relationships "wasAttributedTo" (entity to agent), "wasDerivedFrom" (entity to entity), "wasAssociatedWith" (activity to agent), "wasGeneratedBy" (entity to activity), or "used" (activity to entity).

<sup>1</sup><https://www.w3.org/TR/prov-overview/>

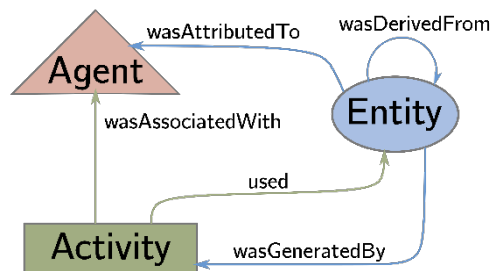


Figure 4.1: PROV Ontology

## 4.2 ProvStore

ProvStore<sup>2</sup> is a publicly available service for storing and managing provenance information.

It is built upon the W3C provenance standard and employs the prov python package. With ProvStore provenance data can be stored in documents and subsequently be transformed, visualized and downloaded easily in multiple ways. Further, it can be published, shared and accessed straightforwardly via RESTful API.

The provenance document can either be made publicly available or shared within specific users with different access roles. Data can be displayed and downloaded in JSON, SVG, Trig, Turtle or XML format. Provenance descriptions are displayed in PROV-N by default and come with some basic statistics regarding quantity of nodes, edges, entities, etc.

There are different visualization layouts available, as Hive Plots, Wheel Plots or Gantt Charts, or more simplified views, as “Data Flow”, “Process Flow” and “Responsibility”, which focus on the comprehensive presentation of flow of information, conducted process and assigned responsibilities, respectively.

---

<sup>2</sup><https://openprovenance.org/store/>

## 5 Use Cases for the Energy Databus – Usage of the Databus in energy system analysis community to form a decentralized data(base) architecture

### 5.1 An example data pipeline on the Energy Databus

In figure 5.1 an example data pipeline is shown which will be used as running example to sketch how the Energy Databus can be used to support the use cases and tackle some of the challenges. The example workflow consists of 3 services and phases starting on the left side with a data preparation service. The cleaned/transformed data is then used in an energy system model. The simulation produces a new dataset which can be used by another service. The general concept is that the Databus is leveraged to collect and handle all meta-data (catalog) for the individual services. In every service the following steps are performed:

- A service can query information about available data from the Databus. It contains:
  - Metadata about the dataset, including basic provenance (e.g. author, license, release date, etc.)
  - Download/Access Location of the dataset
  - Location of additional provenance (Fine-grained provenance information for different models on ProvStore, see section 4.2)

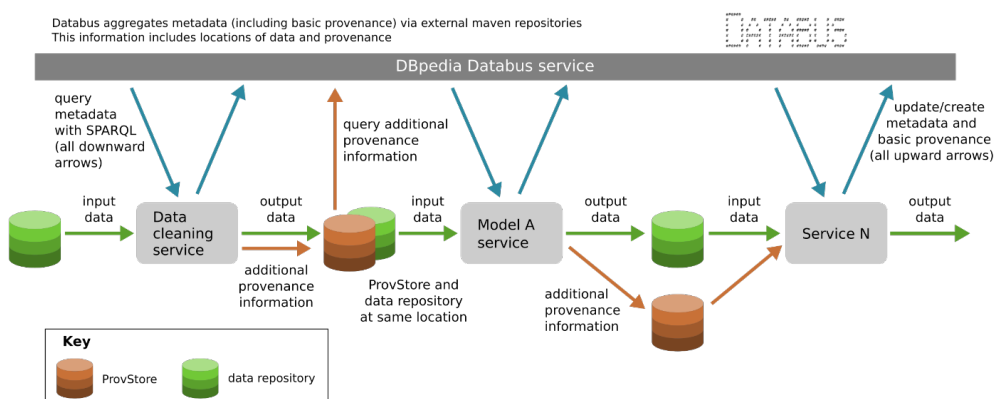


Figure 5.1: Example pipeline of data processing with the data bus

- A service retrieves the metadata via the Databus and accesses the input data
- A service generates a new dataset based on the input (optionally recording detailed provenance) and standardized metadata description (Databus DataID vocabulary)
- The dataset and metadata are hosted in a user-specific/-controlled web-space
- The Databus is notified about the location of the new dataset and its metadata and includes (replicates) the metadata into the centralized catalog in the Databus user namespace/account

## 5.2 Use Case 1: Sharing and publishing of static data

**Use case:** Scientists want to store static data (dump files) in a way that it becomes easily reusable for themselves and fellow scientists.

