NIH-CFDE FAIR COOKBOOK

Apr 29, 2022

CONTENTS

The Common Fund Data Ecosystem (CFDE) aims to assist the Common Fund (CF) Data Coordination Centers (DCCs) with the process of making the digital objects that they host and produce better adhere to the Findable, Accessible, Interoperable, and Reusable (FAIR) principles.

This FAIR cookbook provides introductory materials about various aspects of FAIRness, including practical guides that show how to enhance digital objects by adhering them to community accepted standards. This includes, for example, documenting and implementing tools with the OpenAPI specification, structuring metadata in DATS and C2M2 formats, providing schema.org specification on data and tool hosting websites, and converting datasets from C2M2 to frictionless for CFDE ingestion.

The CFDE FAIR Cookbook recipes are provided with executable code and example data to guide DCCs through the various ways FAIRness improvements can be achieved. The cookbook is open source and encourages contributions from the community to improve its quality and enrich its depth and coverage.

The cookbook also describes how to perform FAIR assessment of the digital objects hosted by each DCC with FAIRshake. The vision is that FAIR scores will improve after the DCCs will follow the recipes to improve the FAIRness of their resources and this will be reflected in improved FAIR evaluations.

Part I

Introduction

CHAPTER

ONE

THE INTRODUCTION TO FAIR PRINCIPLES

Authors: John Cheadle

Maintainers: John Cheadle

Version: 1.0

1.1 Objectives

The aim of this recipe is to summarize the `FAIR Guiding Principles` as well as define terminology and provide additional resources for individuals who are new to FAIR. The recipe also touches on how FAIR is evaluated in general and in the context of NIH CFDE, and provides both internal and external links to pertinent resources.

1.2 FAIR in 5 Questions

1.2.1 What is FAIR?

The `FAIR principles` are a collection of guidelines by which to improve the **Findability**, **Accessibility**, **Interoperability**, and **Reusability** of data objects. These principles emphasize discovery and reuse of data objects with minimal or no human intervention (i.e. automated and machine-actionable), but are targeted at human entities as well.

FAIR principles are described in detail from the GO-FAIR organization. The permanent URL that resolves to the previous link can be found from the FAIRsharing.org record.

1.2.2 When was FAIR established?

The `FAIR guiding principles` were first formally published in 2016 by Wilkinson, et al. Since then, two additional publications centering around metrics and maturity indicators for FAIRness have been authored. A permanent link to these papers is provided below:

- https://doi.org/10.1038/sdata.2016.18
- https://doi.org/10.1038/sdata.2018.118
- https://doi.org/10.1038/s41597-019-0184-5

1.2.3 Why do we care about FAIR?

The potential benefits to adhering to the FAIR principles are myriad. The rationale outlined in the first published paper includes:

- Facilitation of ongoing discovery, evaluation, and reuse in downstream studies after initial publication of a digital object.
- Deep and broad knowledge integration by human and computational stakeholders alike to advance scientific discovery.
- Recent work hints at citation benefits of making data and metadata more accessing as documented in this recent publication https://doi.org/10.1371/journal.pone.0230416

1.2.4 How is FAIR evaluated?

In 2018, Wilkinson et al. published a paper detailing the evaluation of FAIR using `14 exemplar metrics`, which corresponds to each of the FAIR sub-principles. Metrics are manually scored tests of individual FAIR principles, in the form of a questionnaire.

A 2019 paper from the group was published which introduced an automatable framework composed of `maturity indicators (MIs)`, their corresponding compliance tests, and a web evaluator to run sets of compliance tests and report the results. In contrast to metrics, MIs must be evaluated by machine; the focus here is primarily on automation and objective assessment.

Metrics and MIs are open to feedback from the community at large. The dedicated public GitHub repository is located here.

Another 2019 paper by Clark et al. and led by Dr Ma'ayan introduces an alternative set of tools and interpretation of the FAIR guidelines, developing the notion or `rubric`, which are detailed in the section below.

1.2.5 How is FAIR evaluated in the context of the CFDE?

In the context of the CFDE, aspects of FAIR are evaluated using the FAIRshake tool, which interrogates a digital object using an appropriate rubric of metrics. The metrics are questionnaire-style. Many digital objects have already been evaluated with the NIH-CFDE FAIR Rubric. Within FAIRshake other organizations define rubrics containing metrics that are more appropriate for evaluating their own digital objects.

1.2.6 Why evaluate FAIR?

As we hinted, the FAIR principles are `broad guiding principles`, sometimes providing `very specific directions` but sometimes also `leaving room for interpretation` or `customization to local, community specific needs`. For instance, `'enough metadata'` is highly dependent on the domain of application. Therefore, a `one size fits all approach` is not applicable hence the decoupling between the software agent performing the evaluation (FAIRShake or FAIREvaluator) from the need to build a framework to express the evaluation rules (e.g. `rubrics` or `maturity indicators` as consumed respectively by the cited evaluation engines).

1.2.7 How to contribute, reform, evolve evaluation rules?

While it is possible to assess FAIR on general terms, specific domains of knowledge requires dedicated extensions. To give a practical example, `Findability` and `Interoperability` mandate that `sufficient metadata` is provided. However the amount of metadata varies depending on the application and content, so requirements for the description of *transcriptomics experiments* differs from *medical imaging* ones. It is therefore necessary to craft domain specific profiles which can be used by the FAIR assessor tools to determine the level of compliance. When using FAIRShake, this is achieved by the creation of dedicated `rubrics`.

As the field of scoring FAIR is still relatively new, several groups, e.g. RDA FAIR Maturity Model working group, have been established to evolve the metrics or maturity indicators in an attempt to resolve differences in interpretation of the FAIR principles and also to create a forum where discussions can take place.

- FAIR rubrics: https://fairshake.cloud/rubric/
- FAIR metrics github repository: https://github.com/FAIRMetrics/Metrics

1.3 Conclusions

FAIR principles, first officially published in 2016, are meant to be guideposts for rendering a digital object more discoverable and (re)usable. The CFDE evaluates FAIRness using FAIRshake, which leverage collections of metrics that interrogate one or more of the FAIR principles. Community adherence to FAIR principles provides benefits to human and computational stakeholders alike by facilitating ongoing discovery and reuse of digital objects after initial publication and enabling knowledge integration to further scientific discovery.

CHAPTER

TWO

ENGAGING WITH CFDE

On January 29, 2020, the NIH Common Fund posted OTA-20-005 Engaging Common Fund Data Coordinating Centers to Establish the Common Fund Data Ecosystem. This announcement invited Engagement Plans from Common Fund Data Coordinating Centers (DCCs) to engage with the Common Fund Data Ecosystem Coordinating Center (CFDE-CC) and with each other to establish the CFDE. Since the original announcement several rounds of Engagement Plan submissions and review were completed. Future Engagement Plans from Common Fund are expected to continue in the future if the CFDE program continues.

For more information, including a full overview of the engagement process, please refer to the NIH-CFDE project engagement page.

Part II

Recipes

CHAPTER

THREE

THE FAIR COOKBOOK RECIPES

Each recipe in the FAIR cookbook provides detailed guidance on how FAIR principles are implemented and supported in practice within the context of the CFDE.

The ***Discoverability*** section highlights the Crosscut Metadata Model (C2M2) metadata standards, which are designed to improve the findability of data from various data coordination centers (DCCs). Examples of how data from the KidsFirst and LINCS DCCs can be extracted, transformed, and loaded (ETL) into the C2M2 are provided.

The ***Identification*** section discusses the significance of assigning unique, persistent, and resolvable identifiers to entities within the context of FAIR data, how those identifiers are used within the CFDE infrastructure to enable data interoperability, and how to generate identifiers using Minid.

The ***Semantics*** section introduces controlled vocabularies and ontologies, how they are built, and why they are important for data harmonization. A brief overview is provided of existing ontologies and ontology-related services, as well as of the terminologies and ontologies currently used in the CFDE.

The ***FAIR Compliance*** section addresses how adherence to the FAIR principles can be assessed using FAIRshake, a service that catalogs digital objects and measures their FAIR compliance based on some set of criteria. This section also outlines the CFDE FAIR Rubric, the set of metrics specifically designed to evaluate the FAIRness of datasets from different Common Fund DCCs. A tutorial on building FAIR Application Programming Interfaces (API) is included.

Click the following to jump directly to a section of the cookbook:

CHAPTER

FOUR

FINDABILITY & DISCOVERABILITY

This section covers the F of the FAIR principles and thus describes the NIH CFDE Crosscut Metadata Model, abbreviated to C2M2 from now on.

It includes:

- The C2M2 model specifications
- *The CFDE namespace conventions*
- *an ETL example to an early version of the C2M2 model in the DATS form.*

4.1 The C2M2 Model

Authors: Arthur Brady

Maintainers: Arthur Brady

Version: 0.1

License: CC0 1.0 Universal (CC0 1.0) Public Domain Dedication

---

- Information about the Crosscut Metadata Model (C2M2) can be found here.

4.2 Conceptual Description of the Level 1 C2M2

Authors: Rick Wagner

Maintainers: Rick Wagner

Version: 0.1

License: CC0 1.0 Universal (CC0 1.0) Public Domain Dedication

---

4.2.1 Objectives

This is a conceptual and narrative description of the Level 1 Crosscut Metadata Model (C2M2). It covers the things (proper nouns) in the Level 1 C2M2 and their relationships, and describes the tables used to represent them. The last section covers the internal controlled vocabularies used for a few attributes. These notes do not go heavily into things like the format (syntax) of the columns or the specific primary key and foreign key relationships.

4.2.2 Things (Proper Nouns) Described

The Level 1 C2M2 includes tables to describe the following things (entities), and the relationships among them.

- Namespaces
- Project
- File
- Subject
- Biosample
- Collection

This section has descriptions of each thing and a list of its attributes (fields).

Namespaces

A namespace is a logical groupings of things, used to avoid collisions among the names used by different Data Coordination Centers (DCCs). We assume that each DCC assigns a unique local name to each thing that it manages. (If this assumption is violated–if, for example, biosamples and files may be assigned the same local name–then additional local structure may be needed.) Then, anything from any DCC can be given a unique global name by concatenating the namespace id for the DCC from which the thing originates with the local name assigned to the thing by that DCC. Thus, for example, two things originating from DCC1 and DCC2, and each assigned a local name Sample1, will have distinct C2M2 names: DCC1:Sample1 and DCC2:Sample1.

Attributes

- **namespace** A globally unique ID representing this namespace
- **abbreviation** A short display label for this namespace
- **name** A short, human-readable, machine-read-friendly label for this namespace
- **description** A human-readable description of this namespace

Project

There can be a single project for each DCC, or things like studies can be represented as subprojects. The field persistent id could be a website for project, or a DOI for a paper. When we get to collections to describe datasets or cohorts, we'll show what project they were part of.

- **abbreviation** A short display label for this project
- **name** A short, human-readable, machine-read-friendly label for this project
- **description** A human-readable description of this project

- **persistent id** A persistent, resolvable (not nec. retrievable) URI generated by a DCC and attached to this project

File

- **file id** The unique name for this file, compromised of:
  - **namespace** Namespace for the DCC or file creator
  - **id** An ID representing this file, unique within this namespace
- **project** Which project or subproject created this file
- **persistent id** A persistent, resolvable (not nec. retrievable) URI generated by a DCC (using, e.g., our minid server) and attached to this file
- **creation_time** An ISO 8601 – RFC 3339 (subset)-compliant timestamp documenting this file's creation time

  Ex. YYYY-MM-DDTHH:MM:SS±NN:NN
- **size_in_bytes** The size of a file in bytes
- **sha256** The output of the SHA-256 cryptographic hash function after being run on this file: one or both of sha256 and md5 is required; sha256 is preferred
- **md5** The output of the MD5 message-digest algorithm after being run as a cryptographic hash function on this file: one or both of
- **filename** A filename with no prepended PATH information
- **file_format** An EDAM CV term ID identifying the digital format of this file

  Ex. TSV or FASTQ
- **data_type** An EDAM CV term ID identifying the type of information stored in this file

  Ex. RNA sequence reads

Subject

- **subject id** The unique name for this subject, compromised of:
  - **namespace** Namespace for the DCC or subject provider
  - **id** An ID representing this subject, unique within this namespace
- **project** Which project or subproject created this file
- **persistent id** A persistent, resolvable (not nec. retrievable) URI generated by a DCC and attached to this subject
- **creation_time** An ISO 8601 – RFC 3339 (subset)-compliant timestamp documenting this subject record's creation time

  Ex. YYYY-MM-DDTHH:MM:SS±NN:NN
- **granularity** A CFDE CV term categorizing this subject by multiplicity (see Subject Granularity under Controlled Vocabularies). One of:
  - single organism
  - symbiont system
  - host-pathogen system
  - microbiome

  - cell line
  - synthetic

Biosample

- **biosample id** The unique name for this biosample, compromised of:
  - **namespace** Namespace for the DCC or biosample owner
  - **id** An ID representing this biosample, unique within this namespace
- **project** Which project or subproject created this biosample
- **persistent id** A persistent, resolvable (not nec. retrievable) URI generated by a DCC and attached to this biosample
- **creation_time** An ISO 8601 – RFC 3339 (subset)-compliant timestamp documenting this biosample's creation time

  Ex. YYYY-MM-DDTHH:MM:SS±NN:NN
- **assay_type** An OBI CV term ID describing the type of material represented by this biosample
- **anatomy** An UBERON CV term ID used to locate the origin of this biosample within the physiology of its source or host organism

Collection

Like projects, collections can have subcollections. Collections can hold files, biosamples, or subjects, which is done using a relationship.

- **collection id** The unique name for this collection compromised of:
  - **namespace** Namespace for the DCC or collection creator
  - **id** An ID representing this collection, unique within this namespace
- **persistent id** A persistent, resolvable (not nec. retrievable) URI generated by a DCC and attached to this collection
- **abbreviation** A very short display label for this collection
- **name** A short, human-readable, machine-read-friendly label for this collection
- **description** A human-readable description of this collection

4.2.3 Relationships

There are several relationships between things that can be described, like which subject a biosample comes from. These a often mapping tables between the unique names (namespace, id) of different things.

Things in Collections

Collections can contain one or more files, biosamples, or subjects. A collection may contain a combination of different types. There are tables for each type that map the items into their collections. The item is identified by its namespace and id, so is the collection. Effectively, the tables look like the following:

Attributes

Files in collection

- **subject id** The unique name (namespace, id) of the subject
- **collection id** The unique name (namespace, id) of the collection

Biosamples in collection

- **biosample id** The unique name (namespace, id) of the biosample
- **collection id** The unique name (namespace, id) of the collection

Subjects in collection

- **subject id** The unique name (namespace, id) of the subject
- **collection id** The unique name (namespace, id) of the collection

Biosamples and Subjects

To allow for multiple subjects to be represented in a single biosample and vice versa, there is a mapping table between biosamples and subjects.

Attributes

- **biosample id** The unique name (namespace, id) of the biosample
- **subject id** The unique name (namespace, id) of the subject

Files Describing Subjects and Biosamples

To show a relationship between a file and a subject a or biosample, like a sequence file generated from a biosample, there are two more mapping tables.

Attributes

Files describing biosamples

- **file id** The unique name (namespace, id) of the file
- **biosample id** The unique name (namespace, id) of the biosample

Files describing subjects

- **file id** The unique name (namespace, id) of the file
- **subject id** The unique name (namespace, id) of the subject

Subject Role and Taxonomy

A table linking a subject, a subject_role (a named organism-level constituent component of a subject, like 'host', 'pathogen', 'endosymbiont', 'taxon detected inside a microbiome subject', etc.) and a taxonomic label (which is hereby assigned to this particular subject_role within this particular subject)".

Attributes

- **subject**
  - **namespace** The namespaec of the subject
  - **id** The ID of this subject
- **role** The role assigned to this organism-level constituent component of this subject (see Subject Role under Controlled Vocabularies). One of:
  - single organism
  - host
  - symbiont
  - pathogen
  - microbiome taxon
  - cell line ancestor
  - synthetic
- **taxonomy_id** An NCBI Taxonomy Database ID identifying this taxon

4.2.4 CFDE Controlled Vocabularies

Subject Granularity

Term	Description
single organism	One organism
symbiont system	A mixed system of consisting of two or more organisms (symbionts) in symbiosis (living colocated in time and space): one such symbiont may optionally be identified as a host
host-pathogen system	A special case of a symbiont system consisting of one symbiont, designated as a host, plus one or more other symbionts acting to create or sustain disease within the host organism
microbiome	A symbiont system consisting of a collection of (potentially unknown or partially characterized) taxa, where the environment in which the system resides is well-characterized, but the taxonomic composition of the system may be unknown; optionally contains one symbiont specially identified as a host
cell line	A cell line derived from one or more species or strains
synthetic	A synthetic biological entity

Subject Role

Term	Description
single organism	The organism represented by a subject in the 'single organism' granularity category
host	Any organism identified as a host for a subject assigned to the 'symbiont system', 'host-pathogen system', or 'microbiome' granularity categories
microbiome taxon	A constituent taxon of either (a) a subject assigned to the 'environmental microbiome' granularity category or (b) the microbiome (non-host) portion of a subject assigned to the 'host-associated microbiome' granularity category [NB: This role is probably not appropriate for Level 1, because it necessitates the post-facto attachment of downstream analysis procedures (subject -> sample -> library prep -> sequencing -> bioinformatics -> taxonomic classification results) to a subject which was originally uncharacterized at this level]
symbiont	An organism identified as a symbiont within a subject assigned to the 'symbiont system' granularity category
pathogen	An organism identified as a pathogen symbiont in a subject assigned to the 'host-pathogen system' granularity category
cell line ancestor	A taxon identified as a source organism for a subject assigned to the 'cell line' granularity category
synthetic	A synthetic biological entity

4.2.5 Conclusions

This section provides a concise overview of the key objects and concepts covered by the C2M2 model and should be viewed as an initial contact point for anyone interested in mapping data into the C2M2 model, thereby getting ready for a full ETL process.

4.2.6 What to read next?

- *CFDE namespaces*
- *CFDE selected terminologies?*

4.3 Conceptual Description of the Level 2 C2M2

Authors: Rick Wagner

Maintainers: Rick Wagner

Version: 0.1

License: CC0 1.0 Universal (CC0 1.0) Public Domain Dedication

4.3.1 Objectives

Resources (files, subjects, biosamples, etc.) are uniquely named in the context of the C2M2 using a namespace and an local name (the id column of the entity tables). When combined, the namespace and local name form a unique name for the resource through the CFDE. In principle, this could be done using only a full URI for the resource. In practice, the use of namespaces is a mechanism to allow DCCs to name their resources in a simple manner without collision. The Common Fund Data Ecosystem Coordinating Center (CFDE CC) defines a process for DCCs to define their own namespaces in a manner that avoids collions, without requiring CFDE CC involvement. DCCs are welcome to engage the CFDE CC with questions before investing significant effort in creating their ETL process.

4.3.2 Namespaces

The namespace must be a valid URI (not necessarily resolvable) that is under the control of the DCC or a trusted naming authority, and may contain an intial path information. Examples of valid namespaces

- `https://project-a.example.org/`
- `https://project-a.example.org/samples/`
- `tag:project-a.example.org,2020:/`
- `tag:project-a.example.org,2020:samples#`

The namespace itself does not need (nor should) have any semantic meaning on its own, though the namespaces table in the C2M2 may contain descriptive text. Nor does the namespace define relationships between resources, such as project membership or type. Files, biosamples, subjects, and collections declare what project they are associated with. The relationship between entities, such as a file produced from a biosample, is defined separately from what project or namespace those entities are associated with.

4.3.3 Syntax

The local name of a resource is concatenated with the namespace to form the full resource name. To ensure this, the namespace must be a valid URI and may include a trailing separation character, such as `?` or `#`. The local name of a resource may have any syntax, so long as when concatenate with the namespace the full names is also a valid URI.

Given the local name `8675/REAMDE` and the previous namespace examples, the full resouces names would be:

- `https://project-a.example.org/8675/REAMDE`
- `https://project-a.example.org/samples/8675/REAMDE`
- `tag:project-a.example.org,2020:/8675/REAMDE`
- `tag:project-a.example.org,2020:samples#8675/REAMDE`

4.3.4 Validation

There are tools for validating and introspecting URIs, like the Python rfc3986 package. Using our examples above, we can valid the namespaces, the full names, and look at the breakdown of the elements.

```
import rfc3986

# https://project-a.example.org/
rfc3986.is_valid_uri('https://project-a.example.org/')
True
rfc3986.uri_reference('https://project-a.example.org/')
```

(continues on next page)

(continued from previous page)

```
URIReference(scheme='https', authority='project-a.example.org', path='/', query=None,␣
 ↪fragment=None)

# https://project-a.example.org/samples/
rfc3986.is_valid_uri('https://project-a.example.org/samples/')
True
rfc3986.uri_reference('https://project-a.example.org/samples/')
URIReference(scheme='https', authority='project-a.example.org', path='/samples/',␣
 ↪query=None, fragment=None)

# tag:project-a.example.org,2020:/
rfc3986.is_valid_uri('tag:project-a.example.org,2020:/')
True
rfc3986.uri_reference('tag:project-a.example.org,2020:/')
URIReference(scheme='tag', authority=None, path='project-a.example.org,2020:/',␣
 ↪query=None, fragment=None)

# tag:project-a.example.org,2020:samples#
rfc3986.is_valid_uri('tag:project-a.example.org,2020:samples#')
True
rfc3986.uri_reference('tag:project-a.example.org,2020:samples#')
URIReference(scheme='tag', authority=None, path='project-a.example.org,2020:samples',␣
 ↪query=None, fragment='')

# https://project-a.example.org/8675/REAMDE
rfc3986.is_valid_uri('https://project-a.example.org/8675/REAMDE')
True
rfc3986.uri_reference('https://project-a.example.org/8675/REAMDE')
URIReference(scheme='https', authority='project-a.example.org', path='/8675/REAMDE',␣
 ↪query=None, fragment=None)

# https://project-a.example.org/samples/8675/REAMDE
rfc3986.is_valid_uri('https://project-a.example.org/samples/8675/REAMDE')
True
rfc3986.uri_reference('https://project-a.example.org/samples/8675/REAMDE')
URIReference(scheme='https', authority='project-a.example.org', path='/samples/8675/
 ↪REAMDE', query=None, fragment=None)

# tag:project-a.example.org,2020:/8675/REAMDE
rfc3986.is_valid_uri('tag:project-a.example.org,2020:/8675/REAMDE')
True
rfc3986.uri_reference('tag:project-a.example.org,2020:/8675/REAMDE')
URIReference(scheme='tag', authority=None, path='project-a.example.org,2020:/8675/
 ↪REAMDE', query=None, fragment=None)

# tag:project-a.example.org,2020:samples#8675/REAMDE
rfc3986.is_valid_uri('tag:project-a.example.org,2020:samples#8675/REAMDE')
True
rfc3986.uri_reference('tag:project-a.example.org,2020:samples#8675/REAMDE')
URIReference(scheme='tag', authority=None, path='project-a.example.org,2020:samples',␣
 ↪query=None, fragment='8675/REAMDE')
```

4.3.5 Namespace Selection

There are two recommended ways to choose a namespace, tag URIs or HTTPS and an FQDN. Tag URIs and HTTPS URLs using an FQDN allow DCCs to define their own namespaces, while splitting a persitent identifier permits enternal use of the full resource name.

Tag URIs have a simple syntax with the scheme "`tag:`", the entity defining the URI, and the local name: `tag:<naming authority>:<local name>` The `<naming authority>` may be an FQDN and a date that FQDN was managed by the DCC. E.g., for Example.org's Project A, their namespace may be `tag:project-a.example.org,2020:`

HTTPS URLs are similar to XML namespace, where the DCC must have control of the FQDN, but the URI does not need to be a resolvable URL. Example.org could use this to declare a namespace for their samples from the Project A study. `https://project-a.example.org/samples`

4.3.6 Declaration

Namespaces are declared in the `id_namespace` table

Column Name	Type	Description
id	URI	A globally unique ID representing this namespace. Used as a foreign key by entities.
name	slug	A short, human-readable, machine-read-friendly label for this namespace.
description	string	A human-readable description of this namespace.

Example

```
{
  "id":          "tag:project-a.example.org,2020:samples",
  "name":        "Samples from the Example.org Project A study",
  "description": "The sample identifiers sample were derived from the
                  Example.org Project A accession numbers."
}
```

The descriptive text (name and description) should related to the namespce and the local names of the resources.

4.3.7 Conclusions

This section provides a concise overview of the mechanism to declaring a `namespace` associated with a DCC in the context of the CFDE project. It is an important `harmonization` process, which aims to avoid collisions between identifiers, while enhancing `interoperability` and `findability`. It also represents a key process in mapping data into the C2M2 model, thereby getting ready for a full ETL process from a DCC to the CFDE model and future indexing in the CDFE Deriva system.

4.3.8 What to read next?

- *CFDE selected terminologies*

4.4 Schema.org, BioSchemas, JSONSchema, JSON-LD and DATS

A case study with KidsFirst is considered for asset metadata serialization as DATS, and validation with JSONSchema

Authors: Daniel J. B. Clarke

Maintainers: Daniel J. B. Clarke

Version: 1.0

4.4.1 Motivations

The Data Tag Suite (DATS) metadata model as described in this paper and fully codefied in this repository strives to model datasets irrespective of their domains. DATS embodies several key elements making it extroudinarily useful for FAIRification:

- Machine Readibility: Datasets described with a consistent DATS format permit machines to resolve FAIR metadata such as identifiers, authorship, funding, citation, license, consent, access, provenance, and ultimatly topic as well.
- RDF Interoperability: Serialized in strict JSON-LD, the DATS format is renderable as an RDF graph permitting interoperability with ontological vocabularies and existing dataset description formats including schema.org and the Open Biological and Biomedical Ontology (OBO).
- Findability: The utilization of these consistent formats will permit various existing services and endless future ones to be able to identify aspects of the dataset for the purposes of indexing and searching. One such service is google dataset search which utilizes schema.org metadata.
- CFDE Compatibility: Tooling has been created to convert DATS to the C2M2 and for automatically evaluating the FAIRness of Datasets through the DATS metadata model.

4.4.2 Ingredients

1. Access to a manifest or API for serving your existing datasets.

4.4.3 Objectives

1. Convert your existing metadata into the DATS metadata schema
2. Check the validity of your DATS schema

4.4.4 Preparation

We need to get the manifest or access to an API serving the existing metadata. In our case study, KidsFirst, the assets are browsable in the file repository. After enabling all "Columns" click the "Export TSV" button and save that file to `./data/file-table.tsv`.

```
# Python tool for data table processing
import pandas as pd
# Jupyter Notebook display helper
from IPython.display import display
```

```
df = pd.read_csv('./data/file-table.tsv', sep='\t', low_memory=False)
display(df.head())
```

```
       File ID Participants ID                                    Study Name  \
0  GF_000VDK42     PT_PR4YBBH3          Pediatric Brain Tumor Atlas: CBTTC
1  GF_000WBJCD     PT_NK8A49X5           Pediatric Brain Tumor Atlas: PNOC
2  GF_001JWT9N     PT_G16VK7FR           Pediatric Brain Tumor Atlas: PNOC
3  GF_002DRSGP     PT_2HN13G42  Kids First: Congenital Diaphragmatic Hernia
4  GF_004J173A     PT_MH56TZJD  Kids First: Congenital Diaphragmatic Hernia

  Proband    Family Id                  Data Type File Format    File Size  \
0     Yes           --                Aligned Reads         bam  11041645308
1     Yes           --  Annotated Somatic Mutations         maf       122552
2     Yes           --                 Gene Fusions         pdf      5779507
3     Yes  FM_F9S808PW                Aligned Reads        cram  23537329440
4      No  FM_1EFM6M40                Aligned Reads        cram  17290282717

  Participant External ID                                       File Name  \
0                  C29274          62c0c6fe-99f8-4ff7-b3b5-233e6cc2ff0f.bam
1                    P-06  77af1324-3754-4e34-a208-d1342a2f2ca6.mutect2_s...
2                    P-37  9f586dc1-1df8-4f59-9a01-803141fffb94.arriba.fu...
3              CDH14-0006                                     CDH14-0006.cram
4                CDH4-84F                                       CDH4-84F.cram

                                File External ID Aliquot External ID  \
0          62c0c6fe-99f8-4ff7-b3b5-233e6cc2ff0f.bam              746063
1  harmonized/simple-variants/77af1324-3754-4e34-...      A08713, A08710
2  harmonized/gene-fusions/9f586dc1-1df8-4f59-9a0...              A19683
3  s3://kf-study-us-east-1-prd-sd-46sk55a3/source...          CDH14-0006
4  s3://kf-study-us-east-1-prd-sd-46sk55a3/source...            CDH4-84F

           Sample External ID            Biospecimen ID  \
0   7316-126-T-112502.RNA-Seq               BS_A7Q8G0Y1
1  A08692-T.WXS, A08691-N.WXS  BS_6DT506HY, BS_5DPMQQVG
2            A19649-T.RNA-Seq               BS_XGDPK33A
3                  CDH14-0006               BS_BKH9S8YN
4                    CDH4-84F               BS_MP5P3ZPH

  Tissue Type (Source Text)  \
0                     Tumor
1             Tumor, Normal
2                     Tumor
3                    Normal
4                    Normal
```

(continues on next page)

(continued from previous page)

```
                              Diagnosis (Source Text)     Study ID  \
0  Brainstem glioma- Diffuse intrinsic pontine gl...  SD_BHJXBDQK
1  Brainstem glioma- Diffuse intrinsic pontine gl...  SD_M3DBXD12
2  Brainstem glioma- Diffuse intrinsic pontine gl...  SD_M3DBXD12
3                    congential diaphragmatic hernia  SD_46SK55A3
4                                                 --  SD_46SK55A3

                              Latest DID                    Observed Repository
0  bdf6c2f6-1500-4693-ae32-fd18dc4ab9e1                      --       gen3
1  16712090-50f7-4cd1-bf2d-90ce989c2139                      --       gen3
2  49f7aead-ce23-4c38-b566-1d99cb5a5435                      --       gen3
3  c4c9d542-21fb-487d-ac07-916d466774a8  true, false, false, false       gen3
4  02642bc8-ae46-45a1-8932-2765cf1df480                      --       gen3
```

4.4.5 DATS Conversion

The full DATS schema is available here, it includes a JSON Schema definition as well as a visualization of how things fit together.

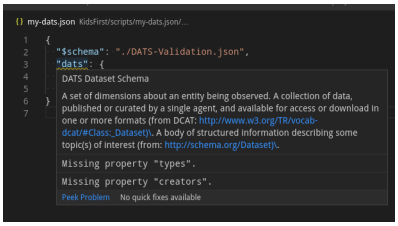
DATS uses a strict JSON-LD serialization.

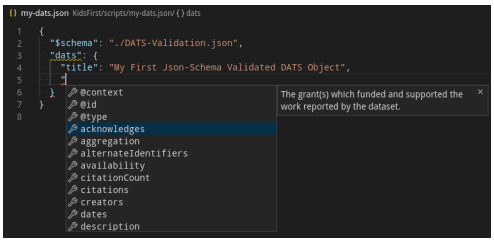
There are several ways to get a sense of what the metadata model entails. In some cases starting with an example is easier, but everything is easier with autocompletion and type-hints. Several code editors support JSON Schema for auto completion (see this).