**Challenges:** There are many uncertainties associated with this supposedly simple task. In which format should the data be released, what metadata is necessary, how to realize stable referenceability, perform update notifications, and how to structure the data to be reusable in a flexible way while being not too complex to understand. These uncertainties in combination with the little amount of time a scientist has to deal with them, leads to temporary solutions. The scientist often lacks concrete action examples to make it better. The consumers are left to deal with the issues and resulting heterogeneity.

**Application:** DBpedia Databus does not host the files itself, these are hosted on the servers (i.e. storage) of its users. Databus consumers upload their files on their own server. Based on their files, they generate (e.g. with the help of an upload client or the user interface) the dataid file in JSON-LD format containing the metadata. The metadata is then registered/announced at the DBpedia Databus and will show up on the Databus user profile (via a Webinterface and the SPARQL backend). The Databus DataID metadata requires a minimal set of metainformation from the publisher in order to support data/service interoperability (FAIR data principles). The Databus system takes care that basic metadata is provided when registering a file including strict validation.

Databus capable tools (like the DBpedia Databus Client<sup>1</sup>) can reduce the effort for both publishers and consumers by allowing flexible on-the-fly conversion between file formats. The Databus assigns for every data asset (file) a stable referenceable identifier. “Update notifications” can be performed in a unified way by registering a new version of the dataset. The Databus offers a tradeoff between fixed and unified but still flexible organization scheme to structure data.

The use case is applied in the demonstrators in deliverable “Demonstration and Best Practices” (Hoyer-Klick et al. 2023) in chapters 3, 4, 5 and 7.

<sup>1</sup><https://github.com/dbpedia/databus-client>

## 5.3 Use Case 2: Recording data provenance and attach descriptive metadata

**Use case:** Scientists want to describe their data with metadata, to make it reusable for themselves and fellow scientists. In this process, they are confronted with different file formats, e.g. Shape Files or Excel files, and internal and external databases and repositories. Their datasets are either of their own making or come from one or more different sources.

Scientists want to version their scenarios or individual datasets to allow the evolution of data without losing identifiability.

**Challenges:** It is often unclear, where to put the metadata. Does it go in the files or database tables themselves or will it go in new files, e.g. README files, a PDF or tables? How to secure, that data and metadata is not separated or confused?

How should the data be modeled, what makes a dataset? Few large datasets lead to few version numbers which are updated quite often. Many small datasets lead to many version numbers which are updated quite seldom. Which is the better practice?

**Application:** In order to keep track of the origin of datasets and transformations of them, provenance techniques are described in chapter 4. In the architecture diagram in figure 5.1 the red arrows indicate the flow of provenance data. Every service that consumes data from and stores data in the Databus, records metadata about the data lineage by additionally connecting to ProvStore (section 4.2) as a service for storing and managing provenance information. While *Derivatives* and *Mods* (section 2.3 already collect high-level provenance about the origin of datasets and activities that generated them, in certain cases it is desirable to trace the workflow on a more fine-grained level. For example, a script applying an agent-based energy system model usually consists of many different calculations and data transformations. For the sake of reproducibility/comparability of data analyses, and to be able to identify reasons for erroneous data, data processing must be recorded step-by-step. This can be done by annotating processing scripts with provenance information that is then automatically stored in ProvStore. An example for annotating Python scripts with additional provenance information can be found here<sup>2</sup>.

Versioning of datasets is a strict requirement of the Databus. Once a version of a dataset was registered it is supposed to be not changed anymore. The Databus offers stable identifiers for the abstract identity but also for individual version. Issued time stamps in the metadata allow a time-oriented view on the evolution of datasets.

### 5.3.1 Structure of provenance data

The PROV W3C standard described in section 4.1 defines a basic skeleton for documents with provenance information. The LOD-GEOSS data infrastructure

<sup>2</sup><https://pypi.org/project/provenance/>

adapts and expands this by mapping specific entities, actors, and activities that are involved in energy system analysis to the PROV-O ontology.