With visual studio code, you can set this up by linking to the schema with a `$schema` field.

```
%%sh
# Create a file DATS-Validation.json with the following contents
#  this is a simple json-schema which references the public DATS schema validator
cat > DATS-Validation.json << EOF
{
  "\$schema": "http://json-schema.org/draft-04/schema",
  "type": "object",
  "properties": {
    "dats": {
      "\$ref": "https://raw.githubusercontent.com/datatagsuite/schema/master/dataset_
↪schema.json"
    }
  }
}
EOF

# Create a file to edit which will validate against the file from the DATS-Validation␣
↪file
cat > my-dats.json << EOF
{
  "\$schema": "./DATS-Validation.json",
  "dats": {
    "title": "My First Json-Schema Validated DATS Object"
  }
}
EOF
```

Modifying the created `my-dats.json`, you should be able to explore the fields through autocompletion with an editor that supports it.





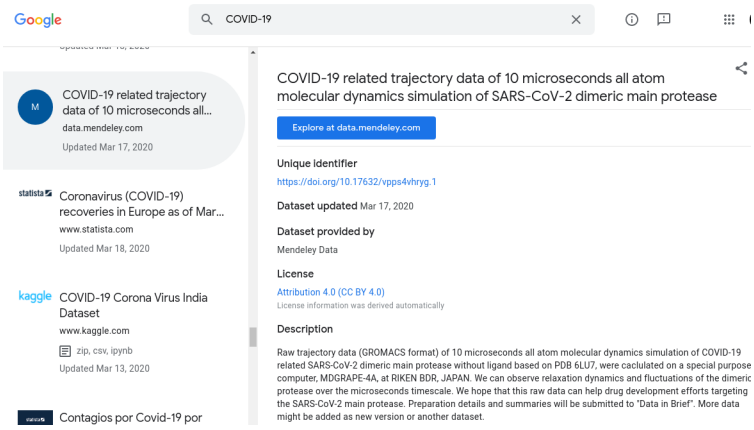
Another way, or perhaps also, is to learn by example. Several *other* DCC's assets were processed and converted to DATS here, example files and scripts can be found in the `DCC_name/output` and `DCC_name/scripts` directories respectively.

It's now time to convert what we can into DATS, striving to capture as much as possible from the original table.

Challenge 1: What do you mean by Dataset?

Even in this case, the definition of a Dataset becomes problematic and unclear. Remember that things we codify are often **models** and as such are not always perfect. Rather than thinking about Dataset with your interpretation of what it is, think of it in terms of how it will end up being used.

This is what a 'dataset' looks like on Google Dataset Search; the same fields will be used, and more; for your own assets.



In other words, irrespective of what your definition is of a 'dataset', you should consider using something that is identifiable enough to have its own unique metadata including dedicated landing page, unique identifier, citation, license, and more. File assets *associated* with that dataset will be listed under the dataset.

Importantly, Datasets should ideally be associatable with singular biosamples when possible, so in some cases, it may make sense to consider each individual file to be its own dataset if each individual file is actually established for every biosample.

Do note that DATS also supports Dataset in Dataset relationships if that becomes necessary.

```
# The KidsFirst table has 5 primary entity types in this file and a unique identifier
display(df[['File ID', 'Participants ID', 'Study ID', 'Biospecimen ID', 'Latest DID
 ↪']].head())

# Using JSON-LD and keeping in mind the arbitrary DATS structure,
#  things should end up looking like so:
def dats_from_record(record):
  return {
    '@type': 'Dataset',
    'identifier': {
      '@type': 'Identifier',
      'identifier': record['Latest DID'],
    },
    'producedBy': {
      # The dataset in question was produced as part of a study
      '@type': 'Study',
      'identifier': {
        '@type': 'Identifier',
        'identifier': record['Study ID'],
      },
    },
    'isAbout': [
      {
        # The dataset in question has a biospecimen
        '@type': 'BiologicalEntity',
        'identifier': {
          '@type': 'Identifier',
          'identifier': record['Biospecimen ID'],
        },
      },
      {
        # The dataset in question is about this participant
        '@type': 'StudyGroup',
        'identifier': {
          '@type': 'Identifier',
          'identifier': record['Participants ID'],
        },
      },
    ],
    'distributions': [
      {
        # The dataset in question has this file
        '@type': 'DatasetDistribution',
        'identifier': {
          '@type': 'Identifier',
          'identifier': record['File ID'],
        },
      }
    ],
  }

# Converting each element to DATS
dats = {
  # schema.org context, gives RDF meaning to `@type` and `predicates` as defined by␣
 ↪schema.org
  '@context': 'http://w3id.org/dats/context/sdo/dataset_sdo_context.jsonld',
  '@graph': [
```

(continues on next page)

(continued from previous page)

```
    dats_from_record(record)
    for _, record in df.head().iterrows()
  ]
}
display(dats)
```

```
       File ID Participants ID     Study ID            Biospecimen ID  \
0  GF_000VDK42     PT_PR4YBBH3  SD_BHJXBDQK               BS_A7Q8G0Y1
1  GF_000WBJCD     PT_NK8A49X5  SD_M3DBXD12  BS_6DT506HY, BS_5DPMQQVG
2  GF_001JWT9N     PT_G16VK7FR  SD_M3DBXD12               BS_XGDPK33A
3  GF_002DRSGP     PT_2HN13G42  SD_46SK55A3               BS_BKH9S8YN
4  GF_004J173A     PT_MH56TZJD  SD_46SK55A3               BS_MP5P3ZPH

                             Latest DID
0  bdf6c2f6-1500-4693-ae32-fd18dc4ab9e1
1  16712090-50f7-4cd1-bf2d-90ce989c2139
2  49f7aead-ce23-4c38-b566-1d99cb5a5435
3  c4c9d542-21fb-487d-ac07-916d466774a8
4  02642bc8-ae46-45a1-8932-2765cf1df480
```

```
{'@context': 'http://w3id.org/dats/context/sdo/dataset_sdo_context.jsonld',
 '@graph': [{'@type': 'Dataset',
   'identifier': {'@type': 'Identifier',
    'identifier': 'bdf6c2f6-1500-4693-ae32-fd18dc4ab9e1'},
   'producedBy': {'@type': 'Study',
    'identifier': {'@type': 'Identifier', 'identifier': 'SD_BHJXBDQK'}},
   'isAbout': [{'@type': 'BiologicalEntity',
     'identifier': {'@type': 'Identifier', 'identifier': 'BS_A7Q8G0Y1'}},
    {'@type': 'StudyGroup',
     'identifier': {'@type': 'Identifier', 'identifier': 'PT_PR4YBBH3'}}],
   'distributions': [{'@type': 'DatasetDistribution',
     'identifier': {'@type': 'Identifier', 'identifier': 'GF_000VDK42'}}]},
  {'@type': 'Dataset',
   'identifier': {'@type': 'Identifier',
    'identifier': '16712090-50f7-4cd1-bf2d-90ce989c2139'},
   'producedBy': {'@type': 'Study',
    'identifier': {'@type': 'Identifier', 'identifier': 'SD_M3DBXD12'}},
   'isAbout': [{'@type': 'BiologicalEntity',
     'identifier': {'@type': 'Identifier',
      'identifier': 'BS_6DT506HY, BS_5DPMQQVG'}},
    {'@type': 'StudyGroup',
     'identifier': {'@type': 'Identifier', 'identifier': 'PT_NK8A49X5'}}],
   'distributions': [{'@type': 'DatasetDistribution',
     'identifier': {'@type': 'Identifier', 'identifier': 'GF_000WBJCD'}}]},
  {'@type': 'Dataset',
   'identifier': {'@type': 'Identifier',
    'identifier': '49f7aead-ce23-4c38-b566-1d99cb5a5435'},
   'producedBy': {'@type': 'Study',
    'identifier': {'@type': 'Identifier', 'identifier': 'SD_M3DBXD12'}},
   'isAbout': [{'@type': 'BiologicalEntity',
     'identifier': {'@type': 'Identifier', 'identifier': 'BS_XGDPK33A'}},
    {'@type': 'StudyGroup',
     'identifier': {'@type': 'Identifier', 'identifier': 'PT_G16VK7FR'}}],
   'distributions': [{'@type': 'DatasetDistribution',
     'identifier': {'@type': 'Identifier', 'identifier': 'GF_001JWT9N'}}]},
```

(continues on next page)

(continued from previous page)

```
 {'@type': 'Dataset',
  'identifier': {'@type': 'Identifier',
   'identifier': 'c4c9d542-21fb-487d-ac07-916d466774a8'},
  'producedBy': {'@type': 'Study',
   'identifier': {'@type': 'Identifier', 'identifier': 'SD_46SK55A3'}},
  'isAbout': [{'@type': 'BiologicalEntity',
    'identifier': {'@type': 'Identifier', 'identifier': 'BS_BKH9S8YN'}},
   {'@type': 'StudyGroup',
    'identifier': {'@type': 'Identifier', 'identifier': 'PT_2HN13G42'}}],
  'distributions': [{'@type': 'DatasetDistribution',
    'identifier': {'@type': 'Identifier', 'identifier': 'GF_002DRSGP'}}]},
 {'@type': 'Dataset',
  'identifier': {'@type': 'Identifier',
   'identifier': '02642bc8-ae46-45a1-8932-2765cf1df480'},
  'producedBy': {'@type': 'Study',
   'identifier': {'@type': 'Identifier', 'identifier': 'SD_46SK55A3'}},
  'isAbout': [{'@type': 'BiologicalEntity',
    'identifier': {'@type': 'Identifier', 'identifier': 'BS_MP5P3ZPH'}},
   {'@type': 'StudyGroup',
    'identifier': {'@type': 'Identifier', 'identifier': 'PT_MH56TZJD'}}],
  'distributions': [{'@type': 'DatasetDistribution',
    'identifier': {'@type': 'Identifier', 'identifier': 'GF_004J173A'}}]}]}
```

There are several improvements we can make to the above:

1. Give context to our identifiers, which only make sense in the context of KidsFirst
2. Provide more metadata as available in our table

```
def dats_from_record(record):
  return {
    '@type': 'Dataset',
    'identifier': {
      '@type': 'Identifier',
      'identifier': record['Latest DID'],
      'identifierSource': 'https://portal.kidsfirstdrc.org/',
    },
    'storedIn': {
      '@type': 'DataRepository',
      'name': record['Repository'],
    },
    'producedBy': {
      '@type': 'Study',
      'identifier': {
        '@type': 'Identifier',
        'identifier': record['Study ID'],
        'identifierSource': 'https://portal.kidsfirstdrc.org/',
      },
      'name': record['Study Name'],
    },
    'isAbout': [
      {
        '@type': 'BiologicalEntity',
        'identifier': {
          '@type': 'Identifier',
          'identifier': record['Biospecimen ID'],
          'identifierSource': 'https://portal.kidsfirstdrc.org/',
```

(continues on next page)

(continued from previous page)

```
        },
        'alternateIdentifiers': [
          {
            '@type': 'AlternateIdentifier',
            'identifier': record['Sample External ID'],
            # NOTE: Preferred identifierSource with globally unique semantic URI
          },
          {
            '@type': 'AlternateIdentifier',
            'identifier': record['Aliquot External ID'],
            # NOTE: Preferred identifierSource with globally unique semantic URI
          },
        ],
      },
      {
        '@type': 'StudyGroup',
        'identifier': {
          # NOTE: Ideally, `${identifierSource}${identifier}` resolves to a landing
↪page for this entity
          '@type': 'Identifier',
          'identifier': record['Participants ID'],
          'identifierSource': 'https://portal.kidsfirstdrc.org/participant/',
        },
        'alternateIdentifiers': [
          {
            '@type': 'AlternateIdentifier',
            'identifier': record['Participant External ID'],
            # NOTE: Preferred identifierSource with globally unique semantic URI
          },
        ],
      },
    ],
    'distributions': [
      {
        'identifier': {
          # NOTE: Ideally, `${identifierSource}${identifier}` resolves to a landing
↪page for this entity
          '@type': 'Identifier',
          'identifier': record['File ID'],
          'identifierSource': 'https://portal.kidsfirstdrc.org/file/',
        },
        '@type': 'DatasetDistribution',
        'formats': [
          record['File Format'],
        ],
        'size': record['File Size'],
        'unit': {
          '@type': 'Annotation',
          'value': 'bytes',
          # NOTE: Preferred valueIRI with globally unique semantic URI
        },
        'access': {
          '@type': 'Access',
          'identifier': {
            '@type': 'Identifier',
            'identifier': record['File Name'],
```

(continues on next page)

(continued from previous page)

```
          },
          'alternateIdentifiers': [
            {
              '@type': 'AlternateIdentifier',
              'identifier': record['File External ID'],
              # NOTE: Preferred identifierSource with globally unique semantic URI
            },
          ],
          'landingPage': 'https://portal.kidsfirstdrc.org/file/' + record['File ID'],
          # NOTE: Ideally accessURL would be specified
        }
      }
    ],
    'types': [
      {
        '@type': 'DataType',
        'information': {
          '@type': 'Annotation',
          'value': record['Data Type'],
        },
      },
    ],
    'extraProperties': [*filter(None, [
      # Metadata that doesn't fit anywhere else in DATS but may be relevant
      {
        '@type': 'CategoryValuesPair',
        'category': 'tissue',
        # NOTE: Preferred categoryIRI with globally unique semantic URI
        'values': [
          {
            '@type': 'Annotation',
            'value': record['Tissue Type (Source Text)'],
            # NOTE: Preferred valueIRI with globally unique semantic URI
          }
        ]
      } if record['Tissue Type (Source Text)'] != '--' else None,
      {
        '@type': 'CategoryValuesPair',
        'category': 'diagnosis',
        # NOTE: Preferred categoryIRI with globally unique semantic URI
        'values': [
          {
            '@type': 'Annotation',
            'value': record['Diagnosis (Source Text)'],
            # NOTE: Preferred valueIRI with globally unique semantic URI
          }
        ]
      } if record['Diagnosis (Source Text)'] != '--' else None, # Don't create␣
↪entries for junk
      {
        '@type': 'CategoryValuesPair',
        'category': 'proband',
        # NOTE: Preferred categoryIRI with globally unique semantic URI
        'values': [
          {
            '@type': 'Annotation',
```

(continues on next page)

(continued from previous page)

```
            'value': record['Proband'],
            # NOTE: Preferred valueIRI with globally unique semantic URI
          },
        ],
      } if record['Proband'] != '--' else None, # Don't create entries for junk,
    ])],
  }

# Converting each element to DATS
dats = {
  # schema.org context, gives RDF meaning to `@type` and `predicates` as defined by␣
 ↪schema.org
  '@context': 'http://w3id.org/dats/context/sdo/dataset_sdo_context.jsonld',
  '@graph': [
    dats_from_record(record)
    for _, record in df.head().iterrows()
  ]
}
display(dats)
```


```
{'@context': 'http://w3id.org/dats/context/sdo/dataset_sdo_context.jsonld',
 '@graph': [{'@type': 'Dataset',
   'identifier': {'@type': 'Identifier',
    'identifier': 'bdf6c2f6-1500-4693-ae32-fd18dc4ab9e1',
    'identifierSource': 'https://portal.kidsfirstdrc.org/'},
   'storedIn': {'@type': 'DataRepository', 'name': 'gen3'},
   'producedBy': {'@type': 'Study',
    'identifier': {'@type': 'Identifier',
     'identifier': 'SD_BHJXBDQK',
     'identifierSource': 'https://portal.kidsfirstdrc.org/'},
    'name': 'Pediatric Brain Tumor Atlas: CBTTC'},
   'isAbout': [{'@type': 'BiologicalEntity',
     'identifier': {'@type': 'Identifier',
      'identifier': 'BS_A7Q8G0Y1',
      'identifierSource': 'https://portal.kidsfirstdrc.org/'},
     'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
       'identifier': '7316-126-T-112502.RNA-Seq'},
      {'@type': 'AlternateIdentifier', 'identifier': '746063'}]},
    {'@type': 'StudyGroup',
     'identifier': {'@type': 'Identifier',
      'identifier': 'PT_PR4YBBH3',
      'identifierSource': 'https://portal.kidsfirstdrc.org/participant/'},
     'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
       'identifier': 'C29274'}]}],
   'distributions': [{'identifier': {'@type': 'Identifier',
      'identifier': 'GF_000VDK42',
      'identifierSource': 'https://portal.kidsfirstdrc.org/file/'},
     '@type': 'DatasetDistribution',
     'formats': ['bam'],
     'size': 11041645308,
     'unit': {'@type': 'Annotation', 'value': 'bytes'},
     'access': {'@type': 'Access',
      'identifier': {'@type': 'Identifier',
       'identifier': '62c0c6fe-99f8-4ff7-b3b5-233e6cc2ff0f.bam'},
      'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
```


(continues on next page)

(continued from previous page)

```
        'identifier': '62c0c6fe-99f8-4ff7-b3b5-233e6cc2ff0f.bam'}],
      'landingPage': 'https://portal.kidsfirstdrc.org/file/GF_000VDK42'}}],
  'types': [{'@type': 'DataType',
    'information': {'@type': 'Annotation', 'value': 'Aligned Reads'}}],
  'extraProperties': [{'@type': 'CategoryValuesPair',
    'category': 'tissue',
    'values': [{'@type': 'Annotation', 'value': 'Tumor'}]},
   {'@type': 'CategoryValuesPair',
    'category': 'diagnosis',
    'values': [{'@type': 'Annotation',
      'value': 'Brainstem glioma- Diffuse intrinsic pontine glioma, Brainstem␣
↪glioma- Diffuse intrinsic pontine glioma'}]},
   {'@type': 'CategoryValuesPair',
    'category': 'proband',
    'values': [{'@type': 'Annotation', 'value': 'Yes'}]}]},
 {'@type': 'Dataset',
  'identifier': {'@type': 'Identifier',
   'identifier': '16712090-50f7-4cd1-bf2d-90ce989c2139',
   'identifierSource': 'https://portal.kidsfirstdrc.org/'},
  'storedIn': {'@type': 'DataRepository', 'name': 'gen3'},
  'producedBy': {'@type': 'Study',
   'identifier': {'@type': 'Identifier',
    'identifier': 'SD_M3DBXD12',
    'identifierSource': 'https://portal.kidsfirstdrc.org/'},
   'name': 'Pediatric Brain Tumor Atlas: PNOC'},
  'isAbout': [{'@type': 'BiologicalEntity',
    'identifier': {'@type': 'Identifier',
     'identifier': 'BS_6DT506HY, BS_5DPMQQVG',
     'identifierSource': 'https://portal.kidsfirstdrc.org/'},
    'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
      'identifier': 'A08692-T.WXS, A08691-N.WXS'},
     {'@type': 'AlternateIdentifier', 'identifier': 'A08713, A08710'}]},
   {'@type': 'StudyGroup',
    'identifier': {'@type': 'Identifier',
     'identifier': 'PT_NK8A49X5',
     'identifierSource': 'https://portal.kidsfirstdrc.org/participant/'},
    'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
      'identifier': 'P-06'}]}],
  'distributions': [{'identifier': {'@type': 'Identifier',
     'identifier': 'GF_000WBJCD',
     'identifierSource': 'https://portal.kidsfirstdrc.org/file/'},
    '@type': 'DatasetDistribution',
    'formats': ['maf'],
    'size': 122552,
    'unit': {'@type': 'Annotation', 'value': 'bytes'},
    'access': {'@type': 'Access',
     'identifier': {'@type': 'Identifier',
      'identifier': '77af1324-3754-4e34-a208-d1342a2f2ca6.mutect2_somatic.vep.maf
↪'},
     'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
       'identifier': 'harmonized/simple-variants/77af1324-3754-4e34-a208-
↪d1342a2f2ca6.mutect2_somatic.vep.maf'}],
     'landingPage': 'https://portal.kidsfirstdrc.org/file/GF_000WBJCD'}}],
  'types': [{'@type': 'DataType',
    'information': {'@type': 'Annotation',
     'value': 'Annotated Somatic Mutations'}}],
```

(continues on next page)

(continued from previous page)

```
   'extraProperties': [{'@type': 'CategoryValuesPair',
     'category': 'tissue',
     'values': [{'@type': 'Annotation', 'value': 'Tumor, Normal'}]},
    {'@type': 'CategoryValuesPair',
     'category': 'diagnosis',
     'values': [{'@type': 'Annotation',
       'value': 'Brainstem glioma- Diffuse intrinsic pontine glioma'}]},
    {'@type': 'CategoryValuesPair',
     'category': 'proband',
     'values': [{'@type': 'Annotation', 'value': 'Yes'}]}]},
  {'@type': 'Dataset',
   'identifier': {'@type': 'Identifier',
    'identifier': '49f7aead-ce23-4c38-b566-1d99cb5a5435',
    'identifierSource': 'https://portal.kidsfirstdrc.org/'},
   'storedIn': {'@type': 'DataRepository', 'name': 'gen3'},
   'producedBy': {'@type': 'Study',
    'identifier': {'@type': 'Identifier',
     'identifier': 'SD_M3DBXD12',
     'identifierSource': 'https://portal.kidsfirstdrc.org/'},
    'name': 'Pediatric Brain Tumor Atlas: PNOC'},
   'isAbout': [{'@type': 'BiologicalEntity',
     'identifier': {'@type': 'Identifier',
      'identifier': 'BS_XGDPK33A',
      'identifierSource': 'https://portal.kidsfirstdrc.org/'},
     'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
       'identifier': 'A19649-T.RNA-Seq'},
      {'@type': 'AlternateIdentifier', 'identifier': 'A19683'}]},
    {'@type': 'StudyGroup',
     'identifier': {'@type': 'Identifier',
      'identifier': 'PT_G16VK7FR',
      'identifierSource': 'https://portal.kidsfirstdrc.org/participant/'},
     'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
       'identifier': 'P-37'}]}],
   'distributions': [{'identifier': {'@type': 'Identifier',
      'identifier': 'GF_001JWT9N',
      'identifierSource': 'https://portal.kidsfirstdrc.org/file/'},
     '@type': 'DatasetDistribution',
     'formats': ['pdf'],
     'size': 5779507,
     'unit': {'@type': 'Annotation', 'value': 'bytes'},
     'access': {'@type': 'Access',
      'identifier': {'@type': 'Identifier',
       'identifier': '9f586dc1-1df8-4f59-9a01-803141fffb94.arriba.fusions.pdf'},
      'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
        'identifier': 'harmonized/gene-fusions/9f586dc1-1df8-4f59-9a01-
↪803141fffb94.arriba.fusions.pdf'}],
      'landingPage': 'https://portal.kidsfirstdrc.org/file/GF_001JWT9N'}}],
   'types': [{'@type': 'DataType',
     'information': {'@type': 'Annotation', 'value': 'Gene Fusions'}}],
   'extraProperties': [{'@type': 'CategoryValuesPair',
     'category': 'tissue',
     'values': [{'@type': 'Annotation', 'value': 'Tumor'}]},
    {'@type': 'CategoryValuesPair',
     'category': 'diagnosis',
     'values': [{'@type': 'Annotation',
       'value': 'Brainstem glioma- Diffuse intrinsic pontine glioma'}]},
```

(continues on next page)

(continued from previous page)

```
    {'@type': 'CategoryValuesPair',
     'category': 'proband',
     'values': [{'@type': 'Annotation', 'value': 'Yes'}]}]},
  {'@type': 'Dataset',
   'identifier': {'@type': 'Identifier',
    'identifier': 'c4c9d542-21fb-487d-ac07-916d466774a8',
    'identifierSource': 'https://portal.kidsfirstdrc.org/'},
   'storedIn': {'@type': 'DataRepository', 'name': 'gen3'},
   'producedBy': {'@type': 'Study',
    'identifier': {'@type': 'Identifier',
     'identifier': 'SD_46SK55A3',
     'identifierSource': 'https://portal.kidsfirstdrc.org/'},
    'name': 'Kids First: Congenital Diaphragmatic Hernia'},
   'isAbout': [{'@type': 'BiologicalEntity',
     'identifier': {'@type': 'Identifier',
      'identifier': 'BS_BKH9S8YN',
      'identifierSource': 'https://portal.kidsfirstdrc.org/'},
     'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
       'identifier': 'CDH14-0006'},
      {'@type': 'AlternateIdentifier', 'identifier': 'CDH14-0006'}]},
    {'@type': 'StudyGroup',
     'identifier': {'@type': 'Identifier',
      'identifier': 'PT_2HN13G42',
      'identifierSource': 'https://portal.kidsfirstdrc.org/participant/'},
     'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
       'identifier': 'CDH14-0006'}]}],
   'distributions': [{'identifier': {'@type': 'Identifier',
      'identifier': 'GF_002DRSGP',
      'identifierSource': 'https://portal.kidsfirstdrc.org/file/'},
     '@type': 'DatasetDistribution',
     'formats': ['cram'],
     'size': 23537329440,
     'unit': {'@type': 'Annotation', 'value': 'bytes'},
     'access': {'@type': 'Access',
      'identifier': {'@type': 'Identifier', 'identifier': 'CDH14-0006.cram'},
      'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
        'identifier': 's3://kf-study-us-east-1-prd-sd-46sk55a3/source/GMKF_Gabriel_
↪Chung_CDH_WGS/RP-1370/WGS/CDH14-0006/v2/CDH14-0006.cram'}],
      'landingPage': 'https://portal.kidsfirstdrc.org/file/GF_002DRSGP'}}],
   'types': [{'@type': 'DataType',
     'information': {'@type': 'Annotation', 'value': 'Aligned Reads'}}],
   'extraProperties': [{'@type': 'CategoryValuesPair',
     'category': 'tissue',
     'values': [{'@type': 'Annotation', 'value': 'Normal'}]},
    {'@type': 'CategoryValuesPair',
     'category': 'diagnosis',
     'values': [{'@type': 'Annotation',
       'value': 'congential diaphragmatic hernia'}]},
    {'@type': 'CategoryValuesPair',
     'category': 'proband',
     'values': [{'@type': 'Annotation', 'value': 'Yes'}]}]},
  {'@type': 'Dataset',
   'identifier': {'@type': 'Identifier',
    'identifier': '02642bc8-ae46-45a1-8932-2765cf1df480',
    'identifierSource': 'https://portal.kidsfirstdrc.org/'},
   'storedIn': {'@type': 'DataRepository', 'name': 'gen3'},
```

(continues on next page)

(continued from previous page)

```
   'producedBy': {'@type': 'Study',
    'identifier': {'@type': 'Identifier',
     'identifier': 'SD_46SK55A3',
     'identifierSource': 'https://portal.kidsfirstdrc.org/'},
    'name': 'Kids First: Congenital Diaphragmatic Hernia'},
   'isAbout': [{'@type': 'BiologicalEntity',
     'identifier': {'@type': 'Identifier',
      'identifier': 'BS_MP5P3ZPH',
      'identifierSource': 'https://portal.kidsfirstdrc.org/'},
     'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
       'identifier': 'CDH4-84F'},
      {'@type': 'AlternateIdentifier', 'identifier': 'CDH4-84F'}]},
    {'@type': 'StudyGroup',
     'identifier': {'@type': 'Identifier',
      'identifier': 'PT_MH56TZJD',
      'identifierSource': 'https://portal.kidsfirstdrc.org/participant/'},
     'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
       'identifier': 'CDH4-84F'}]}],
   'distributions': [{'identifier': {'@type': 'Identifier',
      'identifier': 'GF_004J173A',
      'identifierSource': 'https://portal.kidsfirstdrc.org/file/'},
     '@type': 'DatasetDistribution',
     'formats': ['cram'],
     'size': 17290282717,
     'unit': {'@type': 'Annotation', 'value': 'bytes'},
     'access': {'@type': 'Access',
      'identifier': {'@type': 'Identifier', 'identifier': 'CDH4-84F.cram'},
      'alternateIdentifiers': [{'@type': 'AlternateIdentifier',
        'identifier': 's3://kf-study-us-east-1-prd-sd-46sk55a3/source/GMKF_Gabriel_
↪Chung_CDH_WGS/RP-1370/WGS/CDH4-84F/v2/CDH4-84F.cram'}],
      'landingPage': 'https://portal.kidsfirstdrc.org/file/GF_004J173A'}}],
   'types': [{'@type': 'DataType',
     'information': {'@type': 'Annotation', 'value': 'Aligned Reads'}}],
   'extraProperties': [{'@type': 'CategoryValuesPair',
     'category': 'tissue',
     'values': [{'@type': 'Annotation', 'value': 'Normal'}]},
    {'@type': 'CategoryValuesPair',
     'category': 'proband',
     'values': [{'@type': 'Annotation', 'value': 'No'}]}]}]}
```

Now we see, with some mapping effort, we were able to get all of the metadata from the file manifest table into DATS. It is important to note that there are **manys** fields missing including license, authorship information, and more. These fields need to be found from other places to further complete and improve this model. With our newly created object, let's check to make sure we did not make any mistakes! For this purpose, just as we can use json-schema for auto completion help in our editor, we can also use it for programatic validation of our `dats` object.

```
from jsonschema import Draft4Validator

# Get the first record
record = dats['@graph'][0]

# Validate it against DATS dataset schema
validator = Draft4Validator({'$ref': 'https://raw.githubusercontent.com/datatagsuite/
↪schema/master/dataset_schema.json'})
for error in validator.iter_errors(record):
  display(error.message)
```

```
"{'@type': 'BiologicalEntity', 'identifier': {'@type': 'Identifier', 'identifier':
 ↪'BS_A7Q8G0Y1', 'identifierSource': 'https://portal.kidsfirstdrc.org/'},
 ↪'alternateIdentifiers': [{'@type': 'AlternateIdentifier', 'identifier': '7316-
 ↪126-T-112502.RNA-Seq'}, {'@type': 'AlternateIdentifier', 'identifier': '746063'}
 ↪]} is not valid under any of the given schemas"
```

```
"{'@type': 'StudyGroup', 'identifier': {'@type': 'Identifier', 'identifier': 'PT_
 ↪PR4YBBH3', 'identifierSource': 'https://portal.kidsfirstdrc.org/participant/'},
 ↪'alternateIdentifiers': [{'@type': 'AlternateIdentifier', 'identifier': 'C29274'}
 ↪]} is not valid under any of the given schemas"
```

```
"'title' is a required property"
```

```
"'creators' is a required property"
```

Uh-oh; we've got some errors.

Let's fix them and try again.

For readability, the changes we had to make below are here:

```
@@ -1,6 +1,7 @@
 def dats_from_record(record):
   return {
     '@type': 'Dataset',
+    'title': record['Study Name'],
     'identifier': {
       '@type': 'Identifier',
       'identifier': record['Latest DID'],
@@ -10,6 +11,12 @@
       '@type': 'DataRepository',
       'name': record['Repository'],
     },
+    'creators': [
+      {
+        "@type": "Organization",
+        "name": "KidsFirst",
+      }
+    ],
     'producedBy': {
       '@type': 'Study',
       'identifier': {
@@ -22,6 +29,8 @@
     'isAbout': [
       {
         '@type': 'BiologicalEntity',
+        # NOTE: name is a required field
+        'name': record['Biospecimen ID'],
         'identifier': {
           '@type': 'Identifier',
           'identifier': record['Biospecimen ID'],
@@ -42,6 +51,8 @@
       },
       {
         '@type': 'StudyGroup',
```

(continues on next page)

(continued from previous page)

```
+         # NOTE: name is a required field
+         'name': record['Participants ID'],
          'identifier': {
            # NOTE: Ideally, `${identifierSource}${identifier}` resolves to a landing␣
 ↪page for this entity
            '@type': 'Identifier',
@@ -69,7 +80,7 @@
          'formats': [
            record['File Format'],
          ],
-         'size': record['File Size'],
+         'size': int(record['File Size']),
          'unit': {
            '@type': 'Annotation',
            'value': 'bytes',
@@ -141,4 +152,4 @@
          ],
        } if record['Proband'] != '--' else None, # Don't create entries for junk,
      ])],
-   }
+   }
```

You can see that we put in an invalid type and were missing some fields. In some cases, we need to add metadata that wasn't in the original table such as metadata about our own organization!