### 5.3.2 Storage of provenance data in ProvStore

The ProvStore infrastructure (section 4.2) will be used for storing and managing provenance data in LOD-GEOSS as it already provides many functionalities for storing data in different PROV formats and analysis. Together with the Databus it is part of the LOD-GEOSS federated database infrastructure. Services that process datasets communicate with ProvStore using its RESTful API. One PROV document in the ProvStore can be associated to one or more datasets on the Databus linked through the *DataID* property of entities. Note that a document can also contain entities that do refer to Databus entries, for example, intermediate results of particular processing activities that are relevant to document an analysis workflow but will not be stored persistently. To this end, libraries for communication with ProvStore will be developed for different types of processing services.

### 5.3.3 Provenance graph example

Energy system analysis in the context of the LOD-GEOSS infrastructure can be broken down to 4 main activities for which provenance data is recorded:

1. ReadDB: Reading of data from a resource listed on the the Databus.
2. WriteDB: Publishing data on the Databus.
3. DataTransformation: An offline data transformation that leads to a modified dataset.
4. ModelSolving: Application of a solver to an energy system model.

Figure 5.2 presents a generic provenance graph as it could be generated automatically while recording information about the aforementioned activities. Ideally, the graph can be read from bottom to top. In the present example, Open Energy Modelling Framework (oemof) scripts are applied on data from the Databus and finally handed back to it. Two scientists, scientist A and scientist B, from the same organization perform different actions on the data and the model. First, oedb (open energy data base) data is retrieved from a Databus reference, which is recorded as a **readDB** activity associated to scientist B and the Databus as involved agents. Any kind of data modification such as cleaning is captured as a **data transformation** activity, which creates an intermediate dataset. This dataset is in turn used for configuring (**model configuration**) and solving (**model solving**) an open energy system model. Finally, the results are stored and a reference is published on the Databus in a **writeDB** activity. The resulting provenance graph depicted below allows to keep track of all these activities, involved users, and produced data entities. Interim results are displayed as individual entities, which enhances versioning control. In contrast to utilized data and models, autonomously performing

scripts, as e.g. the CBC (Coin-or branch and cut) solver, are not classified as entity but as (software) agents. Each entity and each agent comes with a set of individual characteristics, like parameters, IDs, timestamps or resources.

Provenance Graphs are highly adaptable and thus constitute a powerful tool to meet different requirements. The focus of analysis can be set differently, e.g. on the involved persons, characteristics of data, or chronological proceeding. The list of captured metainformation is arbitrarily expandable, as well as the granularity of displayed steps. Therefore, a reasonable implementation of storing and visualizing provenance information is a great benefit and can help to meet the challenges in multiple use cases.

The use case is applied in the demonstrators in deliverable "Demonstration and Best Practices" ([Hoyer-Klick et al. 2023](#)) in chapter 7.

## 5.4 Use Case 3: Finding and citing sources for static data

**Use case:** Like any data which is used as input for scientific analyses, scientists want to find as many (high quality) sources of information as they can. They want to build up a well-documented and well referenced collection of values, enabling them to determine the robustness of the data and derive forecasts and generally secure the quality of the chosen input data. Once the data is collected, it is not being changed and rather serves as static basis for further statistical processing.

**Challenges:** The documentation of data sources goes often hand in hand with literature reviews and reference management. Thereby it is often not clear how to best attach literature information to the data. The data and its formats are used by different scientists and is to be used in publications in a homogeneous and most efficient way.

**Application:** A plethora of options exists to discover (new) datasets:

- Databus basic metadata freetext search (overall search, search per user, per artifact) based on a (lucene) index on metadata
- Fine grained search with SPARQL on metadata
- Exploring user Spaces and Collections via Databus Website
- Follow Basic file Databus file provenance chain to identify datasets which are used as input for other datasets or collections
- Use deep provenance data (e.g. in ProvStore) and combine via federated querying with Databus metadata – allows to search e.g. for outputs of a specific model
- Use another "overlay" search system based on a specific Mod – a Mod providing annotations for the content of the dataset using the Open Energy Ontology could enable an (energy domain) community tailored search