This is relevant in a catalog of many datasets but often isn't present in your own data; it's best if you determine how your own data will link back to your organization, than us trying to figure it out! That's why DATS requires that metadata.

```
def dats_from_record(record):
  return {
    '@type': 'Dataset',
    'title': record['Study Name'],
    'identifier': {
      '@type': 'Identifier',
      'identifier': record['Latest DID'],
      'identifierSource': 'https://portal.kidsfirstdrc.org/',
    },
    'storedIn': {
      '@type': 'DataRepository',
      'name': record['Repository'],
    },
    'creators': [
      {
        "@type": "Organization",
        "name": "KidsFirst",
      }
    ],
    'producedBy': {
      '@type': 'Study',
      'identifier': {
        '@type': 'Identifier',
        'identifier': record['Study ID'],
        'identifierSource': 'https://portal.kidsfirstdrc.org/',
      },
      'name': record['Study Name'],
    },
```

(continues on next page)

(continued from previous page)

```
    'isAbout': [
      {
        '@type': 'BiologicalEntity',
        # NOTE: name is a required field
        'name': record['Biospecimen ID'],
        'identifier': {
          '@type': 'Identifier',
          'identifier': record['Biospecimen ID'],
          'identifierSource': 'https://portal.kidsfirstdrc.org/',
        },
        'alternateIdentifiers': [
          {
            '@type': 'AlternateIdentifier',
            'identifier': record['Sample External ID'],
            # NOTE: Preferred identifierSource with globally unique semantic URI
          },
          {
            '@type': 'AlternateIdentifier',
            'identifier': record['Aliquot External ID'],
            # NOTE: Preferred identifierSource with globally unique semantic URI
          },
        ],
      },
      {
        '@type': 'StudyGroup',
        # NOTE: name is a required field
        'name': record['Participants ID'],
        'identifier': {
          # NOTE: Ideally, `${identifierSource}${identifier}` resolves to a landing␣
↪page for this entity
          '@type': 'Identifier',
          'identifier': record['Participants ID'],
          'identifierSource': 'https://portal.kidsfirstdrc.org/participant/',
        },
        'alternateIdentifiers': [
          {
            '@type': 'AlternateIdentifier',
            'identifier': record['Participant External ID'],
            # NOTE: Preferred identifierSource with globally unique semantic URI
          },
        ],
      },
    ],
    'distributions': [
      {
        'identifier': {
          # NOTE: Ideally, `${identifierSource}${identifier}` resolves to a landing␣
↪page for this entity
          '@type': 'Identifier',
          'identifier': record['File ID'],
          'identifierSource': 'https://portal.kidsfirstdrc.org/file/',
        },
        '@type': 'DatasetDistribution',
        'formats': [
          record['File Format'],
        ],
```

(continues on next page)

(continued from previous page)

```
        'size': int(record['File Size']),
        'unit': {
          '@type': 'Annotation',
          'value': 'bytes',
          # NOTE: Preferred valueIRI with globally unique semantic URI
        },
        'access': {
          '@type': 'Access',
          'identifier': {
            '@type': 'Identifier',
            'identifier': record['File Name'],
          },
          'alternateIdentifiers': [
            {
              '@type': 'AlternateIdentifier',
              'identifier': record['File External ID'],
              # NOTE: Preferred identifierSource with globally unique semantic URI
            },
          ],
          'landingPage': 'https://portal.kidsfirstdrc.org/file/' + record['File ID'],
          # NOTE: Ideally accessURL would be specified
        }
      }
    ],
    'types': [
      {
        '@type': 'DataType',
        'information': {
          '@type': 'Annotation',
          'value': record['Data Type'],
        },
      },
    ],
    'extraProperties': [*filter(None, [
      # Metadata that doesn't fit anywhere else in DATS but may be relevant
      {
        '@type': 'CategoryValuesPair',
        'category': 'tissue',
        # NOTE: Preferred categoryIRI with globally unique semantic URI
        'values': [
          {
            '@type': 'Annotation',
            'value': record['Tissue Type (Source Text)'],
            # NOTE: Preferred valueIRI with globally unique semantic URI
          }
        ]
      } if record['Tissue Type (Source Text)'] != '--' else None,
      {
        '@type': 'CategoryValuesPair',
        'category': 'diagnosis',
        # NOTE: Preferred categoryIRI with globally unique semantic URI
        'values': [
          {
            '@type': 'Annotation',
            'value': record['Diagnosis (Source Text)'],
            # NOTE: Preferred valueIRI with globally unique semantic URI
```

(continues on next page)

(continued from previous page)

```
          }
        ]
      } if record['Diagnosis (Source Text)'] != '--' else None, # Don't create␣
 ↪entries for junk
      {
        '@type': 'CategoryValuesPair',
        'category': 'proband',
        # NOTE: Preferred categoryIRI with globally unique semantic URI
        'values': [
          {
            '@type': 'Annotation',
            'value': record['Proband'],
            # NOTE: Preferred valueIRI with globally unique semantic URI
          },
        ],
      } if record['Proband'] != '--' else None, # Don't create entries for junk,
    ])],
  }

# Converting each element to DATS
dats = {
  # schema.org context, gives RDF meaning to `@type` and `predicates` as defined by␣
 ↪schema.org
  '@context': 'http://w3id.org/dats/context/sdo/dataset_sdo_context.jsonld',
  '@graph': [
    dats_from_record(record)
    for _, record in df.head().iterrows()
  ]
}
```

```
# Let's validate *all records*
record = dats['@graph'][0]

# Validate it against DATS dataset schema
validator = Draft4Validator({
  '$ref': 'https://raw.githubusercontent.com/datatagsuite/schema/master/dataset_
 ↪schema.json'
})

for record in dats['@graph']:
  for error in validator.iter_errors(record):
    display({ 'title': record['title'], 'error': error.message })
```

4.4.6 Conclusion

As hoped, everything validates and we have successfully produced DATS. Though we now know our DATS is "valid", we're still not done. As with everything there are levels; the more fields we fill out in the DATS the better off we will be. This is where a FAIR assessment comes in – we can write metrics that *also speak DATS*, but are looking for presence of certain fields, or checking that our `identifier` can actually be verified against the given `identifierSource` metadata attributes.

Nonetheless, we have taken a step in the right direction. Future recipes will discuss performing FAIR assessments on this DATS, converting it to CFDE's C2M2 Frictionless Metadata model and more!

4.5 Experience from Kids First (KF)

Authors: Abhijna Parigi, Marisa Lim

Maintainers: Abhijna Parigi, Marisa Lim

Version: 1.0

4.5.1 Objectives:

This is a cookbook recipe documenting the process of inputting the Kids First data into the C2M2 model to produce `Level 1 tables`.

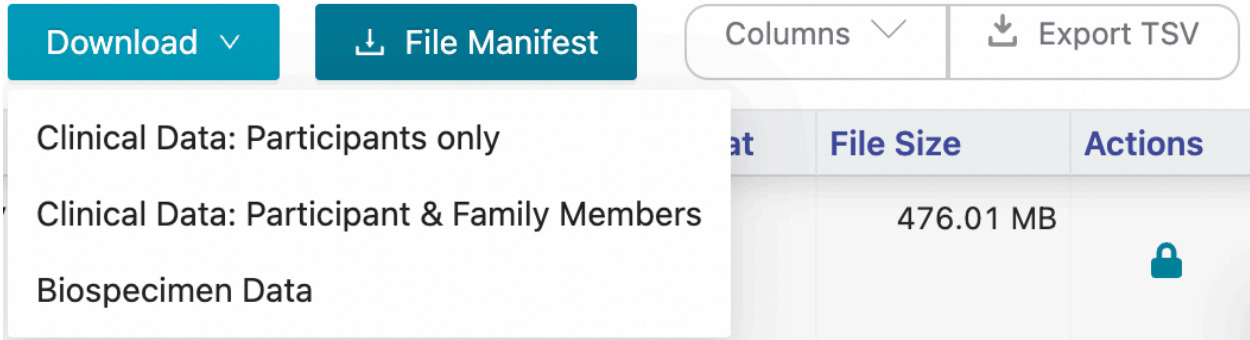
This conversion process is also known as the ETL, which stands for Extract, Transform, and Load.

4.5.2 Step by Step Process

Step 1: Download data from the Kids First (KF) portal

- Visit the KF portal website: https://portal.kidsfirstdrc.org/dashboard
- Log in using your `ORCID credentials` (preferred), `gmail` or `facebook` credentials.
- Select the `File Repository` tab on the main navigation bar at the top of the website.

- Download the Kids First Data by:
  - Clicking the columns options and select all columns. Then click `Export TSV`.
  - Click `File Manifest`.
  - Click `Download` and choose the option `Biospecimen Data` (3rd option of the dropdown menu)



Step 2: Data pre-processing

- Initial preprocessing: remove all the columns that do NOT have any headers.
- Go to the C2M2 documentation page and look for the diagram labeled "C2M2 model diagram". This diagram is important as it shows the "core tables", colored blue and black, as well as the associate tables needed to map the KF datasets to the C2M2 model. Tables 1-4 shown below are examples of mapping used for the "core tables" and table 5 is an examples used for the associate tables. The number of rows for each table coressponds to the number of unique `id` entries.

Note: id_namespace and project_id_namespace have repeating values of cfde_id_namespace:3.

Table 1: file.tsv

source: exported TSV file

C2M2 colnames	KF colnames
id_namespace	
id	File ID
project_id_namespace	
project	Study ID
persistent_id	
creation_time	
size_in_bytes	File Size
sha256	
md5	
filename	File Name
file_format	File Format
data_type	Data Type

Table 2: biosample.tsv

source of `Biospecimen ID`: exported TSV file source of `Experiment Strategy`: `File Manifest` source of `Composition`: `Biospecimen Data`

C2M2 colnames	KF colnames
id_namespace	
id	Biospecimen ID
project_if_namespace	
project	Study ID
persistent_id	
creation_time	
assay_type	Experiment Strategy
anatomy	Composition

Table 3: subject.tsv

source: exported TSV file

C2M2 colnames	KF colnames
id_namespace	
id	Participants ID
project_if_namespace	
project	Study ID
persistent_id	
creation_time	
granularity	.

Note: granularity has this repeating value: cfde_subject_granularity:0. persistent_id and creation_time were left empty.

Table 4: project.tsv

source: exported TSV file

C2M2 colnames	KF colnames
id_namespace	
id	Study ID
persistent_id	
creation_time	
abbreviation	
name	Study Name

Note: persistent_id, creation_time, and abbreviation were left empty.

Table 5: subject_role_taxonomy.tsv

source: exported TSV file

C2M2 colnames	KF colnames
id_namespace	
id	Participants ID
role_id	
taxonomy_id	.

:octocat: warning: `role_id` has this repeating value: `cfde_id_namespace:3` `taxonomy_id` has this repeating value: `NCBI:txid9606`

- Find and replace KF variables in raw data TSVs with C2M2's controlled vocabulary (CV):
  - `file_format`: replace with corresponding EDAM terms from this link
  - `data_type`: replace with corresponding data terms from this link. Look for the `data:` tag.
  - `anatomy`: replcae with corresponding UBERON ID from this link
  - `assay_type` : replace with corresponding terms from this link. This was done programatically (in R) for the first pass because there are only 3 possible values.

```
WGS: OBI:0002117, whole-genome sequencing assay
RNA-Seq: OBI:0001271, RNA-seq assay
WXS: OBI:0002118, exome sequencing assay
```

Step 3: Find the gold tables

- The gold tables are supplied by CFDE to the DCC and contain information that goes into the blue and green tables. They can be downloaded there.
- If you are using the default Git repo structure for KF trial dataset, ensure that the three gold tables are in the folder titled `KF_sample_C2M2_Level_1_bdbag.contents`

Step 4: Add empty association tables

- All association tables (i.e. all lighter blue and grey tables in the diagrams) must be provided in the folder containing the gold tables mentioned in step 3.
- These tables must have the right column names, in the right order, but may remain otherwise empty.

Step 5: Running R script to wrangle data

- Once you have downloaded your data, pre-processed it and made sure the columns are chosen, open up the R script
- Ensure that the path to the raw data tables is set correctly.
- Search (cmd + F) the script for `write.table()`. There should be 5 results. At each spot, make sure you edit the path to the folder in which you wish to write your .tsv files.
- Run the entire script.
- Navigate to your output folder and check the files.
- If following this Git repo structure, paste all newly created tsvs into the folder called `KF_sample_C2M2_Level_1_bdbag.contents`.

Step 6: Building 'green' tables from core entity tables

- This term-scanner script (with modifications to input/output path code) is used to auto-generate the green tables for the C2M2 Model Level 1 model. Currently, this script generates four of the five green tables for Level 1.
- Default paths direct to the HMP example tsv files.

Inputs

- It currently takes in `biosample.tsv` and `file.tsv` (two of the core-entity ETL instance TSVs, aka two of the three black tables) from the `--draftDir` (default is `../draft-C2M2_example_submission_data/HMP__sample_C2M2_Level_1_bdbag.contents`)
- It will load OBO and ontology files from `--cvRefDir` (default is `external_CV_reference_files`):

  `EDAM.version_1.21.tsv`

  `OBI.version_2019-08-15.obo`

  `uberon.version_2019-06-27.obo`

Run script

The term-scanner script is named `build_term_tables.py` and you can run it like so:

```
# with default directory locations: change directory to `model`
cd ./model
python build_term_tables.py

# full command, if not using any default paths
./build_term_tables.py --draftDir [path/to/tsv/file/dir] --cvRefDir [path/to/external/
↪CV/ref/files/dir] --outDir [dir/path/where/you/want/outputs/saved]
```

Run it with `-h` for command line help.

Outputs

It will produce these four green tables for Level 1: `file_format.tsv`,`data_type.tsv`, `assay_type.tsv`, and `anatomy.tsv`. The outputs are saved in `--outDir` (default is `./007_HMP-specific_CV_term_usage_TSVs`).

Step 7: After all tables are created

- Double check that all newly created tsvs (NOT raw data) are in the `KF_sample_C2M2_Level_1_bdbag.contents` folder.
- Add the `C2M2_Level_1.datapackage.json` file to this folder.
- Send the repo to CFDE tech folks.

4.5.3 Conclusion

- This recipe provides an examplar of how to convert a dataset from a DCC, KidsFirst in this example, into the format used by CFDE format for persistence into the DERIVA system.
- Other examples of transformation are available or will be made available as guidance.

4.5.4 What to read next?

- *CFDE C2M2 model*
- *ETL to CFDE C2M2 model*

4.6 Experience from LINCS

A cookbook recipe documenting the easy extract & transform (ETL) process of fitting the LINCS data into the C2M2 model.

Authors: Daniel J. B. Clarke

Maintainers: Daniel J. B. Clarke

Version: 1.2

License: CC0 1.0 Universal (CC0 1.0) Public Domain Dedication

---

4.6.1 Objectives

- Demonstrate an example of an ETL process for preparing data for the C2M2
- Introduce tooling that can make this process easier
- Discuss necessary tradeoffs made to ensure maximal harmonization in the LINCS case

4.6.2 Introduction

The LINCS Data Portal provides various ways to browse the existing LINCS data and highlight which small molecules, cell lines, genes, proteins, and antibodies are featured in a given dataset.

While LINCS revolves around *Datasets*, encompassing in some cases several files for each data level, the C2M2 models information revolving around files. Fortunately, the C2M2 Level 1 provides collections, which enable arbitrary meaningful groupings of files, which is how the LINCS datasets can fit.

4.6.3 Step by Step Process

Step 1: Setting up your environment

The C2M2 Level 1 is described using a Frictionless datapackage specification, which permits various additional tooling to be devised around it. In the case that your data is already accessible in a large table, it may, in some cases, be simpler to construct mysql views from your data that are oriented in a C2M2 compliant manner and can be validated with datapackages. Here however, we will demonstrate producing a C2M2-compliant datapackage directly from the LINCS REST API.

A python helper class was devised to simplify codification of C2M2-compliant datapackages using python3.7's dataclasses which provide convenient syntax highlighting and runtime assertions to catch errors early and easily introspect the C2M2 model with python docstrings. This package is available here and is easily updated to future C2M2 schemas.

Given that you have access to the *FAIR repository*, this can be installed directly from GitHub with:

```
pip3 install 'c2m2-frictionless-dataclass[full] @ git+https://github.com/nih-cfde/
 ↪c2m2-frictionless-dataclass'
```

This is also the time to install any relevant packages for interacting with your DCC's API, in our case we will use urllib to access the REST API.

Step 2: Setup an ETL script

Here we will outline the skeleton of an ETL script which utilizes the `c2m2-frictionless` package. For more complete examples for LINCS and other DCCs using this package, see the *FAIR repository*.

extract_transform.py:

```
#!/usr/bin/env python3

# Import C2M2 Frictionless Helper package
## dataclasses
import c2m2_frictionless
from c2m2_frictionless import C2M2

# TODO: Import other helper methods/setup DCC API
```

(continues on next page)

(continued from previous page)

```
def extract_transform():
  ''' Generate c2m2 dataclasses to be added to the datapackage
  '''
  # Construct the core id_namespace separating this DCC's ids from the ids of
  #  all other DCCs
  ns = 'http://www.lincsproject.org/'
  yield C2M2.id_namespace(
    id=ns,
    abbreviation='LINCS',
    name='Library of Integrated Network-Based Cellular Signatures',
    description='The Library of Integrated Network-Based Cellular Signatures (LINCS) Program aims to create a network-based understanding of biology by cataloging changes in gene expression and other cellular processes that occur when cells are exposed to a variety of perturbing agents.',
  )
  # TODO: yield additional resources, using `ns` for `id_namespace`

def extract_transform_validate(outdir):
  # use the extract_transform generator to produce a datapackage
  pkg = c2m2_frictionless.create_datapackage('C2M2', extract_transform(), outdir)
  # identify ontological terms in the datapackage, and complete the term tables
  c2m2_frictionless.build_term_tables(outdir)
  # validate assertion that the frictionless tableschema is complied with
  c2m2_frictionless.validate_datapackage(pkg)
  # validate assertion that (id_namespace, name) is unique in each resource
  c2m2_frictionless.validate_id_namespace_name_uniqueness(pkg)
  #
  return pkg

if __name__ == '__main__':
  import sys
  # grab an output directory off the command line
  extract_transform_validate(sys.argv[1] if len(sys.argv) >= 2 else '')
```

With the above skeleton in place, we can already run this script `python3 etl.py output` to produce and validate a C2M2 Level 1 datapackage, though this package will only contain one element: the `id_namespace`. As described in the comments, this should be used to make sure that the ids within your DCC will not clash with the ids in another, for more about this please refer to the C2M2 documentation.

Now we are ready to expand the `extract_transform` function such that we walk through the DCC's pertinent dataset descriptions and yield C2M2 equivalent dataclasses.

Step 3: Extracting and transforming your metadata for the C2M2

Step 3.1: Projects

We will first focus on grabbing the `project` structure. It is recommended that a primary project for which all other projects point to is used for your DCC for logical grouping in the UI. Please see the C2M2 documentation for a full description of what a "project" should represent, in short it is the groupings of datasets based essentially on funding awards.

```
def lincs_fetchdata_iter():
  ''' This method paginates through the LINCS API yielding LINCS datasets
  '''
  # ...

# ...
def extract_transform():
  # ...
  # Projects refer to collections of items that were produced i.e. under the same award
  #  for a more complete definition of this and other C2M2 constructs,
  #  please refer to the C2M2 docs (https://www.nih-cfde.org/product/cfde-c2m2/)
  primary_project = C2M2.project(
    # The namespace from before, ensuring any ids we choose don't clash
    id_namespace=ns,
    local_id='lincs',
    # the persistent id here enables you to point to a specific id,
    #  ideally a resolvable one, that behaves as a landing page for the project
    persistent_id='http://www.lincsproject.org/',
    # this abbreviation may be used in the display of your project
    abbreviation='LINCS',
    # this is the full name of your project
    name='Library of Integrated Network-based Cellular Signatures',
    # this is the full description of your project
    description='The Library of Integrated Network-Based Cellular Signatures (LINCS) Program aims to create a network-based understanding of biology by cataloging changes in gene expression and other cellular processes that occur when cells are exposed to a variety of perturbing agents.',
  )
  yield primary_project
  # ...
  projects = {}
  # ...
  for dataset in lincs_fetchdata_iter():
    # LINCS datasets each refer to a `project`
    #  LINCS phase 1, 2, MCF10A, and more, these also fit as C2M2 project
    #
    # We will register the project if not done so already ensuring we
    #  don't end up with duplicates
    if dataset['projectname'] not in projects:
      project = C2M2.project(
        # we'll use the same namespace of our primary project
        id_namespace=primary_project.id_namespace,
        # we'll use the project name as the identifier with lack of a better
        #  solution
        local_id=dataset['projectname'],
        name=dataset['projectname'],
        # we've omitted the optional persistent_id here because we currently
        #  don't have landing pages for these project names and as such
        #  cannot produce a persistent id referring to them. Ideally however
        #  we would use those urls.
      )
      # we'll save the project for de-duplication purposes
      projects[dataset['projectname']] = project
      # we'll yield it so it gets included in our datapackage
      yield project
      # we'll also create a project_in_project associating our primary project
```

(continues on next page)

(continued from previous page)

```
      #  to the project we just made
      yield C2M2.project_in_project(
        parent_project_id_namespace=primary_project.id_namespace,
        parent_project_local_id=primary_project.local_id,
        child_project_id_namespace=project.id_namespace,
        child_project_local_id=project.local_id,
      )
    else:
      # grab the already created project
      project = projects[dataset['projectname']]
    # ...
  # ...
```

Step 3.2: Collections & Files

With our project structure in place, we are now ready to walk through bringing the datasets and files into the package.

```
# ...
def extract_transform():
  # ...
  for dataset in lincs_fetchdata_iter():
    # ...
    # grab creation time as iso 8601
    creation_time = dataset.get('datereleased')
    if creation_time is not None:
      creation_time = datetime.datetime.strptime(
        creation_time, '%Y-%m-%d'
      ).replace(
        tzinfo=datetime.timezone.utc
      ).isoformat()
    # each LINCS dataset group has multiple datasets,
    #  each LINCS dataset has multiple files.
    # We can model this with two collections referring to the independent
    #  sets of files
    #
    # LINCS Dataset Group
    collection = C2M2.collection(
      id_namespace=ns,
      local_id=dataset['datasetgroup'],
      persistent_id=f"http://lincsportal.ccs.miami.edu/datasets/view/{dataset[
↪'datasetgroup']}",
      name=dataset['datasetname'],
      description=dataset.get('description', ''),
      creation_time=creation_time,
    )
    yield collection
    # we can associate our collection with our project
    yield C2M2.collection_defined_by_project(
      collection_id_namespace=collection.id_namespace,
      collection_local_id=collection.local_id,
      project_id_namespace=project.id_namespace,
      project_local_id=project.local_id,
    )
```

(continues on next page)

(continued from previous page)

```
    #
    # each dataset level (corresponding to a file) is specified in the metadata
    for dataset_id, dataset_version in zip(dataset.get('datasetlevels', []), dataset.
↪get('latestversions', [])):
      file = C2M2.file(
        id_namespace=ns,
        # the LINCS dataset id is used as the id
        local_id=dataset_id,
        # The file is necessarily associated with the project
        project_id_namespace=project.id_namespace,
        project_local_id=project.local_id,
        # The landing page of the dataset is used as the persistent id
        persistent_id=f"http://lincsportal.ccs.miami.edu/datasets/view/{dataset_id}",
        # the creation time we converted to ISO8601 before
        creation_time=creation_time,
        # File filename, note that C2M2 doesn't currently have a distinction between
        #  download and landing page, we opted to put the download url in the␣
↪filename.
        filename=f"http://lincsportal.ccs.miami.edu/dcic/api/download?path=LINCS_Data/
↪{datum['centername']}&amp;file={dataset_id}_{dataset_version}.tar.gz",
        # Following are additional optional fields that we were unable
        #  to easily map with LINCS but should be filled in if possible.
        # size_in_bytes=0,
        # sha256='',
        # md5='',
        # file_format='', # EDAM format:
        # data_type='',   # EDAM data:
      )
      yield file
      # each file is in the collection
      yield C2M2.file_in_collection(
        file_id_namespace=file.id_namespace,
        file_local_id=file.local_id,
        collection_id_namespace=collection.id_namespace,
        collection_local_id=collection.local_id,
      )
```

With this, we have already satisfied a valid C2M2 instance, but ideally we'll go a step further to C2M2 which has more annotations including information about Subjects and Biosamples, for more information refer to the C2M2 documentation.

Step 3.3: Subjects & Biosamples

The LINCS data associates Datasets with small molecules, cell lines, genes, proteins, antibodies, and other entities related to experimental conditions and readouts. A dataset consists of a series of experiments mostly carried out in high throughput with a goal of capturing changes in expression or other phenotypic changes that may be captured via images in various contexts under various conditions. For example, perturbed by small molecules in different cell lines.

Due to the sheer number of experiments performed in high throughput, LINCS stores the whole set of experiments in a few files representing the data level (level of processing performed on the raw data). It might be possible to extract these files and produce massive amounts of biosamples, but this information is not currently directly available on the website for the practical reason that having landing pages for each micro-experiment is not useful for the users of this data.

Because of this, LINCS lumps all the biosamples together into one "biosample" referring to the whole experiment. This allows annotating that biosample with the assay used to perform the experiment and associate that biosample with the

subjects of the experiment that were studied. Subjects includes cell line, small molecules, and etc. This closely matches how the LINCS portal serves this data and fits in the C2M2.

```
#
def lincs_fetchstats(datasetid, fields):
  # ...
#
# This dictionary describes how to get metadata for a given "stattype" (subject)
statstypes = {
  'cellline': type('StatType', tuple(), dict(
    # this describes the "type of subject" and is defined in the model
    id='cfde_subject_granularity:4',
    include=True,
    name='cell line',
    description='A cell line derived from one or more species or strains',
    # this is the landing page for this subject
    download_url=lambda datasetid: f"http://lincsportal.ccs.miami.edu/dcic/api/
↪Results?searchTerm=%22{datasetid}%22&category=cellline",
    # this is the id we should use for this subject
    id_from_meta=lambda meta: meta.get('CL_LINCS_ID'),
  )),
  # ...
}
# ...
def extract_transform():
  # ...
  # Each "stattype" corresponds to a "subject granularity" in the C2M2, we'll
  #  register those subject granularities now
  subject_granularities = {}
  # Add all subject granularities from `stattypes`
  for stat_key, stat in statstypes.items():
    subject_granularities[stat_key] = stat.id
  # ...
  subjects = {}
  # ...
  for dataset in lincs_fetchdata_iter():
    # ...
    # we'll associate a single biosample for the purpose of listing assay
    biosample_id = collection.local_id
    # NOTE: We defer biosample creation till after we have the anatomy
    # the biosample is in the collection
    #
    # a dataset_group has a number of aspects that are associated with it
    # including small molecules, proteins, cell lines, etc.. these are the subjects␣
↪of the
    # experiment. We will create all of these and associate them with our single
↪'super' biosample
    dataset_group_subjects = {}
    all_stats = lincs_fetchstats(datum['datasetid'], frozenset(datum.get('statsfields
↪', [])))
    for stat_name, stats in all_stats.items():
      for stat in stats:
        subject = None
        subject_id = statstypes[stat_name].id_from_meta(stat)
        if subject_id:
          if stat_name not in subjects:
            subjects[stat_name] = {}
          if subject_id not in subjects[stat_name]:
```

(continues on next page)

(continued from previous page)

```
            subject = C2M2.subject(
              id_namespace=ns,
              # in an effort to ensure id uniqueness, we'll use the stat_name as a␣
↪prefix
              local_id=f"{stat_name}:{subject_id}",
              # our subject must be a part of the project, but because these subjects
              #   are re-used across all projects, we'll associate them with
              #   the primary project instead of just the project
              project_id_namespace=primary_project.id_namespace,
              project_local_id=primary_project.local_id,
              # here we associate this subject with the subject granularity
              granularity=subject_granularities[stat_name],
            )
            yield subject
            subjects[stat_name][subject_id] = subject
          else:
            subject = subjects[stat_name][subject_id]
          #
        # if we were able to get a subject out of this we'll add it as an
        #  association to the biosample
        if subject and subject.local_id not in dataset_group_subjects:
          dataset_group_subjects[subject.local_id] = subject
          # the biosample was derived from this subject
          yield C2M2.biosample_from_subject(
            biosample_id_namespace=ns,
            biosample_local_id=biosample_id,
            subject_id_namespace=subject.id_namespace,
            subject_local_id=subject.local_id,
          )
          # the subject is in the collection (lincs dataset group)
          yield C2M2.subject_in_collection(
            subject_id_namespace=subject.id_namespace,
            subject_local_id=subject.local_id,
            collection_id_namespace=collection.id_namespace,
            collection_local_id=collection.local_id,
          )
    #
    biosample = C2M2.biosample(
      id_namespace=ns,
      local_id=biosample_id,
      project_id_namespace=project.id_namespace,
      project_local_id=project.local_id,
      # TODO: ontological mappings
      # assay_type='',
      # anatomy=''
    )
    yield biosample
    yield C2M2.biosample_in_collection(
      biosample_id_namespace=biosample.id_namespace,
      biosample_local_id=biosample_id,
      collection_id_namespace=collection.id_namespace,
      collection_local_id=collection.local_id,
    )
    # ...
    for dataset_id, dataset_version in zip(datum.get('datasetlevels', []), datum.get(
↪'latestversions', [])):
```

(continues on next page)

(continued from previous page)

```
      # ...
      # each file describes the biosample
      yield C2M2.file_describes_biosample(
        file_id_namespace=file.id_namespace,
        file_local_id=file.local_id,
        biosample_id_namespace=biosample.id_namespace,
        biosample_local_id=biosample.local_id,
      )
      # each file also describes all of the subjects in the study
      for subject in dataset_group_subjects.values():
        yield C2M2.file_describes_subject(
          file_id_namespace=file.id_namespace,
          file_local_id=file.local_id,
          subject_id_namespace=subject.id_namespace,
          subject_local_id=subject.local_id,
        )
    # ...
```

Step 3.4: Ontology mapping

Without subjects and biosamples in place, we can now add ontological terms which will be a point of harmonization between the different DCCs. For some, this will be simple, especially if the CFDE is prescribing an ontology that is already being used. For others, this may be a more complicated mapping effort. In LINCS, assays were described in a much more specific manner than the ontology for Biomedical Investigations (OBI). OBI is the currently prescribed ontology for assays in the C2M2–as such we need to collapse many of the assay descriptions into broader terms.