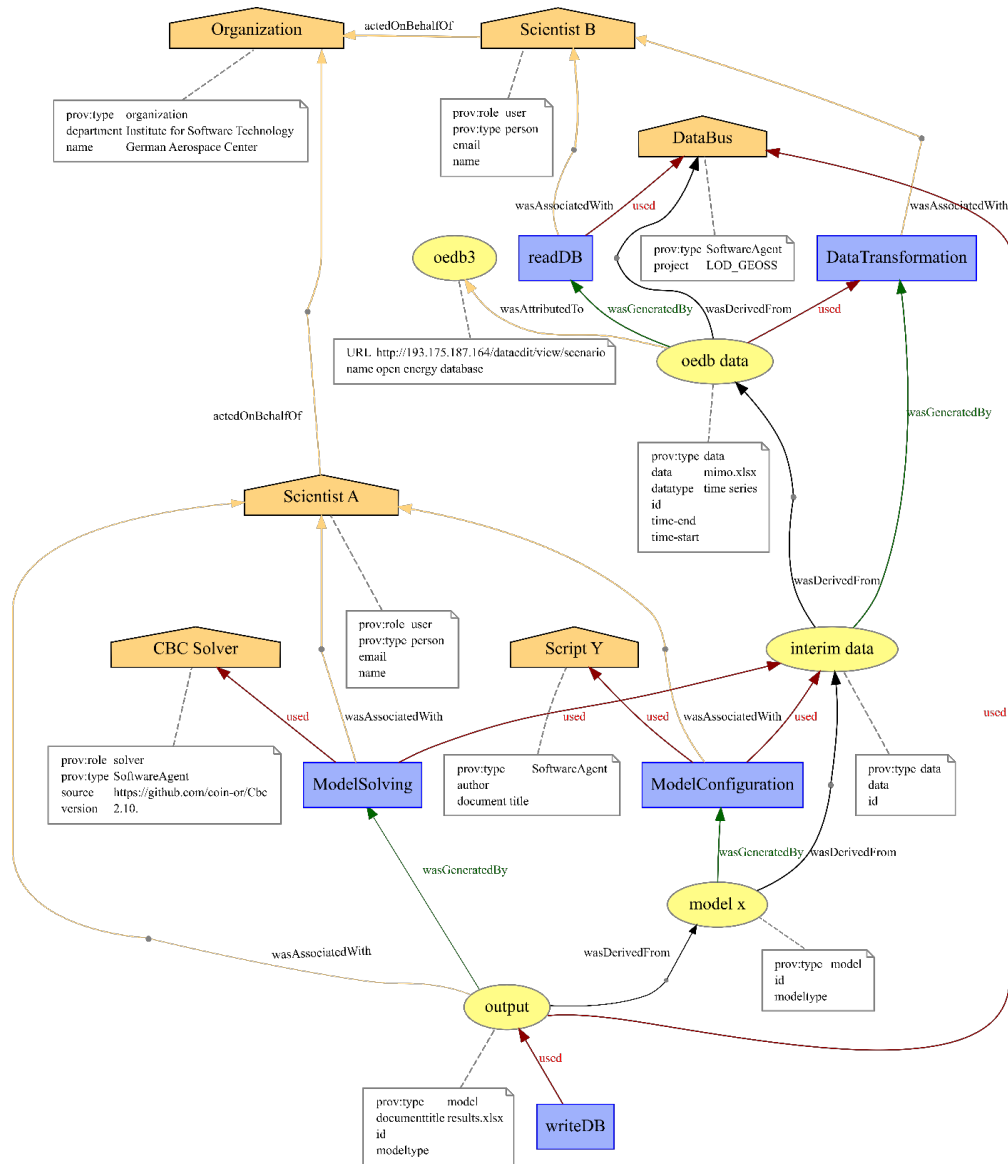


Figure 5.2: A generic provenance graph as it could be generated automatically while recording information about the aforementioned activities.



The versioned Databus identifiers realize a stable way to cite a specific version of a dataset/file independent of its current file location. Moreover, Databus collections allow a self-documenting way to report but also reuse a set/catalog of Databus identifiers (e.g. to reproduce calculations for a paper) by using a single collection ID.

The use case is applied in the demonstrators in deliverable "Demonstration and Best Practices" ([Hoyer-Klick et al. 2023](#)) in chapter 4.