On the other hand, the organs were specified on the cell line in raw text, so these were mapped to the UBERON ontology.

```
# ...
lincs_assay_obi_lookup = {
  'Aggregated small molecule biochemical target activity': 'OBI:0000070',
  'ATAC-seq': 'OBI:0002020',
  'Bead-based immunoassay': 'OBI:0001632',
  'ELISA': 'OBI:0001632',
  'Fluorescence imaging': 'OBI:0001501',
  'KiNativ': 'OBI:0001146',
  'KINOMEscan': 'OBI:0001146',
  'L1000': 'OBI:0000424',
  'LC-MS': 'OBI:0000470',
  'Mass spectrometry': 'OBI:0000470',
  'MEMA (fluorescence imaging)': 'OBI:0001501',
  'Microwestern': 'OBI:0000615',
  'P100 (mass spectrometry)': 'OBI:0000615',
  'PCR': 'OBI:0001146',
  'RNA-seq': 'OBI:0001271',
  'RPPA': 'OBI:0000615',
  'SWATH-MS': 'OBI:0000615',
  'Tandem Mass Tag (TMT) Mass Spectrometry': 'OBI:0000470',
  'To Be Classified': '',
}

lincs_anatomy_lookup = {
  'Ovary': 'UBERON:0000992',
  'cervix': 'UBERON:0000002',
```

(continues on next page)

(continued from previous page)

```
  'kidney': 'UBERON:0002113',
  'vulva': 'UBERON:0000997',
  'gall bladder': 'UBERON:0002110',
  'urinary bladder': 'UBERON:0001255',
  'nervous system': 'UBERON:0001016',
  'Lung': 'UBERON:0002048',
  'skin': 'UBERON:0002097',
  'Intestine': 'UBERON:0000160',
  'Peripheral blood': 'UBERON:0000178', # WARNING: not 'peripheral'
  'miscellaneous': '',
  'adrenal gland': 'UBERON:0002369',
  'lung': 'UBERON:0002048',
  'thyroid': 'UBERON:0002046',
  'urinary tract': 'UBERON:0011143, UBERON:0001556',
  'uterus': 'UBERON:0000995',
  'pancreas': 'UBERON:0001264',
  'lymphatic system': 'UBERON:0006558',
  'muscle': 'UBERON:0001630',
  'biliary tract': '',
  'blood': 'UBERON:0000178',
  'stomach': 'UBERON:0000945',
  'intestine': 'UBERON:0000160',
  'head and neck': 'UBERON:0000974', #'UBERON:0000974, UBERON:0000974',
  'human': '',
  'Brain': 'UBERON:0000955',
  'testes': 'UBERON:0000473',
  'bone': 'UBERON:0002481',
  'large intestine': 'UBERON:0000059',
  'endometrium': 'UBERON:0001295',
  'ureter': 'UBERON:0000056',
  'pleura': 'UBERON:0000977',
  'Skin': 'UBERON:0002097',
  'breast': 'UBERON:0000310',
  'liver': 'UBERON:0002107',
  'Mammary gland': 'UBERON:0001911',
  'ovary': 'UBERON:0000992',
  'prostate': 'UBERON:0002367',
  'esophagus': 'UBERON:0001043',
  'brain': 'UBERON:0000955',
}

lincs_organism_lookup = {
  'Homo sapiens': type('NCBI:taxid9606', tuple(), dict(
    id='NCBI:taxid9606',
    clade='species',
    name='Homo sapiens',
  )),
  'Homo Sapiens': type('NCBI:taxid9606', tuple(), dict(
    id='NCBI:taxid9606',
    clade='species',
    name='Homo sapiens',
  )),
  'Homo Sapien': type('NCBI:taxid9606', tuple(), dict(
    id='NCBI:taxid9606',
    clade='species',
    name='Homo sapiens',
```

(continues on next page)

(continued from previous page)

```
  )),
}
# ...
def first_or_none(it):
  ''' A simple helper for safely returning the first element of a potentially empty␣
 ↪iterator
  '''
  try:
    return next(iter(it))
  except StopIteration:
    return None
# ...
def extract_transform():
  # ...
  taxonomies = {}
  # https://osf.io/gpf3d/
  cfde_subject_role_organism = C2M2.subject_role(
    id='cfde_subject_role:0',
    name='single organism',
    description="The organism represented by a subject in the 'single organism'␣
 ↪granularity category",
  )
  yield cfde_subject_role_organism
  # ...
  for dataset in lincs_fetchdata_iter():
    # ...
    anatomies = set()
    #
    for stat_name, stats in all_stats.items():
      # ...
      if subject:
        # ...
        # In order to capture anatomy at the biosample level, we need to catch
        #  the anatomy on the subject
        if stat_name == 'cellline':
          # in the case of cell lines, we'd have a taxonomy
          lincs_taxon = lincs_organism_lookup.get(stats.get('organism'))
          if lincs_taxon and lincs_taxon.id not in taxonomies:
            taxon = C2M2.ncbi_taxonomy(
              id=lincs_taxon.id,
              clade=lincs_taxon.clade,
              name=lincs_taxon.name,
            )
            taxonomies[lincs_taxon.id] = taxon
            yield taxon
          elif lincs_taxon:
            taxon = taxonomies[lincs_taxon.id]
          else:
            taxon = None
          #
          if taxon:
            # associate the subject with the taxon
            yield C2M2.subject_role_taxonomy(
              subject_id_namespace=subject.id_namespace,
              subject_id=subject.id,
              role_id=subject_role_organism.id,
```

(continues on next page)

(continued from previous page)

```
            taxonomy_id=taxon.id,
          )
        # in the case of cell lines, we'd have an anatomy
        if subject_id in lincs_anatomy_lookup:
          anatomies.add(lincs_cellline_anatomy(lincs_anatomy_lookup[subject_id]))
        else:
          print(f"WARNING: {subject_id} not in celllines")
  # ...
  biosample = C2M2.biosample(
    id_namespace=ns,
    local_id=biosample_id,
    project_id_namespace=project.id_namespace,
    project=project.id,
    # Lookup assay name to get OBI
    assay_type=lincs_assay_obi_lookup.get(dataset.get('assayname'), ''),
    # NOTE: we have several anatomies at this point and are forced to choose one
    # this will require improvements to either the model or to our conversion
    # perhaps we could instead have several biosamples for each file corresponding
    # to each subject. This will be left for future directions
    anatomy=first_or_none(anatomies),
  )
```

Here we show how we were able to capture an anatomy, taxonomies on cell lines, and assays on biosamples. While not complete nor perfect, this provides the much needed interoperation layer with the other DCCs. With this type of annotation, we can look at files across different DCCs by assay, anatomy, or subjects by taxonomy. It improves the findability of these files even if the model does not perfectly capture the real structure of the original page. The CFDE is meant to catalog and provide federated search to all of the DCCs, directing users to the DCC's pages, not to replace the catalogs already in place.

Step 4: Running and validating our extraction and transformation

Our script is now ready to be run, tested, and refactored. During this phase it is ideal to cache network requests if hitting API endpoints. Every run of the script will re-construct the entire data package and run through the various checks, reporting any errors that occur.

```
python3 extract_transform.py output
```

Step 5: Pushing our validated C2M2 datapackage

A client for facilitating the ingestion of your data into the CFDE portal exists. For more information about this, check here. Once installed, the output directory we produced with the `etl.py` script can be sent to DERIVA.

```
# install the DERIVA loader script
pip install cfde-submit

# output here is the directory with the output of the etl.py script
cfde run output

# check the status of the ingestion
cfde status
```

This tool facilitates authentication, manual preview verification when loaded on the server side (you will get an email), followed by CFDE verification and ultimate incorporation of your data into the unified portal.

4.6.4 Conclusion

Taking advantage of this tooling will simplify the process of ETL script development given that most things will be caught with dataclass assertions and datapackage validation. Nonetheless, this enforces minimal compliance with the C2M2 standard. For maximal compliance, it is essential that you review the most up to date C2M2 documentation and it may also be useful to review and perform *FAIR assessments*, which includes more elaborate assertions for compliance with FAIRness beyond the C2M2.

4.6.5 Video Tutorials

The LINCS DCC has additionally created video tutorials to show how the LINCS L1000 dataset can be converted to C2M2 format. The tutorials have been adapted from the demo presented at the April 5th, 2022 CFDE Cross-Pollination Meeting. These videos use the most recent release of the LINCS L1000 signatures as example data, and are intended to complement the protocol above as a separate demonstration of how a different dataset may be converted to the C2M2 model.

Introduction - This video briefly introduces the C2M2 model and the LINCS program, as well as provides an overview of the LINCS L1000 data.

Consolidating Metadata - This video discusses how the LINCS L1000 dataset was processed, including how ontological mappings were generated.

Building a Datapackage - This video walks through the steps of building each type of C2M2-compliant metadata table using the processed LINCS L1000 data.

Code Demo - Jupyter Notebook containing demo code from video tutorials, showing how to set up the C2M2 metadata tables for LINCS L1000 data.

4.6.6 Reference

FAIR Repo

The CFDE FAIR repository is currently private given that it contains details about DCCs that have not yet been verified. Please submit a request to https://www.nih-cfde.org/contact/ if you need access to the repository.

If you have access to the repository, you can access it here. It contains C2M2 conversion scripts much like the one described here for several DCC's publicly facing metadata, organized into directories with each DCC name.

CHAPTER

FIVE

THE NEED FOR IDENTIFIERS

Authors: Philippe Rocca-Serra

Maintainers: Philippe Rocca-Serra

Version: 0.1

5.1 Objectives

The aim of the recipe is to provide an introduction to the notion of identifiers, the different kind of identifiers and the kind of infrastructure required to support their use in the context of CFDE and in the context of producing and consuming FAIR data.

5.2 Definitions

identifier: An identifier is a chain of characters (number, letter, symbol, or any combination of those) which is associated to an entity or a type of entities and which is used to refer to those entities in an indexing system.

unique identifier: A unique identifier (UID) is an identifier which refers to only one instance of thing

locally unique identifier: A locally unique identifier (LUID) is an identifier which is unique to a given context (hence, local) but which may clash if moved out of this specific context. By clash, we mean that if dereferenced in the wrong context, it could return an instance different from the one it pointed to when it was minted in its original context.

globally unique identifier: A globally unique identifier (GUID) is an identifier which is produced such that the probability of the exact same string is extremely low (but not null), hence global. Also known as `universally unique identifier` (UUID). Their production does not depend on central registration and relies on the generation of a 128-bit string.

```
import uuid
id = uuid.uuid4()
print(id)
5b6d0be2-d47f-11e8-9f9d-ccaf789d94a0
```

persistent identifier: A persistent identifier (PID) is an identifier which provides a long-lasting reference to a digital resource. The key notion introduced is that of`persistence`, which can only exist if a `resolution service` exists. Well known examples are `persistent identifiers` are `Digital Object Identifiers (DOI)`,

Archive Retrieval Keys (ARK), Open Researcher and Contributor ID (ORCID) or Persistent Uniform Resource Locator (PURL). More examples can be found on the PID forum.

compact identifier: A compact identifier is a unique string consisting of a Prefix (assigned by curator), a colon (':'), and an Accession (e.g. local identifier string). The prefix is composed of an optional Provider Code, and an assigned Namespace, separated by a slash ('/').

identifier minting service: An identifier minting service is a piece of software infrastructure which allows the production of an identifier. It can be as simple as as UUID generator (see above) or random number generator.

identifier resolution service: An identifier resolution service is a piece of software infrastructure which given an identifier can return a web resource about the entity designated by the identifier. Such service is essential to realize the notion of persistence of identifiers.

5.3 The Importance of PIDs in FAIR and in the Context of CFDE

Data interoperability as described by FAIR assumes that entities and information about them can be retrieved throughinternet communication protocol. This can thus only be realized by relying on PIDs and resolution service to obtain the resources of interest.

For a number of digital objects (scholarly articles, grants), persistent identifiers such DOI exist and for research organizations and persons, RORID and ORCID are gaining popularity becoming more widely used.

Datasets are another type of digital objects which are being identified by DOI with services such as Figshare or Zenodo.

But interoperability goes far beyond reliable identification of these entities. For research scientists, entities such as:

- Taxonomic Information
- Anatomical Information
- Diseases and Conditions
- Molecular Entities, such as:
  - Chemicals
  - Metabolites
  - Genes
  - Transcripts
  - Protein
  - Lipids
- Pathways and Biochemical reactions
- Assays
- Instruments
- Licenses

For all these entities, the **NIH CFDE** has identified a set of semantics artefacts/ontologies, the classes of which are all identified by PIDs and are used to annotated the datasets which will be represented in the C2M2 model and made available through the DERIVA system.

- This specific *NIH-CFDE recipe* details which ontologies have been selected and how their use will be rolled out as the program progresses.

It is important to notice that for such resources, a central authority is responsible for issuing (minting) those identifiers as well as providing the landing pages for the different content types associated with these kind of PIDs. This works well to address the semantic markup needs that arise in the data curation, extraction, transformation, and load processes ETL. However, there is a need to be able to create resolvable identifiers on demands. For instance, to uniquely identify data files. In order to discussion this specific use case, a specific document is available and details the use of minids.

- Refer to the specific recipe on *MINIDS* to learn how to mint such resolvable identifiers for your resources.

5.4 Bibliographic References

1. Klump, J. and Huber, R., 2017. 20 Years of Persistent Identifiers – Which Systems are Here to Stay?. Data Science Journal, 16, p.9. http://doi.org/10.5334/dsj-2017-009
2. McMurry JA, Juty N, Blomberg N, et al. Identifiers for the 21st century: How to design, provision, and reuse persistent identifiers to maximize utility and impact of life science data. PLoS Biol. 2017;15(6):e2001414. Published 2017 Jun 29. http://doi.org/10.1371/journal.pbio.2001414
3. Martin Fenner, Mercè Crosas, Jeffrey S Grethe, David Kennedy, Henning Herm-jakob, Philippe Rocca-Serra, Gustavo Durand, Robin Berjon, Sebastian Karcher,Maryann Martone, and Tim Clark. 2019. A data citation roadmap for scholarly data repositories. Scientific Data6, 1 (2019), 1–9.http://doi.org/10.1038/s41597-019-0031-8 4.https://www.pidforum.org/
4. John Kunze. The Entity (N2T) Resolver: low-risk, low-cost persistent identifica-tion.https://hdl.handle.net/1813/3688
5. Wimalaratne, S., Juty, N., Kunze, J. et al. Uniform resolution of compact identifiers for biomedical data. Sci Data 5, 180029 (2018). https://doi.org/10.1038/sdata.2018.29
6. Research Organization Registry

5.5 Recommendations for Minting Persistent and Resolvable Identifiers

Authors: Rick Wagner Robert Carter Philippe Rocca-Serra

Maintainers: Rick Wagner

Version: 0.1

5.5.1 Objectives

The main objective of this recipe is to provide a set guidelines and recommendations for producing or using resolvable identifiers. This is a key element as it provides `interoperability` and 'reusability', the I and R in FAIR.

5.5.2 Introduction

A `persistent identifier` is a **globally unique name for a resource within the CFDE**. This name can be used to retrieve a description of the resource or to identify an entity like a file or dataset without dereferencing the identifier (i.e., accessing the resource itself). Persistent identifiers are created and maintained by the DCC responsible for the data and must be defined using the characteristics described in this document to ensure their data is `Findable`, `Accessible`, `Interoperable`, and `Reusable` (FAIR) [1]. Because DCCs need to be able to define and choose different identifiers for additional use cases, this document provides a minimum set of requirements that are intended to be flexible and fit within the existing practices of data repositories, publishers, and libraries [2, 3].

As part of the Crosscut Metadata Model (C2M2), persistent identifiers can enable several use cases:

- Unambiguously referring to a resource
- Estimating the amount of data stored by a DCC
- Accessing descriptions of named resources
- Verifying that a file is the one referenced by a persistent identifier
- Determining if two identifiers refer to the same file or files
- Citing data

At present, the scope of these requirements is limited to two entities within the C2M2, files and collection (a grouping of files). These requirements may be extended to support other use cases, in particular data citation. Some of the information provided by the persistent identifier may be duplicated within the C2M2; over time, the C2M2 itself may be built in part from sets of persistent identifiers. The CFDE's Persistent Identifier Recommendations promotes the following recommendation towards improving the FAIRness of Common Fund sponsored data:

- All data identifiers should be expressed as a URL
- Identifier URLs should resolve to landing pages with human-readable metadata
- Data metadata should be embedded in the landing page by using JSON-LD and Schema.org [4]
- Metadata should be available via content negotiation at the same URL
- All data, even temporary or intermediate data, should be associated with an identifier
- Different types of identifiers can be assigned to the same data
- Identifiers must capture data checksums to enable verification of data integrity
- Identifier creation and resolution must become part of the research data lifecycle and, where possible, be integrated with tools and applications that can automate this process

1. Required Identifier Characteristics

All persistent identifiers used within the CFDE **must** meet the following requirements regarding their uniqueness, format, resolution, and description.

- **`Uniqueness`**: Persistent identifiers must be unique within the CFDE and only refer to single file or collection. Multiple identifiers (of the same or different types) may refer to the same file or collection. Identifiers should provide a means (preferably checksums) to determine if multiple identifiers refer to the same resource.
- **`Format`**: Persistent identifiers must comply with the IETF standard for URIs, RFC 3986. This may be either be as an HTTPS URL or a compact URI. The use of HTTPS URLs is strongly preferred.

Examples:

> **HTTPS URL:** https://doi.org/10.25490/a97f-egyk
>
> **Compact URI:** doi:10.25490/a97f-egyk

- **`Resolution`**: Persistent identifiers using compact URIs must use prefixes (also known as URI schemes) registered with N2T or Identifiers.org [5]. This permits the resolution of compact URIs in a consistent manner and helps to ensure uniqueness of identifiers by defining identifier naming authorities. Instructions on how to register prefixes are on the N2T and Identifiers.org sites. The joint list of registered prefixes is available here.

Examples:

N2T: https://n2t.net/doi:10.25490/a97f-egyk

Identifiers.org: https://identifiers.org/doi:10.25490/a97f-egyk

- **`Description`**: When an identifier is resolved, either as a HTTPS URL or from a compact URI to a URL, the minimum result must be an HTML landing page, with human-readable metadata describing the referenced object.

2. Recommended Identifier Characteristics

Beyond these minimal requirements, there are several recommendations that contribute to the CFDE use cases and can improve the reuse and citation of the data referred to by persistent identifiers. Many of these recommendations are drawn from the ref data citation roadmap. They primarily focus on the descriptive metadata provided by the persistent identifier by its landing page, especially in a machine- readable format.

- **`Machine Readability`**: In addition to the HTML landing page, intended to be read by a person using a browser, the identifier and metadata associated with the resource should be embedded in the landing page in JSON-LD. The metadata should also be retrievable as JSON-LD using the `Accept` header to allow clients to retrieve the metadata without unnecessarily parsing HTML.
- **`Metadata Schema`**: The CFDE use cases for **file** and **collection metadata** include validation, disambiguation, and metrics around storage utilization and locality. For example, Google vs. Amazon. Later, the use of the location attribute for the file identifiers with a list of URIs for data protocols may allow for data access. The recommended metadata for collections also includes the elements to allow for data citation, primarily to reduce the amount metadata required for individual files.

DCCs may choose to use different identifiers for files and collections based on their own needs. For example, a DCC may choose to use DOIs for their collections and ARKs for their files. And because resources may be referenced by multiple persistent identifiers, some files may also be referenced by DOIs so that the files may be cited as a citable dataset [6].

Specifics about identifiers for Files

File identifier metadata should include the following:

- **`identifier`**: The persistent identifier.
- **`filename`**: The name of the file without addition path information.
- **`size`**: The size of the file in bytes.
- **`checksum`**: The output of a cryptographic hash function run on the file, preferably SHA256, which can be used to verify data integrity and identify files regardless of a data repository's naming conventions.
- **`checksum_algorithm`**: The checksum algorithm used, one of md5, sha256, or sha512.
- **`location`**: One or more locations using RFC 7595 schemes (e.g., HTTPS, SCP, FTP) or other agreed upon conventions (S3, GS) that map to access protocols. Collections Collection identifier metadata should include the following
- **`identifier`**: The persistent identifier.
- **`name`**: The human-readable name of the collection.
- **`author`**: The person(s) publishing the collection.

- **publisher**: The organization or entity publishing the collection.
- **datePublished**: The date published.
- **version**: Version of the collection.
- **type**: Must be dataset.
- **filename**: The name of a manifest file without addition path information. The manifest file should contain a list of the persistent identifiers for the individual files in the collection.
- **size**: The size of the manifest file in bytes.
- **checksum**: The output of a cryptographic hash function run on the manifest file, preferably SHA256, which can be used to verify data integrity and identify files regardless of a data repository's naming conventions.
- **checksum_algorithm**: The checksum algorithm used, one of md5, sha256, or sha512.
- **location**: One or more locations using RFC 7595 schemes (e.g., HTTPS, SCP, FTP) or other agreed upon conventions (S3, GS) that map to access protocols. CFDE Resources The CFDE Coordinating Center is evaluating the operation a persistent identifier service for files and collections based on different identifier schemes including Handles, DOIs, and ARKs. DCCs interested in creating persistent identifiers for files and collections using this service may contact the CFDE CC for further details.

5.5.3 Conclusion

The section draws the attention to essential properties identifiers must have in order to deliver `interoperability`. These will be implemented for all key objects and some of their key qualifiers.

5.5.4 References

- [1] Wilkinson MD, Dumontier M, Aalbersberg IjJ, Appleton G, Axton M, Baak A, Blomberg N, Boiten J-W, da Silva Santos LB, Bourne PE, Bouwman J, Brookes AJ, Clark T, Crosas M, Dillo I, Dumon O, Edmunds S, Evelo CT, Finkers R, Gonzalez-Beltran A, Gray AJG, Groth P, Goble C, Grethe JS, Heringa J, 't Hoen PA., Hooft R, Kuhn T, Kok R, Kok J, Lusher SJ, Martone ME, Mons A, Packer AL, Persson B, Rocca-Serra P, Roos M, van Schaik R, Sansone S-A, Schultes E, Sengstag T, Slater T, Strawn G, Swertz MA, Thompson M, van der Lei J, van Mulligen E, Velterop J, Waagmeester A, Wittenburg P, Wolstencroft K, Zhao J, Mons B. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data [Internet]. Springer Science and Business Media LLC; 2016 Mar 15;3(1). Available from: http://dx.doi.org/10.1038/sdata.2016.18
- [2] Data Citation Synthesis Group. Joint Declaration of Data Citation Principles [Internet]. Force11; 2014. Available from: https://www.force11.org/group/joint-declaration-data-citation-principles-final
- [3] Fenner M, Crosas M, Grethe JS, Kennedy D, Hermjakob H, Rocca-Serra P, Durand G, Berjon R, Karcher S, Martone M, Clark T. A data citation roadmap for scholarly data repositories. Scientific Data [Internet]. Springer Science and Business Media LLC; 2019 Apr 10;6(1). Available from: http://dx.doi.org/10.1038/s41597-019-0031-8
- [4] Guha RV, Brickley D, Macbeth S. Schema.org. Communications of the ACM [Internet]. Association for Computing Machinery (ACM); 2016 Jan 25;59(2):44–51. Available from: http://dx.doi.org/10.1145/2844544
- [5] Wimalaratne SM, Juty N, Kunze J, Janée G, McMurry JA, Beard N, Jimenez R, Grethe JS, Hermjakob H, Martone ME, Clark T. Uniform resolution of compact identifiers for biomedical data. Scientific Data [Internet]. Springer Science and Business Media LLC; 2018 May 8;5(1). Available from: http://dx.doi.org/10.1038/sdata.2018.29

- [6] DataCite Metadata Working Group. DataCite Metadata Schema Documentation for the Publication and Citation of Research Data v4.3. DataCite [Internet]. DataCite; 2019; Available from: https://schema.datacite.org/meta/kernel-4.3/

5.6 Identifier Minting Service with MINID Client

Authors:

- Philippe Rocca-Serra
- Mike D'Arcy

Maintainers: Philippe Rocca-Serra

Version: 1.0

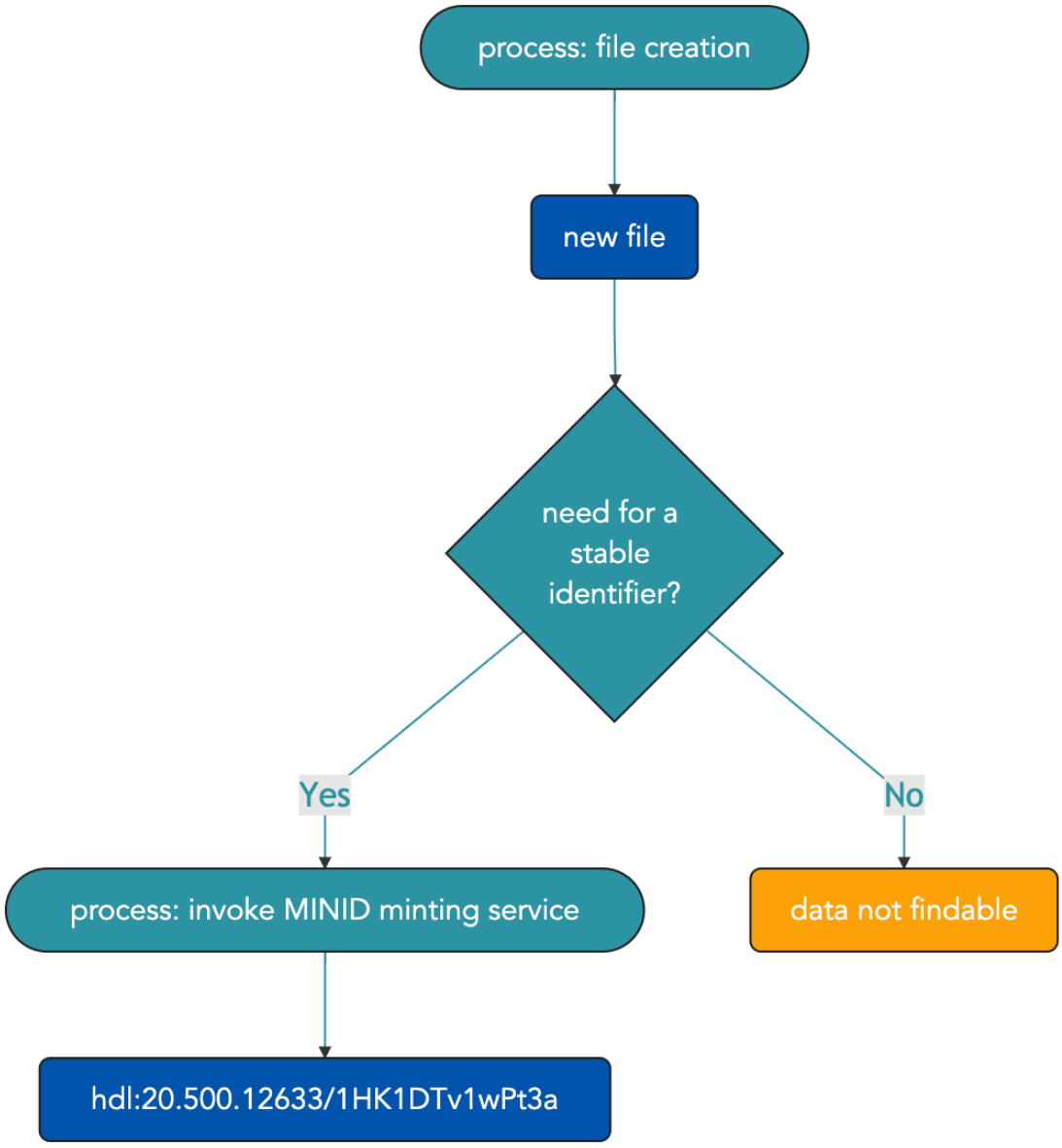
5.6.1 Table of Contents

5.6.2 Main Objectives

The main purpose of this recipe is:

To create a **persistent**, **globally unique** and **resolvable identifier** using the ***Minid client*** accessing the Minid 2.0 release.

5.6.3 Graphical Overview of the FAIRification Recipe Objectives



5.6.4 Installing the minid 2.0 client

This is a prerequisite to be able to call the minid API hosted on a server at the following url http://minid.bd2k.org/

installing with pip

```
pip3 install --pre minid
```

building from source

use the dev branch to obtain to source minid github repository

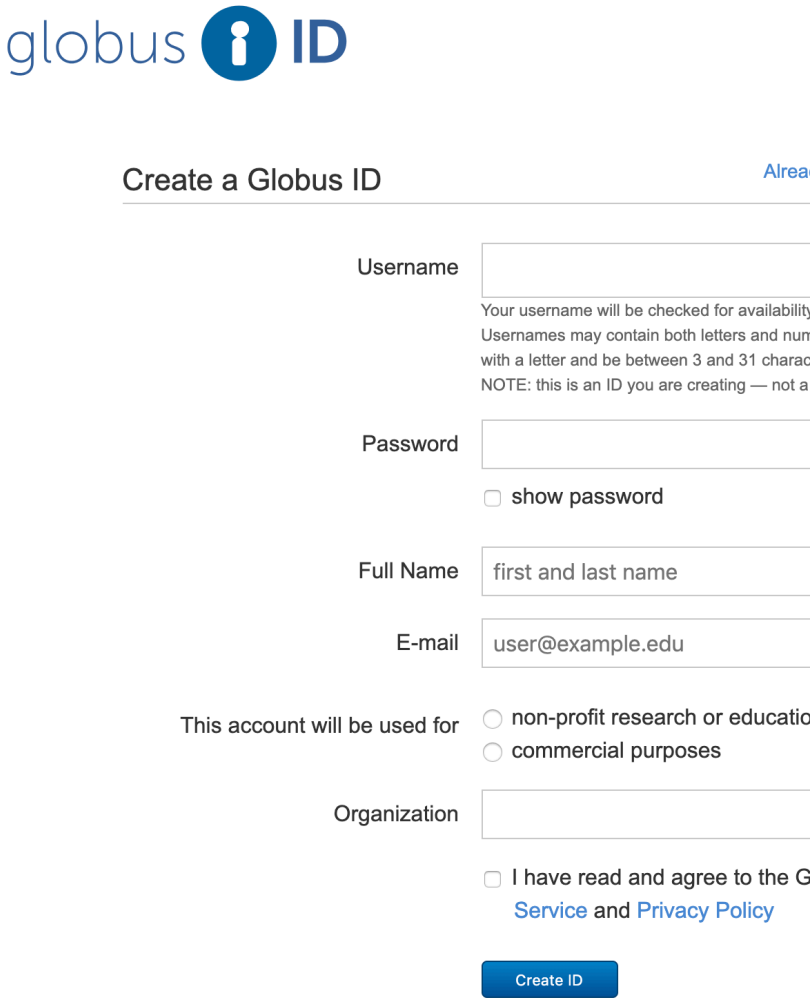
5.6.5 Configuration

1. Prerequisite: create a minig-config.cfg file

As a convenience you need specify this information in a minid configuration file (`~/.minid/minid-config.cfg`) To do so from the command line, issue the following:

```
$ mkdir ~/Users/philippe/.minid
$ cd .minid
$ touch minid-config.cfg
```

1. Create a GlobusID account



Before using the API you first need to create a globus account

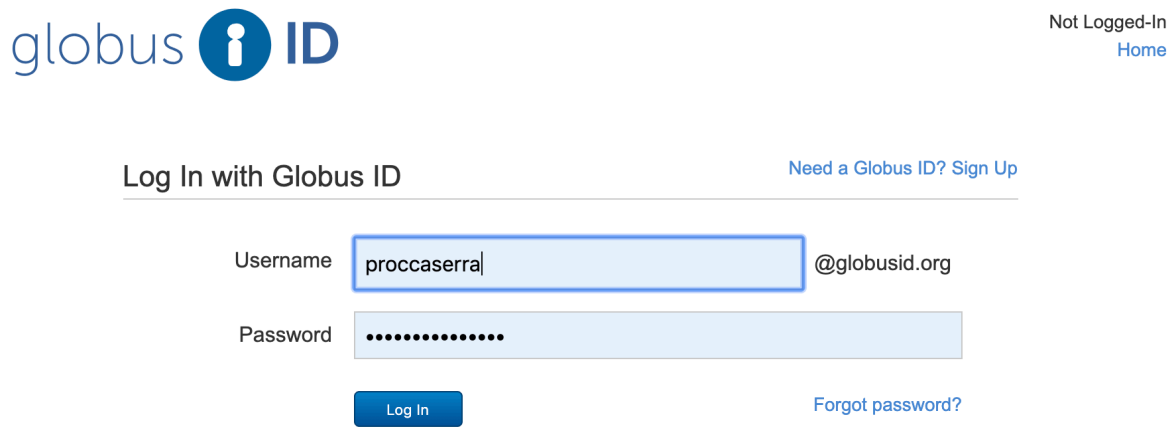
and validate your email address, as part of the registration process. A unique code will be sent to your email address. You must present this code along with your email address when accessing the API.

1. Accessing minid service from the command line

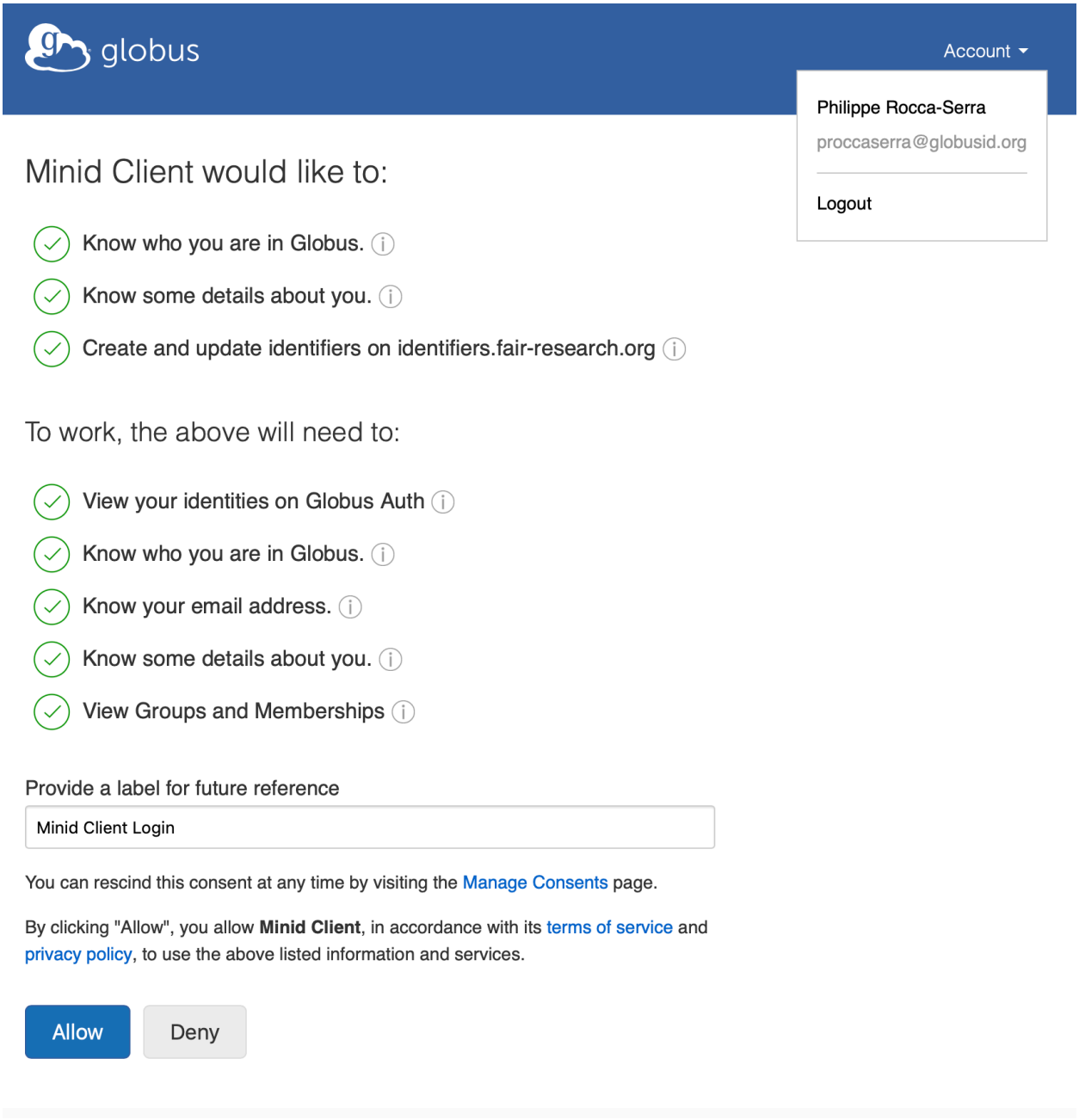
With the completion of the previous steps, you are now ready to use the minid service. The first thing to do is to invoke to `minid login` command

```
$ minid login
```

This will open the GlobusID login page. Simply enter your credentials obtained from 2.


globus ID
Not Logged-In
Home
Log In with Globus ID
Need a Globus ID? Sign Up
Username
proccaserra
@globusid.org
Password
Log In
Forgot password?

followed by:



If all goes well, the following browser screen will be shown:

Minid Client

Login Successful. You may close this tab.

While the terminal will show the following:

```
You have been logged in.
```

This means you are now ready to use the minid service from the command line.

5.6.6 Usage

The CLI supports the following simple operations (Note: the `--test` flag creates names in a test namespace that is removed periodically; remove that flag to create production minids.):

- Check a known minid identifier

```
$ minid check hdl:20.500.12633/1HK1DTv1wPt3a
```

if everything is setup correctly, the command will return:

```
Minid:               hdl:20.500.12633/1HK1DTv1wPt3a
Title:
Checksums:           e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855␣
 ↪(sha256)
Created:
Landing Page:        https://identifiers.fair-research.org/hdl:20.500.12633/
 ↪1HK1DTv1wPt3a
EZID Landing Page:   https://ezid.cdlib.org/id/hdl:20.500.12633/1HK1DTv1wPt3a
Locations:           http://example.com/foo.txt
```

- Create a new identifier (the `--location` option, if provided, must be at the end).

```
$ minid --register [--title <title>] <file_name> [--locations <loc1>..<locN>]
```

- Update metadata about an identifier:

```
$ minid --update [--title <title>] [--status <status>] [--obsoleted_by <minid>] [--
 ↪locations <loc1> <loc2>] <identifier>
```

- View all minid options:

```
$ minid -h
```

Landing pages are accessible via the minid website: http://minid.bd2k.org/minid/landingpage/<identifier>.

File Manifest Format

Minids can only be assigned to a single file. In order to assign a minid to a collection of files, we recommend using a `BDBag <https://github.com/ini-bdds/bdbag>_` or the minid file manifest format.

The minid file manifest format is a JSON-based format that enumerates a list of files as JSON objects that have the following attributes:

- Length: The length of the file in bytes.
- Filename: The filename (or path) relative to the manifest file.
- One or more (only one of each) of the following `algorithm:checksum` key-value pairs:
  - md5:<md5 hex value>
  - sha256:<sha256 hex value>
  - sha512:<sha512 hex value>
- URL: The URL to the file.

The manifest may be used to create a minid for a collection of files or alternatively as input to the minid batch-register command.

Below is a sample file manifest configuration file:

```
 [
     {
         "length":321,
         "filename":"file1.txt",
         "md5":"5bbf5a52328e7439ae6e719dfe712200",
         "sha256":"2c8b08da5ce60398e1f19af0e5dccc744df274b826abe585eaba68c525434806",
         "url" : "globus://ddb59aef-6d04-11e5-ba46-22000b92c6ec/share/godata/file1.
↪txt"
     },
     {
         "length": 632860,
         "filename": "minid_v0.1_Nov_2015.pdf",
         "sha256": "cacc1abf711425d3c554277a5989df269cefaa906d27f1aaa72205d30224ed5f
↪",
         "url" : "http://bd2k.ini.usc.edu/assets/all-hands-meeting/minid_v0.1_Nov_
↪2015.pdf"
     }
 ]
```

5.6.7 Conclusions

Using the `Minid` service, resources can now generate stable, resolvable identifiers for their digitial documents. The `Minid` service thus provides a key component to enable `interoperability` and `reusability` by ensuring digital assets get be looked up using a standard protocol (HTTP request). The service also supports data integrity checks thanks to the native support of checksumming functions, with sha256 being recommended.

5.6.8 References:

1. Madduri R, Chard K, D'Arcy M, Jung SC, Rodriguez A, Sulakhe D, et al. (2019) Reproducible big data science: A case study in continuous FAIRness. PLoS ONE 14(4): e0213013. https://doi.org/10.1371/journal.pone.0213013
2. https://minid.readthedocs.io/en/develop/identifiers.html#minids-vs-handles

CHAPTER

SIX

SEMANTICS

This section covers aspects of `interoperability` related to the use of controlled terminologies to ensure consistent annotation to increase query recalls, disambiguate free text description. It includes:

- A chapter dedicated to explaining what are semantic resources, their typology, their applications and potentials.
- A chapter detailing existing tools as well as services supported hosting, serving, and using controlled terminologies.
- A chapter highlighting the terminology requirements of the C2M2 model and place those in the context of the current practices of terminology use by a number of DCCs.

6.1 Controlled Terminologies & Ontologies

Authors: Philippe Rocca-Serra

Maintainers: Philippe Rocca-Serra

Version: 0.1

6.1.1 Objectives

The aim of this recipe is to provide a compact introduction about `controlled terminologies` and `ontologies`, why these resources are central to the preservation of knowledge and data mining and how such resources are developed.

6.1.2 Controlled terminology or Ontology, what's the difference ?

The need for `controlled vocabulary` often arises in situation where validation of textual information is necessary for operational requirements. The main initial driver for data entry harmonization is to increase query recall. It is most basic form, `keywords` may be used to perform indexation. However, if relying on user input alone, the chances of typographic errors increases with the number of users. These unavoidable events accumulate over time and end up hurting the accuracy of search results and this is the reason for offering sets of predefined values. It reduces the noise. However, this can come at the cost of precision, as the predefined terms may not cover the exact thing users may need to describe. Furthermore, term mis-selection by user is not eliminated and introduces another type of error.

A `controlled terminology` is a *`normative`* collection of terms, the spelling of which is fixed and for which additional information may be provided such as `definition`, a set of `synonyms`, a `editor`, a `version` as well as a `license` determining the condition of use. The set of information about a specific controlled terminology term

is designated as term metadata. In a controlled terminology, terms appear as a flat list, meaning that no relationship between any of the entities the controlled terminology represents is captured in any formal way. This is the main drawback and limitation of controlled terminologies, which are often developed to support a data model or an application.

Anontology on the other hand, is a formal representation of a domain knowledge where concepts are organized hierarchically. The qualifierformal refers to a set of axioms and rules based on logic (e.g. first order logic) to structure, organize and check the consistency of the term hierarchy. As one can sense right away, ontologies are often more sophisticated artefact, supported by a more advanced theorical frameworks and and dedicated tools to develp of them (e.g. Protégé, TopBraid Composer, OBO foundry INCAtools or Robot tool)

6.1.3 How are they built and maintained and why does it matter?

In order to improve over simple controlled terminologies, a huge area of research has developed to provide tools and frameworks supporting the representations of relationships between entities. The field is known as formal semantics in knowledge representation circles. One of most immediately available example of entity relationships and their potential for improving searches is the is_a relationship, which aims to cover the Parent / Child relationship that holds between 2 entities. For instance:

```
-Vertebrate
--mammal
---dolphin
--bird
---pigeon
```

In this representation, classes are *directly* asserted (placed) under a parent class if and only if the rule new class is a child of the parent Class. 'Orchids', which in this hierarchy.

While working on small structured vocabularies, it is still possible to detect potential errors, but this approach does not scale to support real life semantic artefacts which support complex biological and biomedical information systems. Languages (RDF,SKOS,OWL) exist to provide the expressivity required to axiomatize relations between entities. In turn, building on these formal rules and associated proofers, automatic classifiers known as reasoner can inspect semantics artefacts to detect inconsistencies and to suggest parent classes. This is a step known as 'inference' where new knowledge is produced by the software agent rather than direct assertion by humans. This provide a significant support, even if far from supporting all the subtleties of actual knowledge.

There are 6 important features to consider where selecting an semantic artefact for making FAIR datasets:

1. What license and terms of use does it mandate?

2. What format does it come in?

3. Is it well maintained ? i.e. frequent release, term requests handling, versioning and deprecation policies clarified.

4. Are there stable persistent resolvable identifiers for all terms?

5. Who use it and What resources are being annotated with it?

6. Is it well documented? There should be enough metadata for each class in the artefact and enough metadata about the artefact itself

6.1.4 Why are they useful?

As outlined in the introduction, the most immediate use for controlled terminology is to ensure consistency in data entry. But the usefulness of ontologies and controlled vocabularies goes beyond this initial use. The main purpose of biomedical ontologies is to structure knowledge so it can be operated on by software agents.

One needs to also understand that the two processes coexist and operate in parallel. As more experiments are performed, new discoveries are made. This new knowledge needs to be represented in the domain ontology so the new notions can be used to annotate the results of earlier experiments in the context of retrospective analysis.

For example, The Gene Ontology (GO) is a widely used resources to describe `Molecular Processes`, `Biological Functions` and `Molecular Components`. The Gene Ontology Consortium maintains the controlled vocabulary itself but also releases of Genome Wide Gene Ontology Annotations. These are resources which associate genes and genomic features found in those genomes with GO terms. These are very important resources especially in the context of genome wide analysis such as transcriptomics profiling analysis. A particular type of analysis, `enrichment analysis`, relies on the availability of such annotations to detect departures from expected probability distribution in an expression profile and which biological processes are most affected in specific conditions.

The applications are plentiful. The importance of ontologies for structuring information will only grow with the need to the obtain Machine Learning ready datasets and speed up the readiness of datasets. This is what FAIR is all about.

6.1.5 Are all ontologies compatible with each other?

There is not simple answer to that question as it depends heavily on the type of tasks data scientists have in mind. If the purpose is simply to improve query recall on a limited set of fields, a curation policy could be devised to mix and match resources to meet the needs at hands, possibly by building an application ontology.

However, in a more integrated framework, it is important to be aware of the some development choices made by the maintainers of the semantic artefacts.

- **In the context of basic research and model organism based research**, the **`OBO foundry`** is an organization which coordinates the development of interoperable resources. GO, mentioned earlier is one of them. The establishment of domain specific reference ontologies sharing the same underlying rules means that some level of compositional development can be done. By this, it means that axioms can be built connecting classes from compatible resources. This point becomes particularly important when considering the role of `reasoner` when assessing and checking the consistency of artefacts themselves but also when analysing instance datasets themselves.
- **In the context of observation studies**, the OMOP model also relies on controlled terminologies such as SNOMED-CT, RxNORM for drugs and LOINC for clinical and laboratory test descriptions.
- **In the context of Clinical Data collections**, the CDISC models work tightly with CDISC Terminology, National Cancer Institute's Enterprise Vocabulary Services (EVS) and also recommend use of SNOMED-CT and terminologies such as LOINC, both of which come with specific licensing terms users need to get familiar with.

Use Cases and Iterative Approach

The use and implementation of common terminologies will enable a normalization/harmonization of variable labels (data label) and allowed values (data term) when querying a database. Implementing the use of common terminologies in the curation workflow will ensure consistency of the annotation across all studies.

Selection Criteria

A set of widely accepted criteria for selecting terminologies (or other reporting standards) do not exists. However, the initial work by the Clinical and Translational Science Awards' (CTSA) Omics Data Standards Working Group and BioSharing has been used as starting point top define possible criteria for excluding and/or including a terminology resource.

- **Exclusion criteria**:
  - :x: absent licence or terms of use (*indicator of usability*)
  - :x: restrictive licences or terms of use with restrictions on redistribution and reuse
  - :x: absence of term definitions
  - :x: absence of sufficient class metadata (*indicator of quality*)
  - :x: absence of sustainability indicators (*absence of funding records*)
- **Inclusion criteria**:
  - :heavy_check_mark: scope and coverage meets the requirements of the concept identified
  - :heavy_check_mark: unique URI, textual definition and IDs for each term
  - :heavy_check_mark: resource releases are versioned
  - :heavy_check_mark: size of resource (*indicator of coverage*)
  - :heavy_check_mark: number of classes and subclasses (*indicator of depth*)
  - :heavy_check_mark: number of terms having definitions and synonyms (*indicator of richness*)
  - :heavy_check_mark: presence of an help desk and contact point (*indicator of community support*)
  - :heavy_check_mark: presence of term submission tracker / issue tracker (*indicator of resource agility and capability to grow upon request*)
  - :heavy_check_mark: potential integrative nature of the resource (*as indicator of translational application potential*)
  - :heavy_check_mark: licensing information available (*as indicator of freedom to use*)
  - :heavy_check_mark: use of of top level ontology (*as indicator of a resource built for generic use*)
  - :heavy_check_mark: pragmatism (*as indicator of actual, current real life practice)*
  - :heavy_check_mark: possibility of collaborating: the resource accepts complaints/remarks that aim to fix or improve the terminology, while the resource organisation commits to fix or improve the terminology in brief delays (one month after receipt?)

These criteria are simply indicative and need to be modulated depending on the `contexts` described in the introduction, as specific constraints (e.g. regulatory requirements) may take precedence over some of the criteria listed here.

6.1.6 Conclusions

Choosing ontology and semantic resources is a complex issue, which requires careful consideration, taking into account the research context of the data production workflow, regulatory requirements that may apply. The choices made affect the integrative potential of a dataset as they influence the level of `interoperability`. Clearly declaring the semantic resources used to annotate a dataset also influence `findability` and `reusability` and it is good practice to do so.

6.1.7 What to read next?

- *Ontology services*
- *CFDE selected terminologies*

6.1.8 Bibliography

1. RDF.https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/
2. SKOS. https://www.w3.org/2004/02/skos/
3. OWL. https://www.w3.org/OWL/
4. Hermit. http://www.hermit-reasoner.com/
5. Elk. http://www.cs.ox.ac.uk/isg/tools/ELK/
6. OBO Foundry. http://obofoundry.org/
7. CDISC. https://www.cdisc.org/standards
8. CDISC Controlled Terminology. https://www.cdisc.org/standards/terminology
9. LOINC. https://loinc.org/
10. Gene Ontology. http://geneontology.org/
11. Protégé. https://protege.stanford.edu/
12. Topbraid composer. https://www.topquadrant.com/products/topbraid-composer/
13. INCAtools. https://github.com/INCATools
14. ROBOT. R.C. Jackson, J.P. Balhoff, E. Douglass, N.L. Harris, C.J. Mungall, and J.A. Overton. ROBOT: A tool for automating ontology workflows. BMC Bioinformatics, vol. 20, July 2019.

6.2 Ontology Services

Authors: Philippe Rocca-Serra

Maintainers: Philippe Rocca-Serra

Version: 0.1

6.2.1 Objectives

The aim of this chapter is to provide an overview of services and infrastructures available around terminologies and controlled vocabularies. The services referred here encompass the following:

- terminology hosting
- terminology versioning
- terminology browsing
- term selection and metadata display
- free text annotation and semantic markup
- automatic annotation
- ontology mapping services

Clinical Trial Data

Operating in the field of Clinical Trials means that datasets are generated during `interventional studies`, meaning that researchers influence and control the predictor variables, which are usually different intensity levels of therapeutic agents in order to gain insights in terms of benefits in patient outcomes. In this context, regulatory requirements make it so that data must be recorded in standard forms to allow for review and appraisal by US FDA reviewers. This means that the CDISC standards are the *de-facto standard* in the area, which mandates the use of semantics resources such as:

Semantic Resource	Domain	License	Format	Service
CDISC vocabulary	clinical trial data			EVS
NCI Thesaurus	biomedicine			EVS,Bioportal
SNOMED-CT	pathology			EVS,Bioportal
UMLS	pathology			EVS,Bioportal
LOINC	laboratory tests			Bioportal
RxNORM	drugs			Bioportal
GUDID	instruments			accessGUDID

All available from the NCBI EVA system.

> :warning: Some resources are only available under restrictive licences, which prevent derivative work, which may limit access and use.

Observational Health Data

This context refers to data collected during observation studies, which in constrat to `interventional studies`, draws inferences from a sample to a population where the independent variable is not under the control of the researcher because of ethical concerns or logistical constraints [1]. This is typically the case in the context of epidemiological work or exposure follow-up studies in the context of risk assessment and evaluation of clinical outcomes. `Observational health data` can also include `electronic health records (EHR)` or `administrative insurance claims` and allow research around acquiring *`real world evidence`* from large corpora of data. In this specific context, a model and associated set of standards has been particularly successful. With several hundred millions of patient information structured using the **Observational Medical Outcomes Partnership (OMOP)**, the Observational Health Data Sciences and Informatics (OHDSI) `open-science community` has been particularly successful. Therefore, building a FAIRification process around the standard stack produced by the ODHSI community needs to be considered if operated in such a `data context`.

The vocabulary server Athena is the tool developed to support the needs of the OMOP/OHDSI community. Accessing the vocabulary services means accepting the SNOMED-CT terms of use.

The table below presents a small subset of semantic resources available from the Athena service (the largest and most commonly used)

Semantic Resource	Domain	License	Format	Service
SNOMED-CT	pathology			Athena
UMLS	pathology			Athena
LOINC	laboratory tests			Athena
RxNORM	drugs			Athena
UCUM				Athena
ICD9/10				Athena

:warning: Some resources are only available under restrictive licences, which prevent derivative work, which may limit access and use.

For a more detail view and deep-dive into the ODHSI and OMOP semantic support, the reading the chapter dedicated to the `controlled terminology` in the **`Book of OHDSI`**

Basic research context

This refers to datasets and research output being generated using model organisms and cellular systems in the context of basic, fundamental research. In this arena, the regulatory pressure is much less present, but this does not rule out data management good practice and proper archival requirements. As a consequence of fewer constraints, researchers are often confronted with a sea of options. This section aims to provide some guidance when tasked with deciding on which semantic resource to use.

:bell: An important consideration

to bear in mind when writing selecting semantic resources is to assess whether or not `data archival in public repositories will be required`. For instance, submitting to NCBI Gene Expression Omnibus Data archive places no requirement but if depositing to EMBL-EBI ArrayExpress, then selecting a resource such as the Experimental Factor Ontology could ease deposition.

:bell: The FAIRsharing registry

is an ELIXIR resources, which provides invaluable content as the catalogue offers an overview of the various semantics artefacts used by public data repositories.

A number on open source, public and for some of them, interoperable by design resources are developed and distributed via the OBO foundry.

These can be accessed via several vocabulary services maintainde by public institutions:

Service	web address	hosting institution
Ontobee	http://www.ontobee.org/ontology/	University of Michigan Medical School
Ontology Lookup Service	https://www.ebi.ac.uk/ols/index	EMBL-European Bioinformatics Institute
NCBO Bioportal	https://bioportal.bioontology.org/	National Centre for Biomedical Ontologies, Stanford University

---

6.2.2 Conclusions

Semantic artefacts (Lexicons, Controlled Terminologies, Ontologies) can be accessed through a number of publicly available services, each with specific features and services attached to them. For specialized domains, it may be beneficial to access the specialized servers (e.g. Athena), as they, to some extent, limit to search space to only those resources relevant to the fields. This is in contrast to more general purpose, domain agnostic vocabulary services.

6.2.3 What to read next?

- *CFDE selected terminologies?*

6.2.4 References

Smith, B., Ceusters, W., Klagges, B. et al. Relations in biomedical ontologies. Genome Biol 6, R46 (2005). https://doi.org/10.1186/gb-2005-6-5-r46

Rocca-Serra P, Bratfalean D, Richard F, Marshall C, Romacker M., Auffray C, ., … on the behalf of the eTRIKS consortium, . (2016, April 25). eTRIKS Standards Starter Pack Release 1.1 April 2016. Zenodo. http://doi.org/10.5281/zenodo.50825

Malone J, Stevens R, Jupp S, Hancocks T, Parkinson H, Brooksbank C. Ten Simple Rules for Selecting a Bio-ontology. PLoS Comput Biol. 2016;12(2):e1004743. Published 2016 Feb 11. http://doi.org/10.1371/journal.pcbi.1004743

Bairoch A. The Cellosaurus, a cell line knowledge resource. J. Biomol. Tech. (2018) 29:25-38. http://doi.org/10.7171/jbt.18-2902-002.

Sansone, S.-A., McQuilton, P., Rocca-Serra, P., Gonzalez-Beltran, A., Izzo, M., Lister, A.L. and Thurston, M. (2019) FAIRsharing as a community approach to standards, repositories and policies. Nature biotechnology, 37, 358: http://doi.org/10.1038/s41587-019-0080-8.

Hripcsak, G., Ryan, P. B., Duke, J. D., Shah, N. H., Park, R. W., Huser, V., Suchard, M. A., Schuemie, M. J., DeFalco, F. J., Perotte, A., Banda, J. M., Reich, C. G., Schilling, L. M., Matheny, M. E., Meeker, D., Pratt, N., & Madigan, D. (2016). Characterizing treatment pathways at scale using the OHDSI network. Proceedings of the National Academy of Sciences of the United States of America, 113(27), 7329–7336. https://doi.org/10.1073/pnas.1510502113

Hripcsak, George et al. "Observational Health Data Sciences and Informatics (OHDSI): Opportunities for Observational Researchers." Studies in health technology and informatics vol. 216 (2015): 574-8.

6.3 NIH CFDE Selected Terminologies

Authors Philippe Rocca-Serra

Maintainers Philippe Rocca-Serra

Version: 0.1

6.3.1 Objectives

The main objective of this section is to draw the attention to the importance of semantics in `interoperability` and `reusability` as implemented in the C2M2 model and the associated task of ingesting datasets from an array of resources, each tackling a specific research area.

A secondary objective is to prepare for the `extract/transform/load` (ETL) processes by simply being aware of this model requirements.

6.3.2 Overview

For a number of attributes, the C2M2 model delegates to controlled terminologies and ontologies the associated value sets definition. This affords specification stability while allowing flexibility by outsourcing the maintenance needs for value set, typically when new values are required. While the C2M2 model specifications clearly identifies the elements requiring values selected from a controlled terminology, the following table offers a full overview. The table also highlights the planned implementation, with phased releases from compliance level 0 through to compliance level 2, which will see the inclusion of new types, thereby extending the `interoperability` aspect of the FAIR potential of datasets made available by the DCC through the Deriva-based system following the `extraction transformation and load process` from the source repositories.

6.3.3 C2M2 vetted Vocabularies

Domain	Resource Name	License	C2M2 Level 0	C2M2 level1	C2M2 level2
id_namespace_string	CFDE internal CV	NA	:heavy_plus_sign:	:heavy_plus_sign:	:heavy_plus_sign:
subject_role	CFDE internal CV	NA		:heavy_plus_sign:	:heavy_plus_sign:
subject_granularity	CFDE internal CV	NA		:heavy_plus_sign:	:heavy_plus_sign:
protocol	CFDE internal CV	NA		:heavy_plus_sign:	:heavy_plus_sign:
taxonomy	NCBITax	CC0 1.0 (public domain)		:heavy_plus_sign:	:heavy_plus_sign:
anatomy	UBERON	CC-BY		:heavy_plus_sign:	:heavy_plus_sign:
sample_type	OBI	CC-BY		:heavy_plus_sign:	:heavy_plus_sign:
assay_type	OBI	CC-BY		:heavy_plus_sign:	:heavy_plus_sign:
file_format	EDAM	CC BY-SA 4.0		:heavy_plus_sign:	:heavy_plus_sign:
data_type	EDAM	CC BY-SA 4.0		:heavy_plus_sign:	:heavy_plus_sign:
disease	MONDO	CC-BY			:heavy_plus_sign:
disease	DOID	CC-BY			:heavy_plus_sign:

6.3.4 Which terminologies DCCs currently use?

For each of the potential data sources and for a set of core search facets, a survey of semantic resources used by representative DCCs has been summerized in the table below.

> :warning:It is worth noting that the table includes a subsection (indicated in *italic*) which covers identification schemes used for molecular entities. These are distinct from concept annotation with ontology terms, however since they allow interoperability between resources, they have been included).

Domain	MW	LINCS	HMP	GTEx	4D Nucleome	KidsFirst
taxonomy	free text	free text	free text	free text	free text	free text
anatomy	free text	free text	free text	UBERON	free text	NCIT
sample type	free text	free text	free text	UBERON	free text	NCIT
disease	free text	free text	free text	free text	free text	HPO NCIT MONDO
assay type	internal cv/free text	BAO	internal cv/free text	internal cv/free text	internal cv/free text	internal cv/free text
data type	_	free text	free text	_	internal cv/free text	internal cv/free text
chemical compound	pubchem CID,InChi	pubchem CID,InChi	_	_	_	_
gene product	refseq	_	_	_	_	_
protein	uniprot	_	_	_	_	_

6.3.5 Conclusions

- By explicitly identifying a number of semantic artefacts for describing key attributes, the C2M2 defines a curation framework, with the aim of anchoring free text descriptors to controlled terms, which can be exploited for query expansion or resource linking.
- The resource survey that has been carried out is an important step in the FAIRification process as it identifies potential `areas of intervention`, defined as semantic markup of free text description can deliver gains in `interoperability` and `reusability`.
- Taking the notion of `taxonomical descriptors` for example, the harmonization across the various sources can be easily achieved by relying on a resource such as NCBITaxonomy and the curation action is simplified by that limited diversity of *species* found in the different databases.
- On the other hand, the harmonization tasks for domains such as `sample type`, `assay type` can be more involved, not to mention the case of `phenotypic descriptions` or `disease`, even though `level 0 and level 1 compliance` do not expect such a degree of integration.

6.3.6 What to read next?

- *CFDE C2M2 model*
- *ETL to CFDE C2M2 model*

CHAPTER

SEVEN

FAIR COMPLIANCE

This section is dedicated to describing the processes and tools available to perform a quantitative assessment of the level of FAIR compliance of a given resource. It includes:

- A description of the FAIRshake service and how it is being used to evaluate the output of first ETL processes developed under NIH-CFDE.
- A guideline and an example of developing FAIR application program interface, as FAIR also applies to services, and not just datasets.

7.1 Evaluating FAIRness with FAIRshake

A tutorial that demonstrates how to use FAIRshake to perform FAIR evaluations of DATS serialized metadata in the context of the CFDE.

Authors: Daniel J. B. Clarke

Maintainers: Daniel J. B. Clarke

Version: 1.1

7.1.1 Background

Adhering to FAIRness is somewhat abstract. While all of the components of becoming FAIR can be addressed at some level, it remains difficult to provide a concrete answer about whether something is indeed FAIR or not. In general, improvement is only real if it can be measured. To address this limitation of the FAIR guidelines, FAIRshake was created with the basic goal of making FAIR more concrete and measurable. While FAIRshake provides a catalog of community-contributed ways to characterize FAIRness, it is still up to a given project to decide which of these criteria they will adopt and/or create.