## 5.5 Use Case 4: Integration of data updates

**Use case:** Scientists want to always use the latest data, e.g. the newest statistics of the expected population growth published by a federal ministry. They therefore regularly scan the respective websites, then download, convert and integrate the data in their data repository.

**Challenges:** First of all, it must be avoided that this work has to be repeated by every scientist. In addition, it would be desirable if this work were no longer performed by scientists, but by the data producer. This could be achieved through an automated connection between the data infrastructure of the data producer and an overarching data catalog.

**Application:** Dynamic Databus collections allow to import the latest data from various users based on the special "Latest version property". The user can create such a collection representing their information need and the collection will automatically update whenever a newer of the imported data exists. This allows that an energy model can be run automatically based on the latest data available on the Databus.

This is a variant of the use case 3 "Using shared data" in the demonstrators in deliverable "Demonstration and Best Practices" ([Hoyer-Klick et al. 2023](#)) by explicitly looking for the latest data by using the version tag `latest`.

## 5.6 Use Case 5: Transformation and modification of data

**Use case:** Scientists want to reuse an existing multi-dimensional dataset, e.g. for comparing, validating or supplementing their own data, but need it with (slightly) different resolutions. They e.g. need 15 minutes instead of one-hour intervals, regions instead of countries or the energy consumption by sector instead of a cumulated value. They want to use especially this dataset, since this is already the most adequate one. They can judge the differences as being minor and ignore them, while stating that they did in their publication. Or they can utilize conversion functionality, thereby manipulating their own or the reference dataset.

**Challenges:** The more dimensions a dynamic dataset contains, the more likely it is, that the respective resolutions do not fit the needs. At the same time the

complexity and size of a dataset grows with the number of its dimensions, reinforcing the scientists' desire to reuse this dataset instead of having to generate it themselves. Usually, when a dataset contains aggregated data, the underlying more detailed data is not referenced and/or available.

**Application:** Users can derive converted datasets and publish them for reproducibility and reuse on the Databus. For very common transformations mappings can be created to perform on-the-fly conversion during consuming the data from the Databus on the consumer side. Additionally, a third party could write Databus mods, realizing transformation or shaping of data in a general fashion (e.g. convert all units of measurement into a specific resolution). Moreover, services and application descriptions (metadata) can be linked to specific artifacts, therefore e.g. announcing (conversion) APIs for a dataset.

## 5.7 Use Case 6: Handling of scenarios

**Use case:** Scientists want to define a certain scenario, they are investigating and give it a unique name. In their perspective, a scenario is purely a data collection containing all the data they used for the calculations. They are thereby able to quickly and explicitly reference their input, thereby raising the transparency regarding used assumptions and level of details, etc. For the scientist the usage of identifiable scenarios makes the data more tangible, versionable and exchangeable.

**Challenges:** Usually, a scenario is a list of references pointing to the respective datasets. But it is not always clear, how to persistently handle those references. And sometimes copies of the original datasets are preferred over a mere reference because of potentially ephemeral external sources

**Application:** The Databus uses and assigns PIDs (persistent identifiers) for persistent handling of references. The PIDs will persist even if the original source data may vanish. The Databus can handle multiple and distributed copies of the same dataset, giving users the possibility to upload their contents to multiple repositories increasing the robustness of accessing their data (more than one URL for one URI / PID). The Databus allows the handling of data collections (in this context a data collection shall be equal to a scenario). In this case, a data collection is a list of PIDs referencing other datasets. A data collection can have its own PID. This is a variant of the use case 4 "Publication and Discovery of Secantio data" in the demonstrators in deliverable "Demonstration and Best Practices" ([Hoyer-Klick et al. 2023](#)) by using collections to publish scenario sets. .

## 6 Extension of the Open Energy Metadata String

The OEMetadata (Hülk et al. 2022) schema has been developed in previous projects of the Open Energy Family. It is an extensive set of metadata based on the tabular data package specifications and the FAIR principles (Wilkinson et al. 2016). It builds up upon the DataCite<sup>1</sup> standards and Frictionless data<sup>2</sup>. It contains multiple fields, which are described in the repository<sup>3</sup>, in a nested JSON structure.

### 6.1 Transforming JSON to Linked Data (JSON-LD)

JSON-LD<sup>4</sup> stands for JSON Linked Data and is a lightweight data interchange format that extends the standard JSON (JavaScript Object Notation) format to enable the integration of structured data on the web. It serves as a bridge between the human-readable nature of JSON and the machine-readability of linked data, allowing for the seamless representation and sharing of data across different domains.