FAIRshake provides:

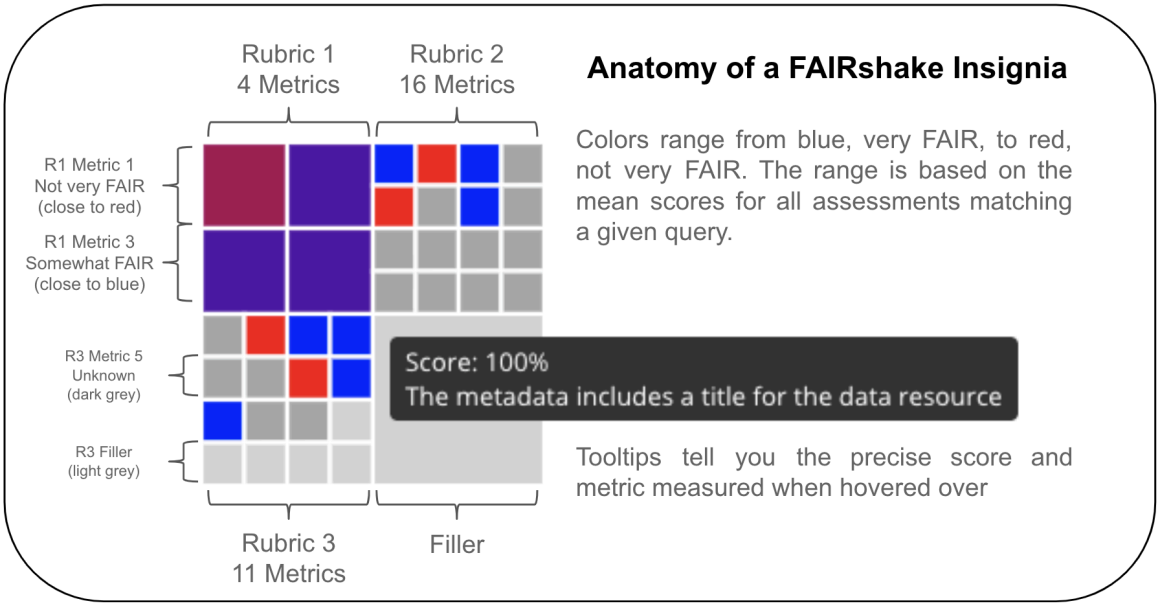
- A catalog of digital objects: these can be, for example, datasets, APIs, workflows, each having their own unique identity and is the target of a FAIR assessment. That-is whatever the digital object is, you want to assess how much it is Findable, Accessible, Interoperable and Reusable.
- A catalog of projects: where a project contains any set of digital objects grouped for the purpose of analytic and findability, e.g., all digital objects that belong to a specific NIH Common Fund program could be bundled into one project. If you plan on automating FAIR assessments, it makes sense to do it as part of a project so that assessments can be compared only against other assessments within your project.

- A catalog of metrics: these are any singular FAIR criterion, or a FAIR compliance question, that can often be answered with yes/no/percentage of compliance. It is often the case that manual assessments are qualitative yes (1) / no (0) while automated assessments can often be more granular.
- A catalog of rubrics: these are sets/bundles of FAIR metrics meant to be answered together, e.g., a "FAIR" API should satisfy several independent metrics already registered in FAIRshake, these can be a part of one or more rubrics.
- Facilitation of FAIR assessments: any digital object can be assessed with a given rubric in the context of a project both manually through the FAIRshake website, or 'automatically' by enabling assessment registration over API. Some automatic assessments have been integrated into the manual assessment UI on FAIRshake but this is still under development. Contributing your own automatic assessment modules will be discussed in this tutorial.
- Aggregations of FAIR assessments: FAIRshake provides the FAIR insignia, a look at the average assessments of a given digital object, project, or rubric. It also provides project analytics in the form of a report with summary statistics charts.

7.1.2 Motivation

By selecting and adhering to a rubric or set of metrics shared by other projects, a resource can independently assert metric conformance, thus improving interoperability between resources that share those metrics. Each metric defines a concrete ideal criterion that is desired and satisfaction of that metric can be represented as a percentage or a number between 0 and 1. Some metrics may be categorical, in which case their contribution can be defined as discrete scores from 0 (least desirable) to 1 (most desirable). Given these normalized bounds, we can always compute a single scalar within the same range by finding the mean value of scores.

The FAIR insignia aggregates each metric separately to inform someone where they can do better (metrics that have a low percentage) and where they are already doing well (metrics that have a high percentage). Digital objects may be assessed by different rubrics (sets of metrics), which are often made up of different metrics.



These insignias capture a visual snapshot of a resources' aggregated assessments at a glance. Interactive tooltips shown by hovering over a particular square reveal which metric is represented by that square. Clicking a given box will bring you to a landing page with detailed information about the metric.

It is important to note that these insignias can only represent knowledge that is reported and as such, a "low score" should be interpreted as something to look into and not something to be accused of. It's also important to note that FAIR in general **does not represent quality of data but rather an expectation of how easy it might be to find, access, interoperate with and reuse that data**. By using automated mechanisms or strict clear-cut guidelines, we can determine a score for this expectation.

As a simple example, consider a metric which wants to assess whether a citation can be located for a dataset from its landing page (a url). A human would look on the page and report whether they found it or not on the page, however a robot might depend on data citation guidelines, which would expect to find a DOI or semantically annotated microdata or JSON-LD. While a robot might miss the obvious human-readable citation available on the page, it would also mean that a browser extension or bioinformatic crawling effort **would likely also miss it**. As such, a metric that is *not completely satisfied* may impair a use-case that depends on FAIR. FAIR Assessments can help identify situations like this and drive improvements.

To increase your FAIR score, you should identify which metrics may need improvements, learn more about what that metric covers both theoretically (what the metric says) and concretely (how it was actually assessed). It is important to make sure that you are improving the overall FAIRness of your resource, and not just "hacking the FAIR metrics." In the context of the CFDE, periodic FAIR assessments are performed using a common rubric based on compliance with the C2M2. Assessmenets are also detailed in another FAIR recipe dedicated to the CFDE rubric and is also addressed later in this recipe in the context of automated FAIR assessments. We encourage you to question when scores for certain metrics do not reflect what you feel is correct; it will take investigation of the metric itself, your own resource, and the C2M2 metadata models capturing of your resource. We encourage you to re-purpose the rubric we are using along with any additional metrics you hope to satisfy and assess your own resources. Comparing the assessments on your actual data and the assessment on your C2M2 converted data may reveal areas for improvement.

7.1.3 Ingredients

1. A digital object or set of digital objects to assess for FAIRness
2. A rubric from FAIRshake encapsulating the FAIR metrics you wish to use to perform the FAIR assessments
3. Machine-readable (ideally standardized) metadata description for enabling automated assessments

7.1.4 Objectives

In this recipe we will look at the process of performing a FAIR evaluation using FAIRshake starting from scratch and covering various decisions that must be made along the way. We will use the CFDE DCC resources already *transformed to the C2M2* as the target of our assessment. This is because an automated assessment that is common across all CF DCCs is not possible without a common machine-readable metadata standard.

1. Use FAIRshake to facilitate FAIR Rubric discovery and development
2. Assess a digital object manually
3. Identify avenues for performing automated assessments
4. Perform an automated assessment on a single digital object serialized with machine-readable metadata, and demonstrate how it can be applied globally
5. Develop an understanding of how well the digital object(s) comply with the chosen rubric
6. Learn how to contribute new automated assessments to the FAIRshake ecosystem

7.1.5 Recipe

Because these ingredients are generic, we will scope the recipe by considering a concrete scenario for each part of the recipe.

Introduction

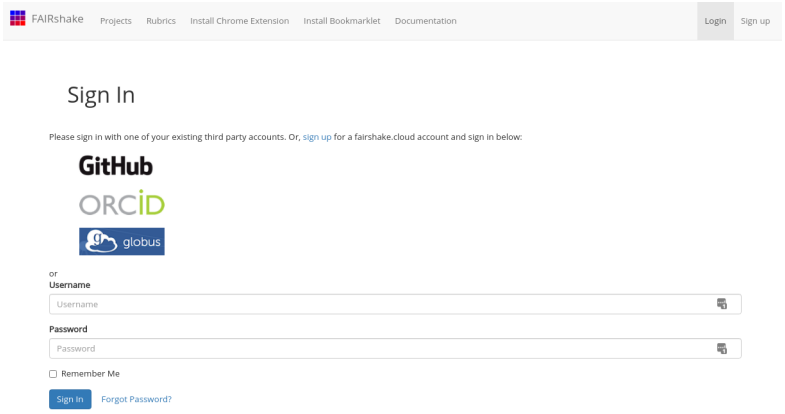
Scenario

Janice is a researcher at a Common Fund program who wants to assess her dataset using the CFDE rubric on FAIRshake that was used for the CFDE resources. After performing this assessment, she hopes to discover ways to improve her score. She has a resource in mind, but would first like to get familiarized and set up with FAIRshake which she understands might be helpful.

Using FAIRshake

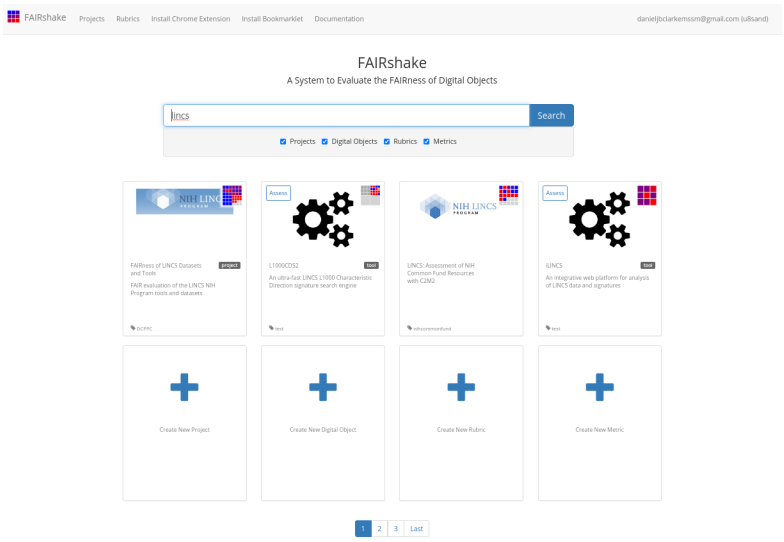
FAIRshake can be accessed at fairshake.cloud. There are several YouTube tutorials and some general and technical documentation accessible on the website.

After logging in to the website, you will be able to create content on the site including registering a project, digital object, rubric, metric, or performing a FAIR assessment.



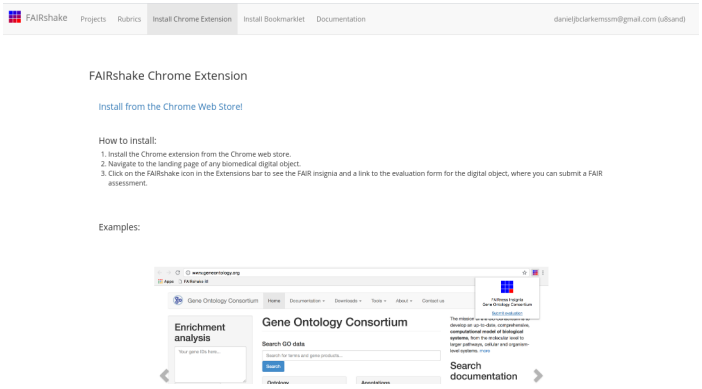
After logging in:

You are brought back to the home page where you can perform searches to locate projects, digital objects, rubrics or metrics by name, perform assessments or add new elements.

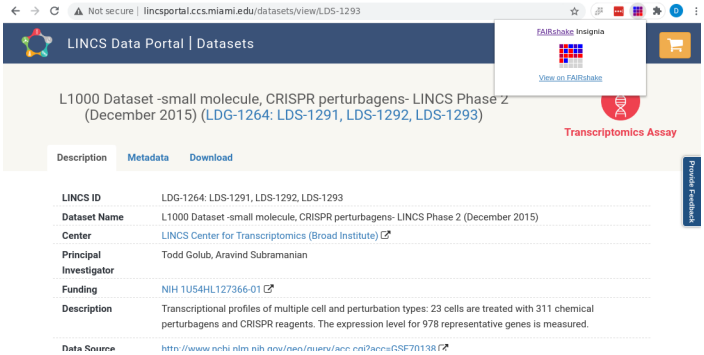


Scenario

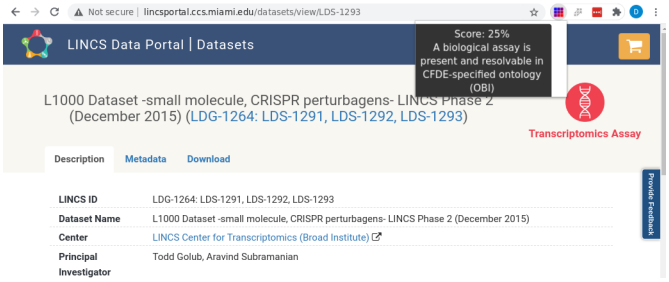
Janice is interested in a resource she contributed to: L1000 dataset of CRISPR perturbagens. She had a hard time finding it in the search bar so she decided to try the FAIRshake Chrome extension.



After installing the extension she went to the resource's own landing page (not the one on FAIRshake!) and activated the extension to find that, in fact, an assessment already exists for her already-C2M2 cataloged resource.



She points her mouse over some of the red squares revealing information she does not quite understand.

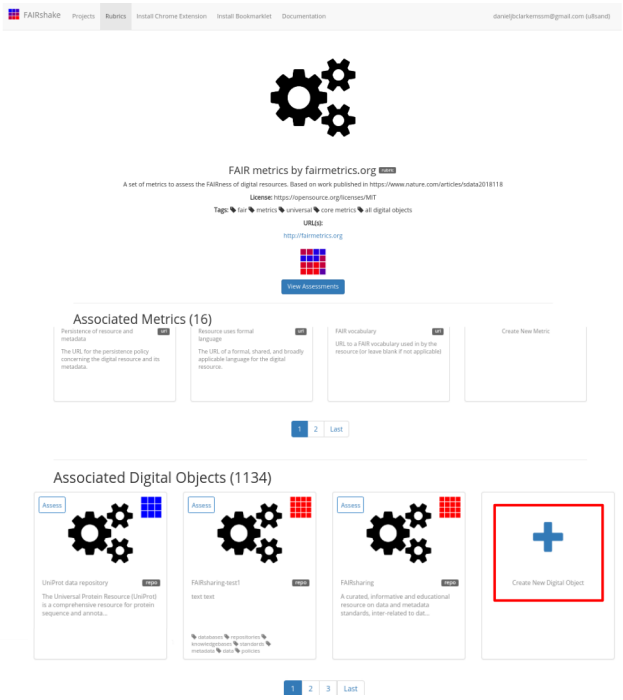


Although an assay is listed and described accurately on the page, there is no OBI term available on the page. She hopes to understand how this answer came to be and learn how the detailed and valuable assay information shown on the landing page can be made more FAIR.

Assess a digital object manually

The C2M2 rubric was developed by the CFDE team to represent the concrete areas of FAIRness that the CFDE plans to focus on and ideally satisfy in order to accomplish several use-cases determined at the beginning of the project. The metrics chosen represent some broad subset of FAIR, but do not necessarily cover all aspects necessary to make digital objects FAIR in *your* community.

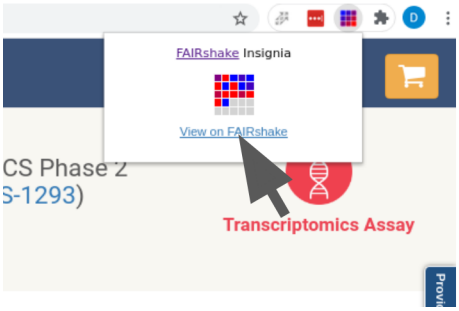
To that end, and to get a better sense of the scope of the FAIR metrics that could be developed to better serve your community, let's take a look at the FAIR metrics by fairmetrics.org Rubric. This rubric is a FAIRshake entry for the universal FAIR metrics published in this paper, representing a universal set of broad criteria that should apply to all digital objects.



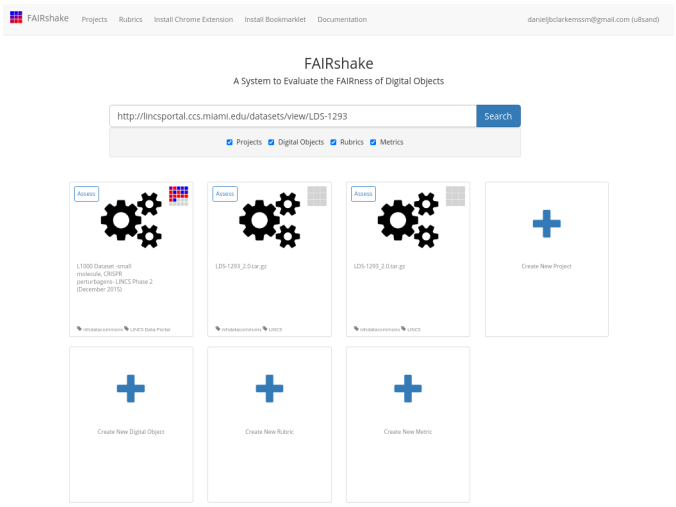
If you are following along, feel free to pick a digital object of your own that you know well and see if you can complete and publish a manual assessment with it! You can always find and delete it from your account if you choose not to keep it, just be sure not to publish it unless you're happy with it.

Scenario

Janice decides to perform a manual assessment of her resource using the FAIRmetrics rubric before digging into the specifics of the CFDE assessment.

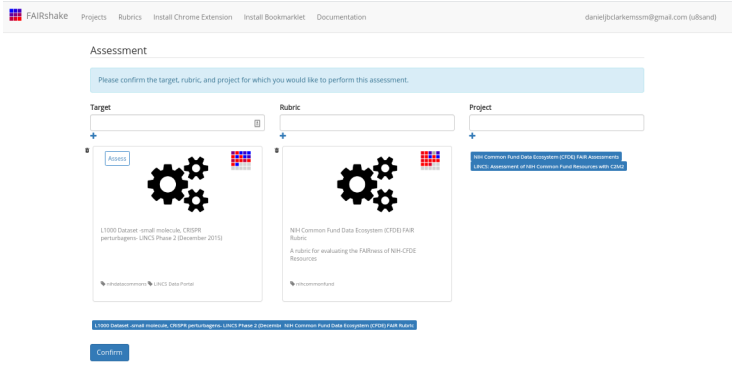


Which brings her to FAIRshake to see the relevant information available on FAIRshake related to the page she was on.

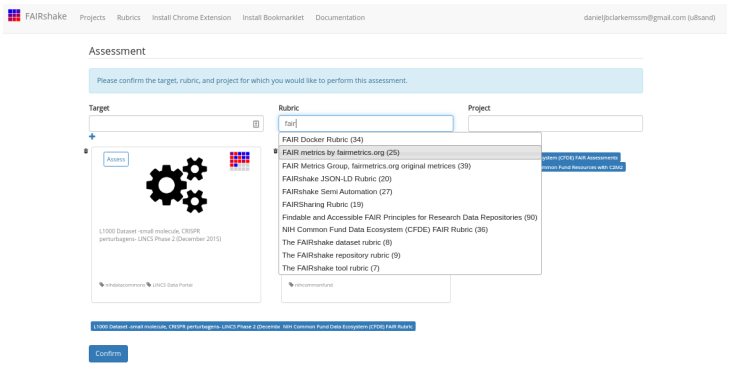


Alternatively, she could have found or registered her digital object directly on the FAIRshake website with the 'Create New Digital Object' button.

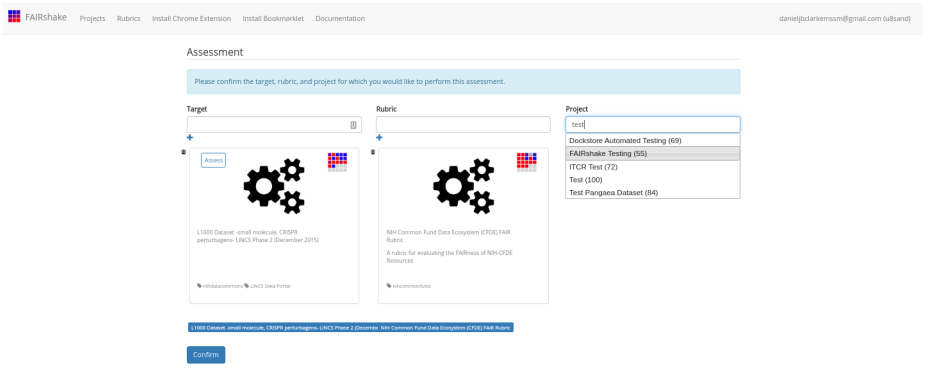
Clicking the assess button, she ends up at the assessment preparation page.



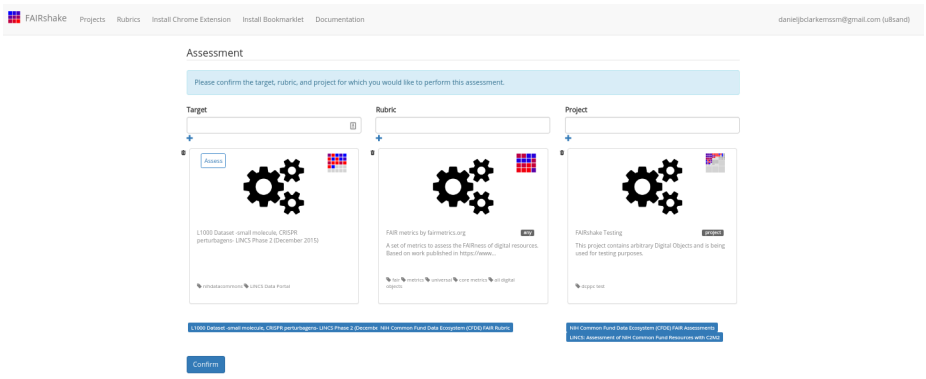
The digital object and its only rubric were selected automatically, but she ends up instead select the fairmetrics rubric.



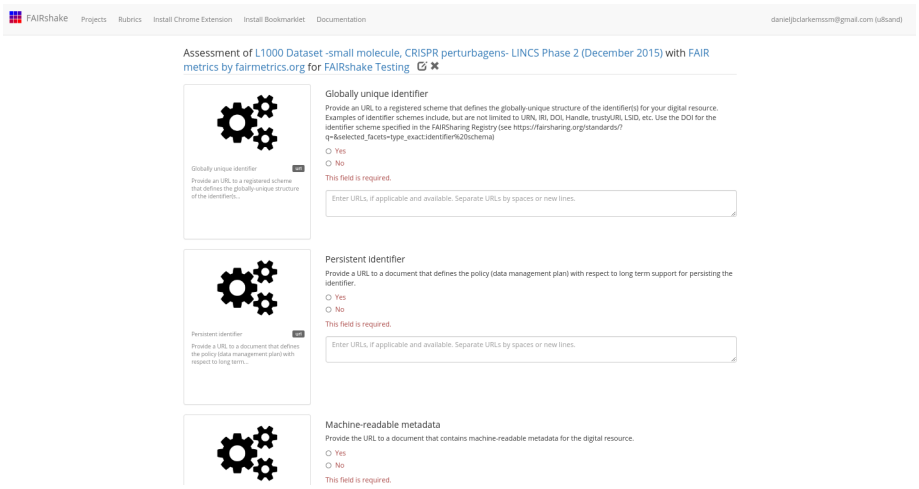
Instead of using the CFDE project, she will perform this assessment as part of the FAIRshake testing project. It will likely make sense to create our own project if we expect to do a bunch of related assessments.



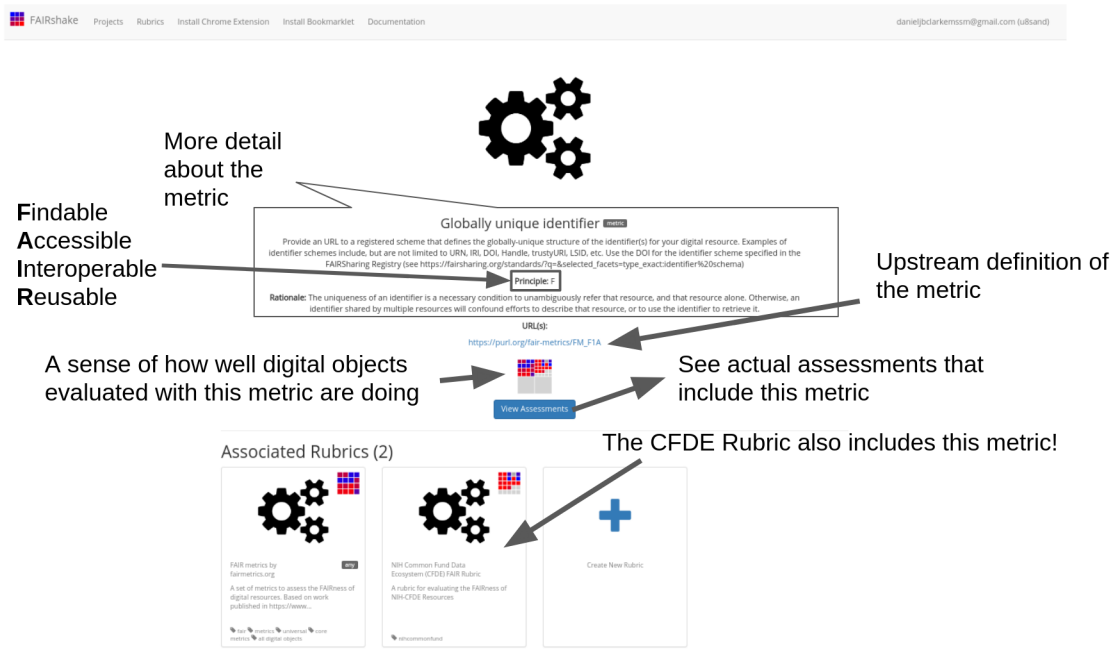
It is also important to note that project here can be left blank if our assessment is not for any project.



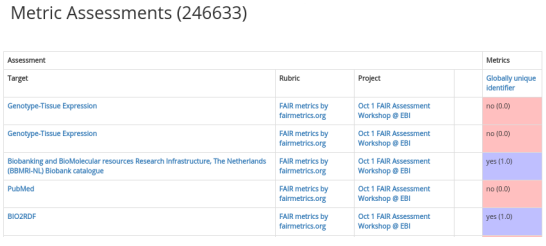
Confirming this, Janice begins a manual assessment.



Each metric represents a concept pertinent to FAIRness, which is described shortly before each prompt, but potentially in more depth on the metrics' landing page. Clicking on the metric "card" to the left of the question she gets much more information in a new tab.



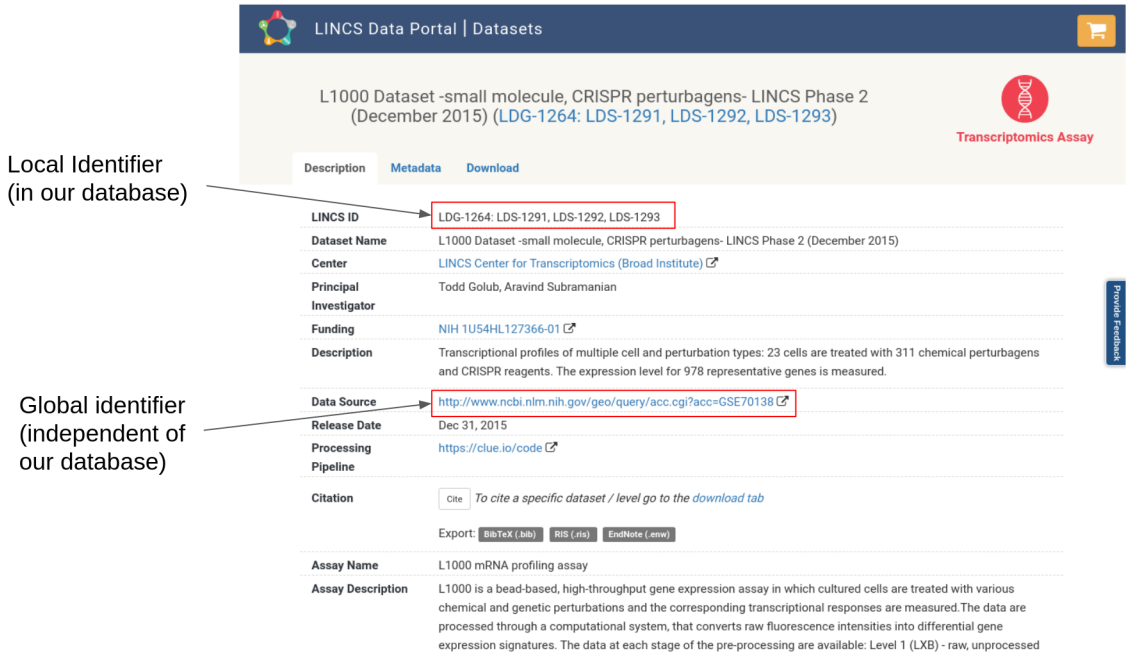
Clicking 'View assessments' she can even see what other digital objects in the database got as an answer during an assessment through a tabular view.



Clicking on any of these links will allow you to explore the projects, rubrics, or digital objects that were assessed to provide a more elaborate sense of why a particular score was received and in what context. We can see, for example, that these

top entries refer to assessments made during an EBI workshop.

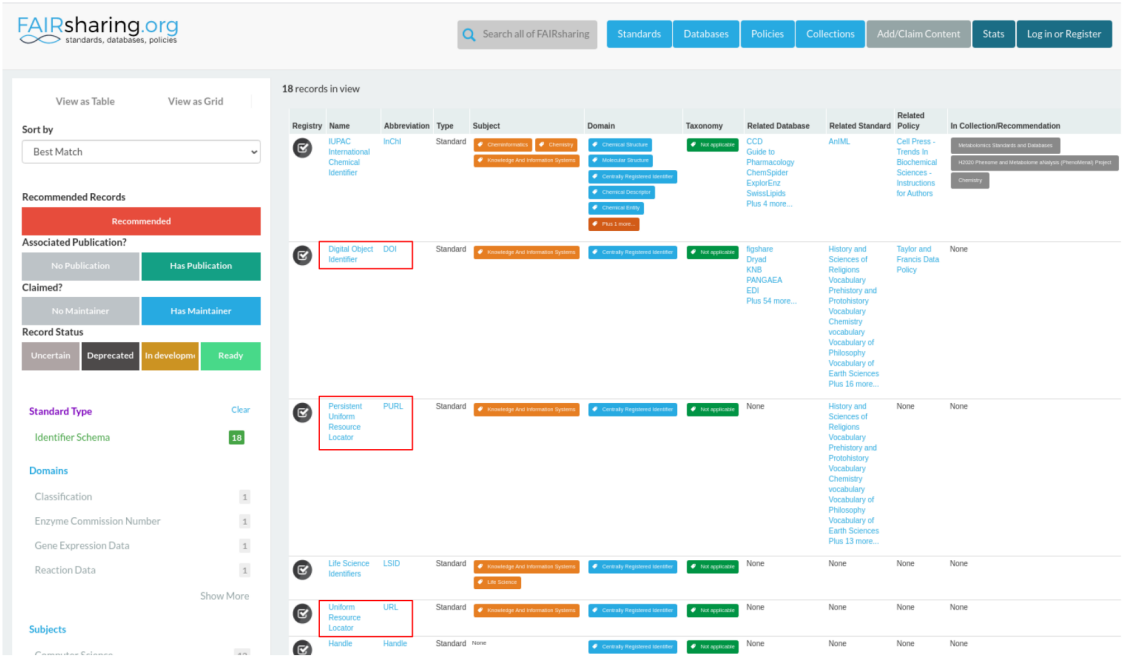
Getting back to the assessment, Janice must determine whether the digital object satisfies the criterion at hand. This one asks us to provide a standard that defines the globally-unique structure of the identifier used for the resource.



She finds out quite quickly that there are several identifiers:

- the data source: http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE70138
  - the data source's local identifier GSE70138
- the local identifier: LDS-1293
- the url is an identifier http://lincsportal.ccs.miami.edu/datasets/view/LDS-1293

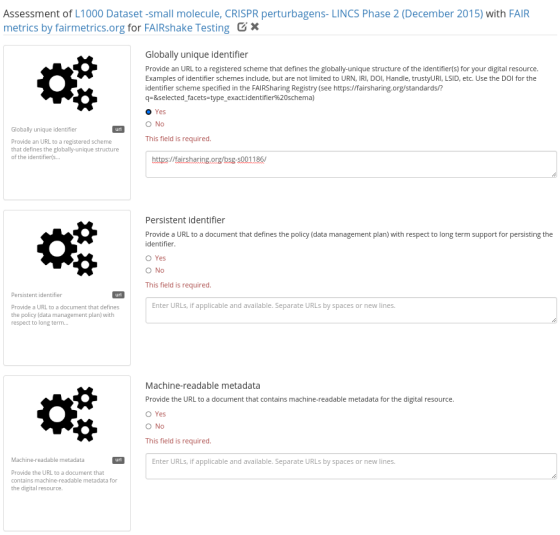
While all of these are identifiers, not all of them are used outside of the resource itself and thus shared as "globally" accepted. The *scheme* however is shared because URL appears in the FAIRsharing database along with DOI and other standardized identifier schemes.



A **URL** provides some level of standardization more than, say, a digital object that *doesn't have a resolvable URL*. But other identifier schemes may carry with them even more information, like a **DOI**, which adds additional semantic interoperability conditions not present on URLs. Thus, in certain circumstances, a URL might be good enough, but in others, a more specific standardized identifier might be more pertinent.

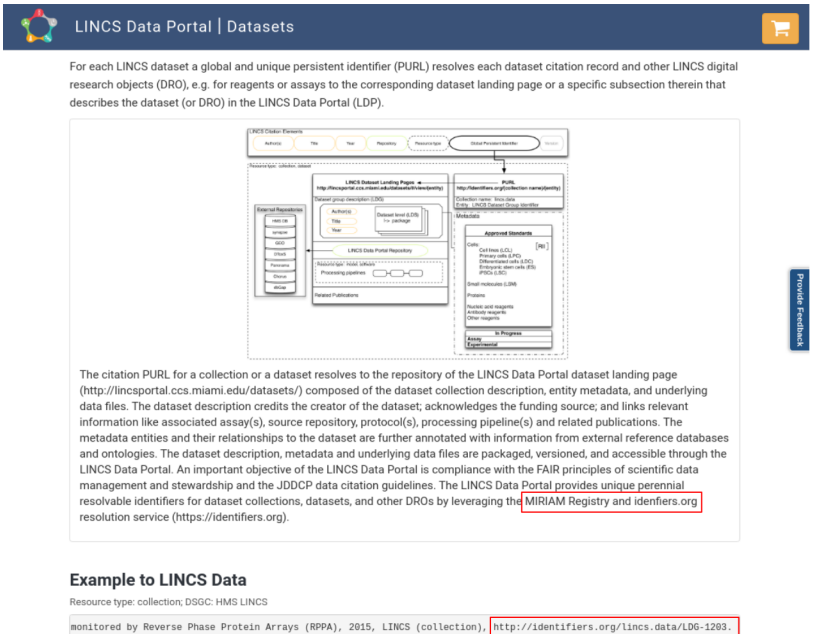
While a DOI guarantees authorship information associated with it, a URL may very well contain absolutely anything. Furthermore, many organizations have come together to try to guarantee that DOIs will not change, while URLs can be changed or removed by the owner of the resource.

Thus the matric *is* satisfied in a broad context. Although if the question was more specific, for instance – "is there a DOI for this digital object?", she might have answered differently. Hopefully, this demonstration helps to illuminate the need for establishing more specific metrics relevant to your community. The more quantitative a metric is, the more stable and useful it will be when measured.



The next metric, persistent identifier, addresses persistence specifically and asks for a document describing the persistent identifier strategy. There is no obvious identifier type that guarantees this so she chooses to investigate further. After

some digging she finds information about citation in the terms:

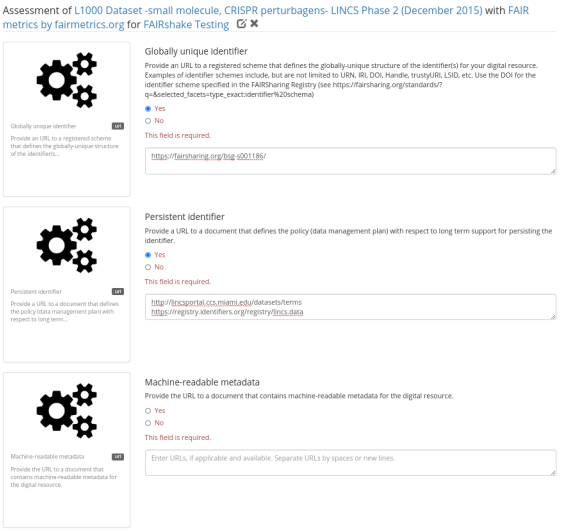


This reveals that our local identifiers are registered in identifiers.org, also recognized as a standard in FAIRsharing. In fact she could create a few more identifiers with this knowledge:

- `lincs.data:LDS-1293`
- http://identifiers.org/lincs.data/LDS-1293

Even if LINCS decides to change the URL structure of its webpage, there is an expectation that these identifiers will be persistent and *not* change in structure. According to the terms, these are meant to be "global and unique persistent identifiers." These identifiers could likely satisfy the persistent identifier criterion citing the scheme as it is registered in identifiers.org, but they are not immediately obvious and available on the landing page.

This demonstrates a scenario where even though LINCS *has* persistent identifiers somewhere, they might not be discovered during the FAIR assessment. Whether we found the answer or not, we can learn something that can be improved.

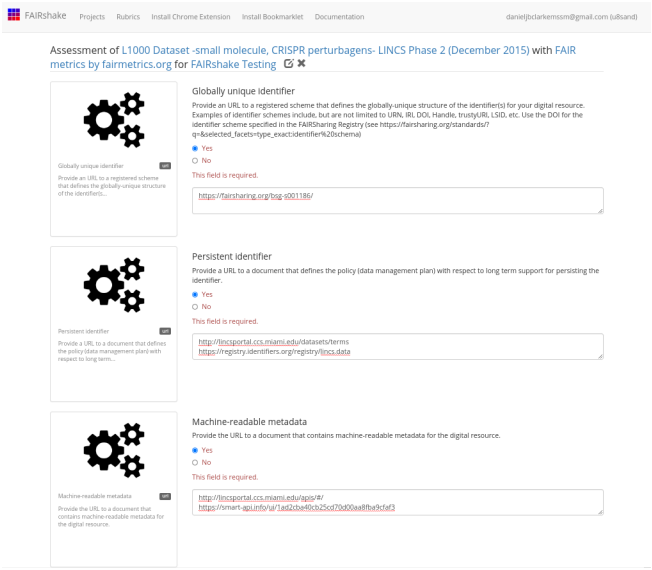


Lastly we will look at the machine readable metadata before discussing automated assessments.

FAIR strives to make things more Findable, Accessible, Interoperable, and Reusable, not just from a human perspective, but also for a machine. With the massive amounts of data available in the public domain, many researchers conduct research by automatically locating data and operating with it without ever directly picking and choosing data sets. To this end it is important that the FAIR principles also be considered from a machine perspective. For example, although the assay is well described on our page, would someone be able to automatically identify datasets on our page with certain criteria such as data or assay-type?

In this vein, machine readable metadata should ideally be available and documented. Again, it is not quite clear from the landing page or even from browsing the website, that there *is* a public API documentation documented and registered in SmartAPI, another community resource also recognized by FAIRsharing.

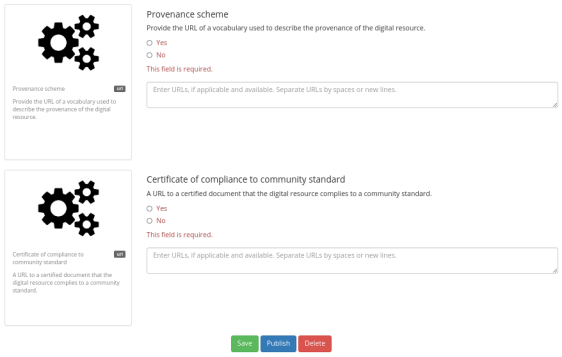
This API provides a structured way of accessing the information on the website making dataset selection and filterability more viable, but nonetheless still not trivial. As such we could say that we have machine-readable metadata, but it does not express the full picture.



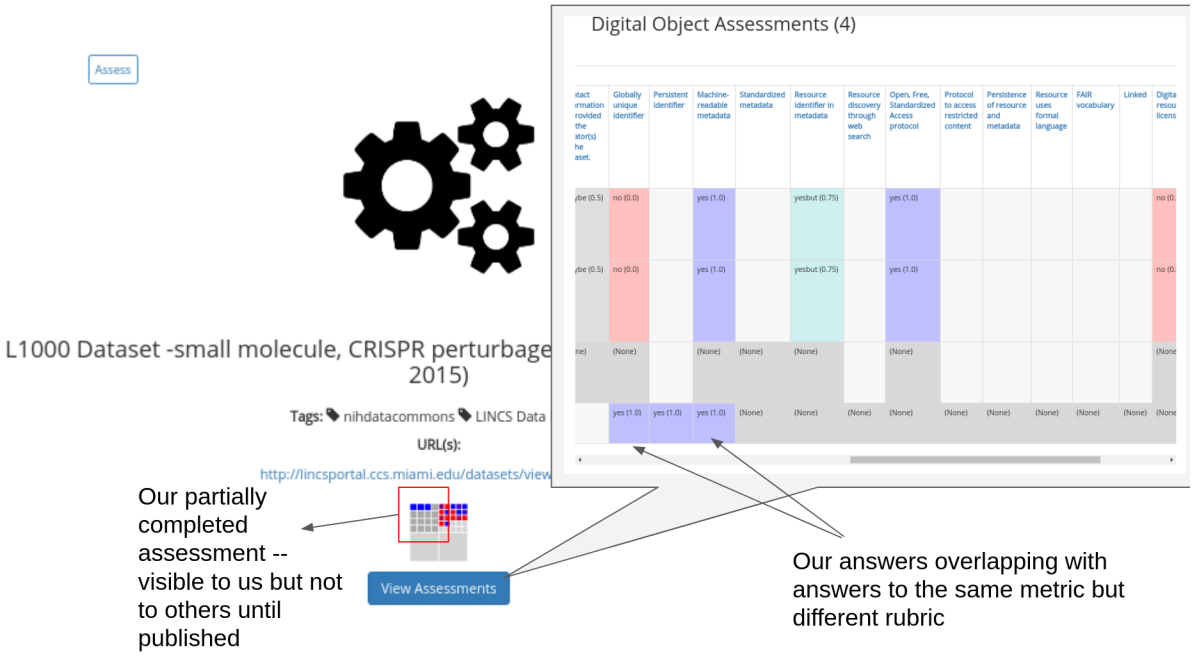
Hopefully, it is clear that the FAIR metrics are broad ideas of things to think about when it comes to FAIR, but we will likely need some more strict and concrete criterion if we are to measure FAIRness with precision. Furthermore, finding this information is a time consuming process and would be intractable with a large enough set of digital objects.

This is where automated assessments and quantifiable metrics come in to help measure the moving target that is FAIRness. It is important to recognize at this point that a "good" or "bad" score produced by manual assessment does little more than prompt discussion. Can someone who *does not* know your resource well come up with the same FAIR assessment as you? If your information is not blatantly obvious the answer will probably be **no**, and this is still valuable even if it is not the precise situation.

When we are done with our assessment, (or just want to save it for later) we can save, publish or delete it at the bottom of the assessment. Once published, an assessment cannot be modified, only one assessment on the same target, rubric, project can be worked on (without publishing) at a time. It is important to note that comments and urls will only be accessible to the authors of the digital object, the assessment, or the project in which it is assessed.



If you complete and publish an assessment, your answers will become associated with the digital object that you assessed, and this information will be used for rendering the insignia and performing the analytics for that digital object.



Although the assessments seem to agree that the digital object has machine readable metadata, it is unclear from an outsider's perspective whether or not a globally unique identifier is present. Next, we will find out exactly why, since those were reported by an automated assessment.

Perform automated assessments

The C2M2 Metadata model defines a unified structure which all DCCs will be converting their own metadata to. With this machine-readable metadata, we can assess FAIRness in an automatic fashion based on the fields available to us. Scripts which demonstrate the conversion of several DCC metadata into the C2M2 are available *here*, and scripts to assess that unified metadata for its compliance with the CFDE Rubric are *here*.

We produced reports over time on the assessments that were executed on the CFDE portal and will continue to do so. This report is summarized *here*. The assessment script can be executed once you have generated C2M2 converted data; documentation for this is out of the scope of this recipe.

If you have a frictionless datapackage containing your data, you can perform a FAIR assessment on that datapackage to identify gaps in your metadata. The script is also capable of performing FAIR assessments on the public repository via

the DERIVA API.

Please note that you need access to the *CFDE FAIR Repo* to access these scripts.

```
git clone https://github.com/nih-cfde/FAIR.git
cd Demos/FAIRAssessment/c2m2

# see script help for more options
python3 assess.py --help

# perform a complete assessment with your frictionless datapackage
python3 assess.py --offline-package=/your/datapackage.json --output-file=output.jsonl
```

Please note that this script tests a number of metrics including validating terms against ontologies, probing links to see if they are available and more, and as such may take some time to run on large amounts of data.

The resulting file, results.tsv, contains a table with the results of the assessment which include answers to each metric for each file in your Frictionless Datapackage. These results should be inspected to determine areas which can be improved. They can be interrogated offline with your favorite spreadsheet program or in the context of the other data *using the same report we produce*, or they can be published onto FAIRshake directly see *Registering assessments on FAIRshake*.

Building your own Automated Assessment

For assessments on completely new sets of digital objects with a completely new rubric, you need to build your own automated assessments. We will walk through how one might go about doing this.

Preparing to Build an Automated Assessment

Certain standards are well-defined and designed in a way that makes it possible to computationally verify whether a digital object is complying with the standard. In an ideal world, all standards should be made in this way, such that an automated mechanism exists for confirming compliance. In reality, however, many standards are not–ultimately necessitating harmonization before datasets, APIs, or anything to be used together.

Some examples of well-defined standards are TCP/IP and HTTP. The effectiveness of these standards and their adoption enables the internet to function and grow as it does. Another, more relevant standard is RDF. RDF defines a way to serialize metadata and permits harmonization via ontologies or shape constraint languages (such as SHACL). Another standard that is not explicitly based on RDF is JSON Schema. JSON Schema builds off of JSON and allows one to use json itself to define what is a valid JSON instance of some metadata. A JSON Schema document can effectively become its own standard given that it is well described and validatable using a JSON Schema validator.

In the case of assessing digital objects that comply with standards that are defined using mechanisms easily validated, automated assessments become simple and in many cases involve simply taking advantage of already constructed mechanisms for asserting compliance with those standards. In the case that those standards are not well-defined; the best course of action would be to convert those digital objects to an alternative and validatable standard, or alternatively formally codify the standard. In either case, you're doing some FAIRification in an effort to even begin the assessment. We have to do this step because we can not measure compliance with a standard if we do not have a quantifiable standard in the first place! Well we could do it but only manually.

Case Study: Performing an Automated Assessment on DATS

One can think of an automated assessment as a unit/integration test for compliance with a standard. Ideally, this test will reveal issues with integration at the digital object provider level at the benefit of the consumer of those digital objects. Automated assessments are only possible on existing machine-readable metadata and validatable standards, such as DATS. As such we will utilize DATS for our assessment; not only will we assess compliance with DATS itself, we will go further with several additional 'optional' parts of DATS including ontological term verification and other sanity checks.

While there are several ways one can go about making an assessment, one way is to construct the rubric and metrics metadata while you construct the code to assert that metric.

```
rubric = {
  '@id': 25, # ID in FAIRshake
  'name': 'NIH CFDE Interoperability',
  'description': 'This rubric identifies aspects of the metadata models which promote␣
↪interoperable dataset querying and filtering',
  'metrics': {},
}

def metric(schema):
  ''' A python decorator for registering a metric for the rubric. Usage:
  @metric({
    '@id': unique_id,
    'metric': 'metadata'
  })
  def _(asset):
    yield { 'value': 1.0, 'comment': 'Success' }
  '''
  global rubric
  def wrapper(func):
    rubric['metrics'][schema['@id']] = dict(schema, func=func)
  setattr(wrapper, '__name__', schema['name'])
  return wrapper

def assess(rubric, doc):
  ''' How to use use this rubric for assessing a document. Usage:
  assess(rubric, { "your": "metadata" })
  '''
  assessment = {
    '@type': 'Assessment',
    'target': doc,
    'rubric': rubric['@id'],
    'answers': []
  }
  # print(assessment)
  for metric in rubric['metrics'].values():
    # print('Checking {}...'.format(metric['name']))
    for answer in metric['func'](doc):
      # print(' => {}'.format(answer))
      assessment['answers'].append({
        'metric': { k: v for k, v in metric.items() if k != 'func' },
        'answer': answer,
      })
  return assessment
```

With these functions setup, all we have left is to define the metrics and their metadata, then the assess function can operate on a given document. Let's write a metric for assessing DATS:

```
@metric({
  '@id': 107, # ID in FAIRshake
  'name': 'DATS',
  'description': 'The metadata properly conforms with the DATS metadata specification
 ↪',
  'principle': 'Findable',
})
def _(doc):
  from jsonschema import Draft4Validator
  errors = list(Draft4Validator({'$ref': 'http://w3id.org/dats/schema/dataset_schema.
 ↪json'}).iter_errors(doc))
  yield {
    'value': max(1 - (len(errors) / 100), 0),
    'comment': 'DATS JSON-Schema Validation results in {} error(s)\n{}'.format(
      len(errors) if errors else 'no',
      '\n'.join(map(str, errors))
    ).strip(),
  }

# ... additional metrics ...
```

With this added metric, which uses jsonschema to validate the conformance of the metadata document to the DATS metadata model, an assessment would now produce answers for this specific metric. We have normalized the answers between 0 and 1, you get a 1 for full conformance or a 0 for >= 100 validation errors. It is important to note that this is not the complete picture, perhaps you have a field for a landing page, but that website is down!

```
@metric({
  '@id': 16, # ID in FAIRshake
  'name': 'Landing Page',
  'description': 'A landing page exists and is accessible',
  'principle': 'Findable',
})
def _(doc):
  landingPages = set(
    node['access']['landingPage']
    for node in jsonld_frame(doc, {
      '@type': 'DatasetDistribution',
      'access': {
        'landingPage': {},
      }
    })['@graph']
    if node['access'] and node['access']['landingPage']
  )
  if landingPages:
    for landingPage in landingPages:
      if requests.get(landingPage).status_code < 400:
        yield {
          'value': 1,
          'comment': 'Landing page found {} and seems to be accessible'.
 ↪format(landingPage)
        }
      else:
        yield {
          'value': 0.75,
          'comment': 'Landing page found {} but seems to report a problem'.
 ↪format(landingPage)
```

(continues on next page)

(continued from previous page)

```
        }
  else:
    yield {
      'value': 0,
      'comment': 'Could not identify any landing pages'
    }
```

Above we have an example which uses jsonld framing to find landing pages, for each of those landing pages we attempt to load the page and expect to get a reasonable http status code (a value less than 400, 200-299 for success, or 300-399 for redirects). This could be improved further to be more stringent (ensure we can find the title of our document on the landing page or something along those lines), but even this basic loose criterion is not always satisfied.

Ultimately, this can become a command line application that we run in parallel on lots of DATS metadata. You can refer to the scripts *here* for examples on how you can accomplish this. It is also possible to resolve additional metadata in the process of the assessment through forward chaining or other methods, an example of an assessment like that is also *in that repository*: `data_citation_assessment.py` which uses a url to negotiate and resolve microdata according to this Data citation paper's guidelines.

Publishing codified FAIRshake metrics and resolvers for assessment reproducibility

It is useful for reproducibility purposes, but also for reusability purposes for automated FAIR assessment code to be shared publicly. To that end, a repository for storing that code and its association with the FAIRshake metrics was developed and can be found here. This catalog and the code in it can also be used to perform future FAIR assessments that use the same metrics, rubrics, or resolvers. Pull requests are welcome, but existing automated mechanisms can immediately be used by installing the package and using some of the core functions. Performing this assessment with that repository works like so:

```
#!/bin/python
# assumption: DATS objects are generated line by line
# usage: assess.py < input_dats.txt > output_assessments.txt

import sys, json
from fairshake_assessments.core import assess_many_async
from fairshake_assessments.rubrics.rubric_36_nih_cfde import rubric_36_nih_cfde

for assessment in assess_many_async(map(json.loads, sys.stdin)):
  print(json.dumps(assessment))
```

Note that other rubrics, metrics, and resolvers (e.g. ways of finding DATS from a `url`) are available in the `fairshake-assessments` and are associated with some of the FAIRshake metrics.

Registering assessments on FAIRshake

Now that we have performed our assessment, we should publish these results on FAIRshake for us and the world to see where improvements can be made. It is important to note that the assessment results are a function of all parties (the digital object, the standard, the underlying repository or system that serves the digital object) and as such must be compared relative to the same baseline.

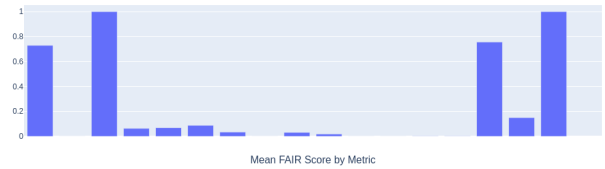
The fairshake-assessments library can also help with this.

```
#!/bin/python
# assumption: assessment objects formatted according to the FAIRshake API in a .jsonl␣
↪file
# usage: API_KEY='' register.py < assessments.jsonl
# see https://fairshake.cloud/accounts/api_access/ for an API_KEY

import sys
import json
from fairshake_assessments.core import (
  get_fairshake_client,
  find_or_create_fairshake_digital_object,
  publish_fairshake_assessment,
)

project = 87 # project id on fairshake

fairshake = get_fairshake_client(api_key=os.environ['API_KEY'])
for assessment in map(json.loads, sys.stdin):
  target = find_or_create_fairshake_digital_object(fairshake=fairshake, **assessment[
↪'target'])
  publish_fairshake_assessment(
    fairshake=fairshake,
    project=project,
    **assessment
  )
```

Reviewing the state of FAIRness in a project

Once an assessment has been published on FAIRshake, it becomes possible to browse those assessments both via a high level summary page and via a more granular tabular view.

The rubric we used for the CFDE is available from here. It includes most of the universal FAIR metrics, but also some metrics that address specific CFDE use-cases such as 'A relevant file type is present and resolvable with EDAM'. This rubric was used to assess the metadata produced by the CFDE for several DCCs as part of this project, you can also see statistics for those assessments there.



Each bar is hoverable on FAIRshake indicating which metric is receiving that which particular score. The score itself is quantified between 0 and 1 much like the scores used to color the FAIR insignia, 0 representing little to no digital objects satisfy that particular metric and 1 meaning that the majority of digital objects satisfy the metric.

7.1.6 Conclusion

In this recipe, we have detailed and described the manual and automatic process of FAIRification with FAIRshake for a CFDE case study. While the assessment described here was for the CFDE DATS serialized assets, the same process is applicable to any standard and any type of digital object. Examples exist for assessing APIs, GitHub repositories, and tools, among other case studies using standards applicable to each. As more standards become codified and accessible through FAIRshake, they will become simpler to evaluate, ultimately increasing the FAIRness of the standard itself and anything using that standard. It should be noted that the process of using FAIRshake for performing assessments is mainly designed to increase awareness about standards that digital object producers can apply to improve the FAIRness of the digital assets they produce and publish.

7.1.7 References

FAIR Repo

The CFDE FAIR repository is currently private given that it contains details about DCCs that have not yet been verified. Please submit a request to https://www.nih-cfde.org/contact/ if you need access to the repository.

If you have access to the repository, you can access information in it about:

- Scripts to convert several DCC's publicly facing metadata into C2M2 compatible frictionless datapackages organized by DCC name
- Scripts to automatically assess C2M2 compatible frictionless datapackages against the C2M2 rubric on FAIRshake
- Reports showing the satisfaction of the converted DCC metadata with the C2M2 rubric over time (FAIR assessments over time)

7.2 Evaluating FAIRness with the CFDE Rubric on FAIRshake

Author(s): Steve Edwards, John Cheadle

Maintainer(s): Steve Edwards, John Cheadle

Version: 1.0

License: GPLv2+

7.2.1 FAIRshake Rubric for CFDE

All projects that are part of the Common Fund Data Ecosystem (CFDE) are evaluated using the CFDE FAIR Rubric, which is a set of 17 metrics that evaluate the Findability, Accessibility, Interoperability, and Reusability (FAIR) of each datasets across the different CF data coordination centers (DCCs). This Rubric is available via the FAIRshake tool and it is described in detail below. It should be noted that this Rubric is an initial draft and can be adjusted based on feedback and input from the DCCs.

NIH Common Fund Data Ecosystem (CFDE) FAIR Rubric

A rubric for evaluating the FAIRness of NIH-CFDE Resources

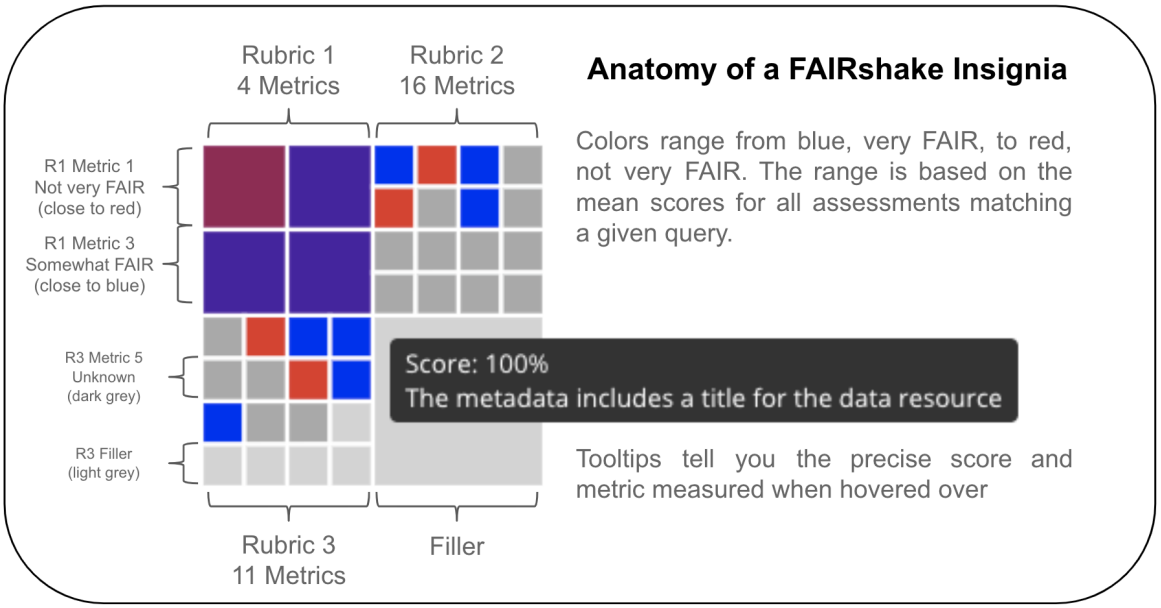
Tags: nihcommonfund

URL(s):

https://github.com/nih-cfde/FAIR

7.2.2 Understanding the FAIRshake Insignia

The FAIRshake insignia (pictured above) is a visual representation of a score given to a digital object based on a scoring rubric composed of metrics. It offers users a quick graphical view of the evaluation of the FAIR principles for that object. The number of colored boxes (non-gray) corresponds to the number of metrics in the rubric, whereas the color of the box indicates how well a metric adheres to FAIR principles (blue means full adherence and red not). Users can hover over the boxes to observe the score for each metric. The image below is from the FAIRshake documentation page.



7.2.3 FAIR Principles and Metrics Resources

The original publication that described the FAIR principles lays out very abstractly what it means to be FAIR. Later on, the same group that published the popular FAIR guidelines paper developed the concept of FAIR metrics and authored a publication about them.

FAIRshake was then developed to host FAIR evaluations. FAIRshake allows using other metrics besides those that were published in the FAIR metrics paper. A publication that describes the FAIRshake can be found on the fairsharing.org site.

Globally Unique Identifiers

Provide a URL to a registered scheme that defines the globally-unique structure of the identifier(s) for your digital resource. Examples of identifier schemes are available on the fairsharing.org site.

Principle: F1

Metric(s): FM-F1A

Rationale: The uniqueness of an identifier is a necessary condition to unambiguously refer to a resource, and that resource alone. Otherwise, an identifier shared by multiple resources will confound efforts to describe that resource, or to use the identifier to retrieve it.

URL(s):

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_F1A.pdf
- https://purl.org/fair-metrics/FM_F1A
- https://doi.org/10.25504/FAIRsharing.r49beq

Machine-Readable Metadata

Provide the URL to a document that contains machine-readable metadata for the digital resource.

Principle: F2

Metric(s): FM-F2

Rationale: Metadata plays an important role in enabling users to find a resource of interest. Metadata may be indexed to facilitate keyword searches over structured and unstructured metadata. However, only with structured metadata can an indexing system provide increased precision of combining keyword searches with restrictions on particular attributes, e.g., license, or standards used.

URL(s):

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_F2.pdf
- https://purl.org/fair-metrics/FM_F2
- https://doi.org/10.25504/FAIRsharing.ztr3n9

Standardized Metadata

The URI of a registered metadata format in FAIRSharing (for example, https://fairsharing.org/FAIRsharing.tn873z if your data follows the INSD Sequence XML format)

Principle: F4

Metric(s): FM-F4, FM-R1.3

Rationale: Having a structured metadata document is a great first step, but it should also follow a known community standard to reduce the work needed to index that metadata by search engines.

URL(s):

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_F4.pdf
- https://purl.org/fair-metrics/FM_F4
- https://doi.org/10.25504/FAIRsharing.Lcws1N
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_R1.3.pdf
- https://purl.org/fair-metrics/FM_R1.3
- https://doi.org/10.25504/FAIRsharing.cuyPH9

Resource Identifier in Metadata

The identifier that should explicitly appear in the metadata.

Principle: F3

Metric(s): FM-F3

Rationale: Metadata are intended to provide information about a digital resource. However, data and their metadata are created and published separated (they are in different files and in different formats). Since F1 specifies that metadata and data must have different identifiers, it is important that metadata contain the resource identifier, so that the resource can be exactly accessed by its identifier (A1).

URL(s):

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_F3.pdf
- https://purl.org/fair-metrics/FM_F3
- https://fairsharing.org/FAIRsharing.o8TYnW

NIH Program Name is Available for Querying

This confirms that the data resource includes the name of the CF program under which the work was performed.

Principle: F2

Metric(s): FM-F2, FM-R1.2

Rationale: There are many examples where a user may want to see all data that resulted from a specific NIH CF program.

URL(s):

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_F2.pdf
- https://purl.org/fair-metrics/FM_F2
- https://doi.org/10.25504/FAIRsharing.ztr3n9

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_R1.2.pdf
- https://purl.org/fair-metrics/FM_R1.2
- https://doi.org/10.25504/FAIRsharing.qcziIV

NIH Project Name is Available for Querying

This confirms that the data resource includes the name of the NIH project that collected the data.

Principle: F2

Metric(s): FM-F2, FM-R1.2

Rationale A user may want to find all the data from a specific project when analyzing the data of interest. This could be for discovery purposes or to identify confounding variables within the data.