JSON-LD introduces a set of additional properties and rules that enrich JSON with semantic meaning, making it more suitable for interoperability and integration with existing web technologies. The core concept of JSON-LD revolves around the use of a context, which provides a common vocabulary for defining the meaning and interpretation of data elements within a JSON document.

The context specifies the semantic meaning of the JSON properties by associating them with appropriate terms or URLs that define their intended interpretation. This linking process allows the JSON-LD document to leverage existing ontologies or create new ones to describe the data more precisely.

### 6.2 The JSON-LD Context for the OEMetadata

The context for the OEMetadata<sup>5</sup> leverages multiple known ontologies to model the Linked Data structure of the frictionless-based metadata string. The most important ontologies included are:

---

<sup>1</sup><https://datacite.org/>

<sup>2</sup><https://frictionlessdata.io/>

<sup>3</sup>[https://github.com/OpenEnergyPlatform/oemetadata/blob/develop/metadata/latest/metadata\\_key\\_description.md](https://github.com/OpenEnergyPlatform/oemetadata/blob/develop/metadata/latest/metadata_key_description.md)

<sup>4</sup><https://www.w3.org/TR/json-ld11/>

<sup>5</sup><https://raw.githubusercontent.com/OpenEnergyPlatform/oemetadata/master/metadata/latest/context.json>

1. **Dublin Core Metadata** ([Kunze & Baker 2007](#)): This vocabulary is used for its common metadata properties like `dct:title`, `dct:description` and `dc:subject`.
2. **CSVW**<sup>6</sup>: A ontology used to describe tabular data. It is used to model the table/column structure.
3. **Time Ontology**<sup>7</sup> and the **DBpedia Ontology**<sup>8</sup>: Properties from these ontologies are used to model the time periods noted in the OEM.

---

<sup>6</sup><https://www.w3.org/ns/csvw>

<sup>7</sup><https://www.w3.org/TR/owl-time/>

<sup>8</sup><https://www.dbpedia.org/resources/ontology/>

## Acknowledgements

The authors would like to thank the Federal Ministry for Economic Affairs and Climate Action of Germany (BMWK) for supporting this work with a grant for the project LOD-GEOSS (03EI1005A-G).

This work was also supported by the Helmholtz Association as part of the program “Energy System Design”.



# Bibliography

- Frey, J., Götz, F., Hofer, M. & Hellmann, S. (2021), Managing and compiling data dependencies for semantic applications using databus client, in ‘15th International Conference on Metadata and Semantics Research’, Springer.  
**URL:** <http://svn.aks.w.org/papers/2021/databus-client/public.pdf>
- Hoyer-Klick, C., Frey, U., Sehn, V., Muschner, C., Launer, J., Hülk, L., Kronshage, S. & Kuckertz, P. (2023), Demonstration and best practices, Technical report, LOD-GEOSS Project, funded by German Ministry for Economic Affairs and Climate Action, Grand 03EI1005A-G.  
**URL:** <https://doi.org/10.5281/zenodo.8119096>
- Hülk, L., Huber, J., Hofmann, C. & Muschner, C. (2022), ‘Open Energy Family - Open Energy Metadata (OEMetadata)’.  
**URL:** <https://github.com/OpenEnergyPlatform/oemetadata>
- Knublauch, H. & Kontokostas, D. (2017), ‘Shapes constraint language (SHACL)’.  
**URL:** <https://www.w3.org/TR/shacl/>
- Kunze, J. & Baker, T. (2007), The dublin core metadata element set, Technical report.
- Prud’hommeaux, E. & Seaborne, A. (2008), ‘SPARQL Query Language for RDF’, W3C Recommendation. <http://www.w3.org/TR/rdf-sparql-query/>.  
**URL:** <http://www.w3.org/TR/rdf-sparql-query/>
- Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., Gray, A. J. G., Groth, P., Goble, C., Grethe, J. S., Heringa, J., ’t Hoen, P. A. C., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S. J., Martone, M. E., Mons, A., Packer, A. L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S.-A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson, M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J. & Mons, B. (2016), ‘The fair guiding principles for scientific data management and stewardship’, *Scientific Data* **3**(1), 160018.