URL(s):

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_F2.pdf
- https://purl.org/fair-metrics/FM_F2
- https://doi.org/10.25504/FAIRsharing.ztr3n9
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_R1.2.pdf
- https://purl.org/fair-metrics/FM_R1.2
- https://doi.org/10.25504/FAIRsharing.qcziIV

The Institution that Created this Dataset is Available

This confirms that the identity of the institution where the dataset was created is available within the metadata for the dataset.

Principle: F2

Metric(s): FM-F2, FM-R1.2

Rationale: This information can be used to find additional datasets by the creators of this dataset as well as for citation purposes when reusing the data.

URL(s):

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_F2.pdf
- https://purl.org/fair-metrics/FM_F2
- https://doi.org/10.25504/FAIRsharing.ztr3n9
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_R1.2.pdf
- https://purl.org/fair-metrics/FM_R1.2
- https://doi.org/10.25504/FAIRsharing.qcziIV

A Landing Page Exists and is Accessible

This confirms that the resource containing the dataset has a main page with information about the resource.

Principle: F

Metric(s): FM-F4, FM-A1.1

Rationale: For users to determine if the data in the resource is suitable for their purposes, a central website with information about the resource and links to the data is important.

URL(s):

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_F4.pdf
- https://purl.org/fair-metrics/FM_F4
- https://doi.org/10.25504/FAIRsharing.Lcws1N
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_A1.1.pdf
- https://purl.org/fair-metrics/FM_A1.1
- https://fairsharing.org/FAIRsharing.yDJci5

Open, Free, Standardized Access Protocol

Provide a URL to the access protocol.

Principle: A1.1

Metric(s): FM-A1.1

Rationale: Digital resources and their metadata should be retrievable through standardised communication protocols. Open, free, and standardised communication protocols reduce the cost and effort for any part to gain authorized access to a digital resource. Having a protocol that is open allows any individual to create their own standard-compliant implementation, that it is free reduces the possibility that those lacking monetary means cannot access the resource, and that it is universally implementable ensures that such technology is available to all (and not restricted, for instance by country or creed). The resource should be accessible through an open, free, and standardized communication protocol.

URL(s):

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_A1.1.pdf
- https://purl.org/fair-metrics/FM_A1.1
- https://fairsharing.org/FAIRsharing.yDJci5

A Biological Assay is Present and Resolvable in the BioAssay Ontology

Confirm that biological assays are described using a formal ontology.

Principle: I

Metric(s): FM-I1, FM-I2, FM-I3

Rationale: Interoperability requires 1: (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation; 2: (Meta)data use vocabularies that follow the FAIR principles; 3: (Meta)data include qualified references to other (meta)data. This ontology meets all of those criteria.

URL(s):

- https://www.ebi.ac.uk/ols/ontologies/bao

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I1.pdf
- https://purl.org/fair-metrics/FM_I1
- https://doi.org/10.25504/FAIRsharing.jLpL6i
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I2.pdf
- https://purl.org/fair-metrics/FM_I2
- https://doi.org/10.25504/FAIRsharing.0A9kNV
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I3.pdf
- https://purl.org/fair-metrics/FM_I3
- https://doi.org/10.25504/FAIRsharing.B2sbNh

A Relevant Anatomical Part is Present and Resolvable in the UBERON Ontology

Confirms that references to anatomical structures use a formal ontology.

Principle: I

Metric(s): FM-I1, FM-I2, FM-I3

Rationale: Interoperability requires 1: (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation; 2: (Meta)data use vocabularies that follow the FAIR principles; 3: (Meta)data include qualified references to other (meta)data. This ontology meets all of those criteria.

URL(s):

- https://www.ebi.ac.uk/ols/ontologies/uberon
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I1.pdf
- https://purl.org/fair-metrics/FM_I1
- https://doi.org/10.25504/FAIRsharing.jLpL6i
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I2.pdf
- https://purl.org/fair-metrics/FM_I2
- https://doi.org/10.25504/FAIRsharing.0A9kNV
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I3.pdf
- https://purl.org/fair-metrics/FM_I3
- https://doi.org/10.25504/FAIRsharing.B2sbNh

A Relevant Disease is Present and Resolvable in the MONDO Ontology

Confirms that references to diseases use a formal ontology.

Principle: I

Metric(s): FM-I1, FM-I2, FM-I3

Rationale: Interoperability requires 1: (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation; 2: (Meta)data use vocabularies that follow the FAIR principles; 3: (Meta)data include qualified references to other (meta)data. This ontology meets all of those criteria.

URL(s):

- https://www.ebi.ac.uk/ols/ontologies/mondo
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I1.pdf
- https://purl.org/fair-metrics/FM_I1
- https://doi.org/10.25504/FAIRsharing.jLpL6i
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I2.pdf
- https://purl.org/fair-metrics/FM_I2
- https://doi.org/10.25504/FAIRsharing.0A9kNV
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I3.pdf
- https://purl.org/fair-metrics/FM_I3
- https://doi.org/10.25504/FAIRsharing.B2sbNh

A Relevant File Type is Present and Resolvable in the EDAM Ontology

Confirms that the file types in the dataset are described using a formal ontology.

Principle: I

Metric(s): FM-I1, FM-I2, FM-I3

Rationale: Interoperability requires 1: (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation; 2: (Meta)data use vocabularies that follow the FAIR principles; 3: (Meta)data include qualified references to other (meta)data. This ontology meets all of those criteria.

URL(s):

- https://www.ebi.ac.uk/ols/ontologies/edam
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I1.pdf
- https://purl.org/fair-metrics/FM_I1
- https://doi.org/10.25504/FAIRsharing.jLpL6i
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I2.pdf
- https://purl.org/fair-metrics/FM_I2
- https://doi.org/10.25504/FAIRsharing.0A9kNV
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I3.pdf
- https://purl.org/fair-metrics/FM_I3
- https://doi.org/10.25504/FAIRsharing.B2sbNh

A Relevant Taxonomy is Present and Resolvable in the NCBITaxon Ontology

Confirms that references to the species from which data where collected are described using a formal ontology.

Principle: F

Metric(s): FM-I1, FM-I2, FM-I3

Rationale: Interoperability requires 1: (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation; 2: (Meta)data use vocabularies that follow the FAIR principles; 3: (Meta)data include qualified references to other (meta)data. This ontology meets all of those criteria.

URL(s):

- https://www.ebi.ac.uk/ols/ontologies/NCBITaxon
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I1.pdf
- https://purl.org/fair-metrics/FM_I1
- https://doi.org/10.25504/FAIRsharing.jLpL6i
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I2.pdf
- https://purl.org/fair-metrics/FM_I2
- https://doi.org/10.25504/FAIRsharing.0A9kNV
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I3.pdf
- https://purl.org/fair-metrics/FM_I3
- https://doi.org/10.25504/FAIRsharing.B2sbNh

A Relevant Cell Line is Present and Resolvable in the Cellosaurus Ontology

Confirms that cell lines, for experiments performed on immortalized cell lines, are described using a formal ontology.

Principle: I

Metric(s): FM-I1, FM-I2, FM-I3

Rationale: Interoperability requires 1: (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation; 2: (Meta)data use vocabularies that follow the FAIR principles; 3: (Meta)data include qualified references to other (meta)data. This ontology meets all of those criteria.

URL(s):

- https://web.expasy.org/cellosaurus/
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I1.pdf
- https://purl.org/fair-metrics/FM_I1
- https://doi.org/10.25504/FAIRsharing.jLpL6i
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I2.pdf
- https://purl.org/fair-metrics/FM_I2
- https://doi.org/10.25504/FAIRsharing.0A9kNV
- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_I3.pdf
- https://purl.org/fair-metrics/FM_I3
- https://doi.org/10.25504/FAIRsharing.B2sbNh

Contact Information is Provided for the Creator(s) of the Dataset.

Contact information (typically name and email address) for the creator(s) of the dataset.

Principle: R

Metric(s): FM-R1.2

Rationale: It is important to identify the creators of the data set as part of defining provenance of the data (who/what/when produced the data). This informs users of who should get credit if this dataset is used in other contexts.

URL(s):

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_R1.2.pdf
- https://purl.org/fair-metrics/FM_R1.2
- https://doi.org/10.25504/FAIRsharing.qcziIV

Digital Resource License

Provide a URL to the license that governs the use of the digital resource.

Principle: R

Metric(s): FM-R1.1

Rationale: Both digital resources and their metadata must be licensed (or equivalent e.g. terms of use, smart contract). The lack of a license indicates that no rights to reuse are granted, thereby deterring lawful use. Note that the combination of resources with restrictive license conditions may lead to adverse effects, and ultimately preclude their combined use. To satisfy this, two URLs must be provided -> one for the metadata and one for the digital resource.

URL(s):

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_R1.1.pdf
- https://purl.org/fair-metrics/FM_R1.1
- https://doi.org/10.25504/FAIRsharing.fsB7NK

Metadata License

Provide a URL to the license that governs the use of the digital resource.

Principle: R

Metric(s): FM-R1.1

Rationale: Both digital resources and their metadata must be licensed (or equivalent e.g. terms of use, smart contract). The lack of a license indicates that no rights to reuse are granted, thereby deterring lawful use. Note that the combination of resources with restrictive license conditions may lead to adverse effects, and ultimately preclude their combined use.

URL(s):

- https://github.com/FAIRMetrics/Metrics/blob/master/Distributions/FM_R1.1.pdf
- https://purl.org/fair-metrics/FM_R1.1
- https://doi.org/10.25504/FAIRsharing.fsB7NK

7.2.4 Rubric Metrics Summary Table

https://doi.org/10.25504/FAIRsharing.cuyPH9

https://purl.org/fair-metrics/FM_R1.3

https://raw.githubusercontent.com/FAIRMetrics/Metrics/master/FM_A1.1

7.3 Developing FAIR API for the Web

Authors: Daniel J. B. Clarke

Maintainers: Daniel J. B. Clarke

Version: 1.1

License: CC0 1.0 Universal (CC0 1.0) Public Domain Dedication

---

7.3.1 Background

An Application Programming Interface (API) refers to a mechanism for interfacing with a web service programmatically. Unlike a Graphical User Interface (GUI) designed to be used by an end-user, API are designed to be used by other computer programs. Designing and documenting an API for an application enables your application to be more interoperable with, and ultimately more reused, by other applications. Depending on the type of application, there are different ways to design APIs. Most important is that APIs should be documented well. We will not cover FAIR APIs of software libraries, but instead focus on developing FAIR APIs for the web.

More and more web-based applications are becoming available every day. These applications typically perform complex operations on large databases. While web-based applications provide users with the capacity to access a tool, a database, or other resource programmatically, they are not always able to interoperate with other independent web applications. A web-based application that offers a FAIR API is more accessible to operating as part of workflows, or integration systems such as semantic search engines. This makes FAIR API development very relevant for data catalogs or web tools developed by the CF DCCs.

While a slew of standards exist for web API development and documentation, each has their own level of FAIRness. Here we are going to focus on RESTful APIs, which can be described with OpenAPI (previously Swagger) to take advantage of RESTful API flexibility while still permitting machine readable introspection. Several other standards are machine readable by default, including SOAP, SPARQL or GraphQL among many others, but despite this, RESTful APIs are the most widely used because of their low barrier to entry. Some standards exist for RESTful APIs, in many cases, these can also be described by OpenAPI. We will consider a specific extension of OpenAPI: Smart-API, which adds a few additional fields and also has its own get-started guide.

7.3.2 Motivation

Documenting APIs or building them from the ground up with SmartAPI in mind have a number of advantages:

- Human readable documentation of that API with a number of packages that can generate it from the OpenAPI schema
- Server/client libraries from a number of packages that can generate them for numerous programming languages based on the OpenAPI schema
  - People can access your application features using their favorite programming language

    - People can create an application that shares the same API as another application for interoperability
- Interoperability with GraphQL
- Enabling simple use cases like enhancing findability with API Catalogs
- Enabling future use cases

SmartAPI specifications inherit all of the benefits of OpenAPI, while adding the potential for interoperability with RDF semantically linked data. This can help enable future use cases like BioThings API, powering semantically linked APIs for biomedical knowledge exploration.

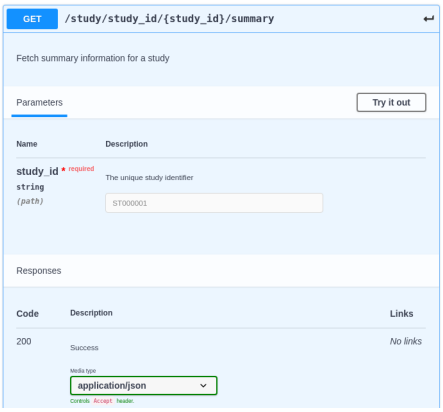
7.3.3 Ingredients

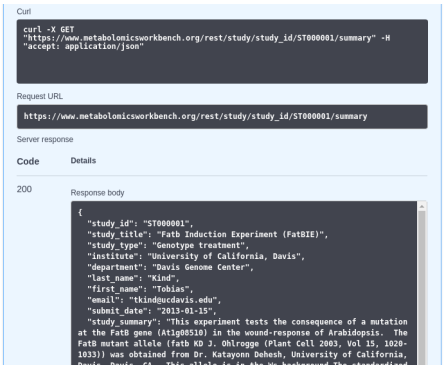
- Web Application
- Existing API Documentation
- OpenAPI/SmartAPI Editor (see step 1)

7.3.4 Objectives

We will look at the existing REST service provided by the Metabolomics Workbench catalog: https://www.metabolomicsworkbench.org/tools/mw_rest.php. This API is described for human consumption, including examples for each endpoint. We will tackle some of the endpoints using OpenAPI.

Although OpenAPI can be edited by most standards editors because it is typically written in YAML (a slightly 'nicer' version of JSON that is equivalent), it is helpful to use an OpenAPI editor like https://app.swaggerhub.com/home. This will catch errors as you edit, and permit testing of the endpoints as you encode immediately.

An example endpoint in an OpenAPI Editor: 

A real response in an OpenAPI Editor: 

7.3.5 Recipe

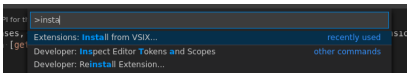
The complete `swagger.yaml` constructed in this recipe is available here for your reference, it will be valuable to follow the tutorial and construct it iteratively.

Step 1: Setting up the OpenAPI Editor

Several options exist, including the Swagger Editor, especially with APIs that are enabled to support CORS. Unfortunately, the API we will work with here **does not**, so we will need to obtain a Swagger Editor that can operate even when CORS is not enabled. Because the de-facto swagger editor is a web-app, most editors have this issue. We specifically modified an Open Source Visual Studio Code Extension so that it supports this specific use case.

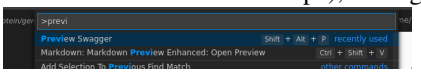
Until our pull request is merged, the modified extension can be accessed here. The `vsix` file can be installed with Visual Studio Code.

It can be installed from `Ctrl+Shift+P` with the action "Install from VSIX"

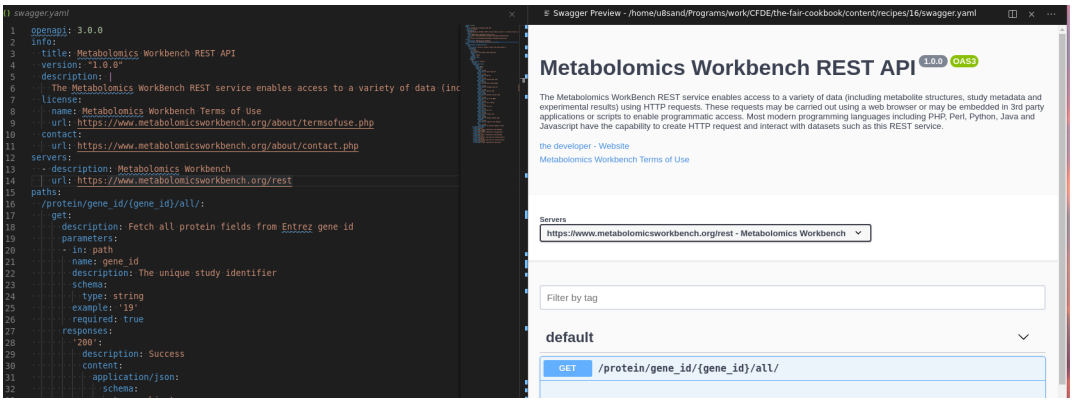


And selecting the `.vsix` file you downloaded.

Once installed, a swagger file can be edited by opening the swagger.yaml (that we will be writing throughout the rest of the recipe), using `Ctrl+Shift+P` again and choosing the action "Preview Swagger"



The result will be a webview that opens to the side with the Swagger Editor.



Edits to the file will update the view **in real time**, and the view may be used to craft/test API requests.

Step 2: Beginning an OpenAPI specification

We start by annotating useful descriptions for the API in the `info` field, this includes adding descriptors, version, license, and contact information. This gives your API an identity and this way it can be used in OpenAPI catalogs, for which there are several including SmartAPI, making it possible to find your own APIs.

The `servers` field has the base url(s) for accessing the API we're about to describe.

```yaml
openapi: 3.0.0
info:
  title: Metabolomics Workbench REST API
  version: "1.0.0"
  description: |
```

(continues on next page)

(continued from previous page)

```
    The Metabolomics WorkBench REST service enables access to a variety of data␣
 ↪(including metabolite structures, study metadata and experimental results) using␣
 ↪HTTP requests. These requests may be carried out using a web browser or may be␣
 ↪embedded in 3rd party applications or scripts to enable programmatic access. Most␣
 ↪modern programming languages including PHP, Perl, Python, Java and Javascript have␣
 ↪the capability to create HTTP request and interact with datasets such as this REST␣
 ↪service.
  license:
    name: Metabolomics Workbench Terms of Use
    url: https://www.metabolomicsworkbench.org/about/termsofuse.php
  contact:
    url: https://www.metabolomicsworkbench.org/about/contact.php
servers:
  - description: Metabolomics Workbench
    url: https://www.metabolomicsworkbench.org/rest
```

Step 3: Describing a path

The Metabolomics API offers several examples, let's tackle one of them:

Example request	Example URL
Show all publicly available studies (Project, Study, Analysis ID)	https://www.metabolomicsworkbench.org/rest/study/study_id/ST/available

This tells us what the endpoint does to an extent; let's add that to our API under the `paths`.

```
paths:
  /study/study_id/ST/available:
    get:
      description: Fetch summary information for all studies
```

The path is relative to the server url, and `get` refers to the REST method (`GET` as opposed to `POST`, `PUT`, `DELETE`, …), in REST `GET` refers to reading a resource and is what happens when you send the following packet to a web server. These packets can be crafted using `curl`, the `-v` flag helps see input and output packets and the `-X` flag allows you to set the method (`GET`, `POST`, …), the `-H` flag lets you specify headers.

```
curl -v -X GET -H 'Content-Type: application/json' https://www.metabolomicsworkbench.
 ↪org/rest/study/study_id/ST/available
```

```
...
> GET /rest/study/study_id/ST/available HTTP/1.1 # PATH HERE
> Host: www.metabolomicsworkbench.org            # REQUEST HEADERS HERE
> User-Agent: curl/7.70.0
> Accept: */*
> Content-Type: application/json
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK                              # RESPONSE STATUS CODE HERE
< Date: Wed, 27 May 2020 14:29:27 GMT          # RESPONSE HEADERS HERE
< Server: Apache/2.4.6 (CentOS)
< X-Frame-Options: SAMEORIGIN
< Vary: Accept-Encoding
```

(continues on next page)

(continued from previous page)

```
< X-XSS-Protection: 1; mode=block
< Transfer-Encoding: chunked
< Content-Type: application/json
<                                                    # BODY HERE
{
  "1": {
    "project_id":"PR000001",
    "study_id":"ST000001",
    "analysis_id":"AN000001"
  },
  "2":{
    "project_id":"PR000002",
    "study_id":"ST000002",
    "analysis_id":"AN000002"
  },
  ...
  "1783":{
    "project_id":"PR000928",
    "study_id":"ST001364",
    "analysis_id":"AN002271"
  }
}
...
```

Note that your web browser does the same, albeit with a few different headers for end-to-end compression and browser information for webpage optimization.

```
> GET https://www.metabolomicsworkbench.org/rest/study/study_id/ST/available HTTP/1.1
> Host: www.metabolomicsworkbench.org
> Content-Type: text/html
> User-Agent: Mozilla/5.0 ...
> Accept-language: en-US,en;q=0.5
> Accept-Encoding: gzip, deflate
```

The `GET` at the start is changed to `POST` or another value when sending actual data (in the body of the packet). (`curl -v -X POST ... -d "packet data"`)

We did not know what the response would be by the webpage, but OpenAPI provides a means to describe this as well.

```
paths:
  /study/study_id/ST/available:
    get:
      description: Fetch summary information for all studies
      responses:
        '200':
          description: Success
          content:
            application/json:
              schema:
                type: object
                additionalProperties:
                  type: object
                  properties:
                    project_id:
                      type: string
                    study_id:
```

(continues on next page)

(continued from previous page)

```
                    type: string
                  analysis_id:
                    type: string
```

The `200` here refers to the HTTP status code, these are standardized by HTTP but the gist is:

Code	Meaning
2xx	OK (created, no content, …)
3xx	Redirect (temporary, permanent, …)
4xx	Not OK (not found, permission denied, …)
5xx	Server Error

We can explicitly describe what the response means for our application in the `description` under the response code. The `content -> application/json` refers to the `Content-Type` returned on the response header. This header tells you that the body will be json, and not i.e. `text/html` which we view most webpages as.

Finally, the `schema` is JSON-Schema describing how the JSON should look. It also supports describing each field individually in depth, and specifying optional or mandatory fields. We use `additionalProperties` to refer to the `values` in the object, and `properties` on a `type: object` to refer to the `key` specific description.

The benefit of fully describing and endpoint like this is that a developer can fully understand what to expect from an endpoint, enabling them to determine code logic validity prior to having to test it at runtime. Furthermore, the SmartAPI initiative has an extension for relating those types to RDF for visions like the BioThings.

Step 4: Adding a path with parameters

Let's tackle our next endpoint:

Example request	Example URL
Fetch analysis information for a study	https://www.metabolomicsworkbench.org/rest/study/study_id/ST000001/analysis

While the example shows `ST000001` in reality, the idea is that this can be any study ID, such as those coming out of the previous endpoint.

```
paths:
  ...
  /study/study_id/{study_id}/summary:
    get:
      description: Fetch summary information for a study
      parameters:
      - in: path
        name: study_id
        description: The unique study identifier
        schema:
          type: string
          pattern: '^ST\d*$'
        example: ST000001
        required: true
      responses:
        '200':
          description: Success
```

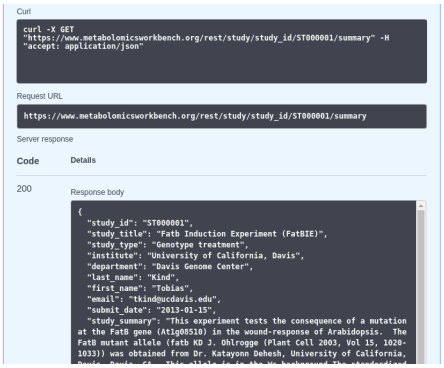
We see some new concepts here, the first is the path which has a variable in it delineated by `{study_id}` where we would want the `study_id` to go. Paired with this we add an entry into the `parameters` array and state the same name `study_id` is `in: path` referring to this path-style variable substitution.

Parameters can be added in other places as well. Google, for instance, describes searches like so: `https://www.google.com/search?q=smartapi&sourceid=chrome&ie=UTF-8`, the `?q=...&sourceid=...` are referred to as query parameters and they are in the form: `?{name}={value}&{name}={value}&...`. You could describe these in OpenAPI as well by using `in: query`.

In the case of a `POST` you might use, `in: body` referring to the content you send with your request.

Again, every parameters is validatable with JSONSchema. In our current example, we have used a JSONSchema pattern to constrain the type of string acceptable by the endpoint. In the case of a body, it could be an elaborate JSON object such as our response schema.

Finally, we have included an `example` which will help developers with rapid testing of endpoints given valid examples. Using this example, we can use our OpenAPI Editor to trigger a new request:



With the output, we can complete our path by annotating the response:

```yaml
paths:
  ...
  /study/study_id/{study_id}/summary:
    get:
      ...
      responses:
        '200':
          description: Success
          content:
            application/json:
              schema:
                type: array
                items:
                  type: object
                  properties:
                    study_id:
                      type: string
                    study_title:
                      type: string
                    study_type:
                      type: string
                    institute:
                      type: string
                    department:
                      type: string
                    last_name:
```

(continues on next page)

(continued from previous page)

```
                    type: string
                  first_name:
                    type: string
                  email:
                    type: string
                  submit_date:
                    type: string
                  study_summary:
                    type: string
                  subject_species:
                    type: string
```

Step 5: Identifying components

In some cases, a common set of JSON objects are reused throughout the API. It is often meaningful to turn these into their own 'component' and reference them. It is also extremely helpful to include descriptions especially for fields that are not directly interpretable.

```
paths:
  ...
  /study/study_id/{study_id}/summary:
    get:
      ...
      responses:
        '200':
          description: Success
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/StudySummary'
  ...
components:
  schemas:
    StudySummary:
      description: Summary information about a study
      type: object
      properties:
        study_id:
          type: string
          description: A unique identifier for this study
        study_title:
          type: string
        study_type:
          type: string
          description: The type of treatment used in the study
        institute:
          type: string
          description: The institution that performed the study
        department:
          type: string
          description: The department in the institute that performed the study
        last_name:
          type: string
          description: The last name of the PI responsible for the study
```

(continues on next page)

(continued from previous page)

```
        first_name:
          type: string
          description: The first name of the PI responsible for the study
        email:
          type: string
          description: The email to contact for information about the study
        submit_date:
          type: string
          description: The date this study was submitted to metabolomics workbench
        study_summary:
          type: string
          description: A detailed summary describing the study
        subject_species:
          type: string
          description: The species of the subject of the study
```

Under `components`, as many individual components can be specified, and they can be referenced using `$ref` with JSON-Schema pointers as shown above.

Step 6: SmartAPI extension

When it comes to *automatic* interoperability, the SmartAPI extension to OpenAPI is almost essential. It provides mechanisms for adding RDF annotations to parameters or responses. We will demonstrate it on a new endpoint:

Example request	Example URL
Fetch all protein fields from Entrez gene id	https://www.metabolomicsworkbench.org/rest/protein/gene_id/19/all/

```
paths:
  /protein/gene_id/{gene_id}/all/:
    get:
      description: Fetch all protein fields from Entrez gene id
      parameters:
      - in: path
        name: gene_id
        description: The unique study identifier
        schema:
          type: string
        example: '19'
        required: true
      responses:
        '200':
          description: Success
          content:
            application/json:
              schema:
                type: object
                properties:
                  gene_id:
                    type: string
                    description: Entrez Gene ID
                  mgp_id:
                    type: string
                    description: MGP ID
```

(continues on next page)

(continued from previous page)

```
                gene_name:
                  type: string
                  description: Verbose gene name
                gene_symbol:
                  type: string
                  description: Entrez Gene Symbol
                taxid:
                  type: string
                  description: Taxonomy taxon ID
                species:
                  type: string
                  description: Species name
                species_long:
                  type: string
                  description: Species official name
                mrna_id:
                  type: string
                  description: ID of the MRNA
                refseq_id:
                  type: string
                  description: ID on refseq
                protein_gi:
                  type: string
                  description: ID on GI
                uniprot_id:
                  type: string
                  description: ID on GI
                protein_entry:
                  type: string
                  description: Protein term
                protein_name:
                  type: string
                  description: Verbose protein name
                seqlength:
                  type: string
                  description: Length of the sequence
                seq:
                  type: string
                  description: The protein sequence itself
        x-responseValueType:
        - x-path: gene_id
          x-valueType: https://identifiers.org/ncbigene
        - x-path: gene_symbol
          x-valueType: https://identifiers.org/genecards
        - x-path: tax_id
          x-valueType: https://identifiers.org/taxonomy
        - x-path: mrna_id
          x-valueType: https://www.ncbi.nlm.nih.gov/nuccore
        - x-path: refseq_id
          x-valueType: https://www.ncbi.nlm.nih.gov/protein
        - x-path: uniprot_id
          x-valueType: https://identifiers.org/uniprot
```

Here we see our usual path setup with a new section: `x-responseValueType`, this is the smartAPI extension. Each `x-path` refers to the path in the JSON object (using `.` on nested keys). For instance, the `gene_id` does not need any `.` because it is the root of the response object. The `x-valueType` here identifies the id namespace or context for which that value has meaning. Typically, this is a prefix path, in other words, you would produce a full URI with:

`{x-valueType}/{actual_value}`.

identifiers.org is a public resource cataloging actual identifier schemes making it an ideal way to namespace a given identifier. It has the added benefit of providing an API for accessing additional metadata such as cached schema.org annotations on the landing page of the resulting identifier.

With the annotations fully described here, it becomes possible to eventually utilize your API for federated RDF queries without any additional effort. This was demonstrated by the BioThings, which can integrate SmartAPI APIs with proper annotations. It also permits end users to find your APIs knowing their identifiers (i.e. ncbigenes).

Step 7: Publishing and utilizing your OpenAPI/SmartAPI

Once you have a working OpenAPI document, this open up numerous possibilities that you can now do. Firstly, your API can be published on smart-api.info, permitting people and machines to locate it and potentially utilize it.

But you can also produce interactive documentation much like the output seen in the OpenAPI editor to publish on your webpage.

It is even possible to automatically generate statically or dynamically code in many different programming languages for API clients or Server stubs (i.e. for API first or compatibility) with the openapi-generator.

An initiative by IBM provides a mechanism for interoperating an OpenAPI documented endpoint with GraphQL.

These various capabilities make an API extremely accessible, lowering many barriers to entry for interoperability and reusability of your API.

7.3.6 Conclusion

We've walked you through a case study where we documented an API using OpenAPI and SmartAPI extensions to build a document that will vastly improve the FAIRness of your API. After registering your API on an API catalog platform, such as smart-api.info you will enable developers and programs to find, introspect and interoperate with our API. Furthermore, existing tooling around OpenAPI can enable the accessibility and reusability of your APIs in many programming languages and other standardized systems.

Part III

Community

CHAPTER

EIGHT

FAIR COOKBOOK GOVERNANCE

The purpose of this document is to formalize the governance process that the FAIR Cookbook project will be relying on. This document clarifies how decisions are made and how the various elements of our community interact, including the relationship between open source collaborative development and work that may be funded by for-profit or non-profit entities.

CHAPTER

NINE

CONTRIBUTING TO THE NIH-CFDE PUBLISHED-DOCUMENTATION REPOSITORY

Hello, and thank you for wanting to contribute to the CFDE published-documentation Repository!

By contributing to this repository, you agree:

1. To obey the *Code of Conduct*
2. To release all your contributions under the same terms as the license itself: the *Creative Commons Zero* (aka Public Domain) license

If you are OK with these two conditions, then we welcome both you and your contribution!

If you have any questions about contributing, please open an issue and we will lend a hand ASAP.

Thank you for being here and for being a part of the CFDE project.

CHAPTER

TEN

LICENSE

CHAPTER

ELEVEN

DEDICATED TO THE PUBLIC DOMAIN

CHAPTER

TWELVE

CC0 UNIVERSAL PUBLIC DOMAIN DEDICATION LICENSE

12.1 Statement of Purpose

The laws of most jurisdictions throughout the world automatically confer exclusive Copyright and Related Rights (defined below) upon the creator and subsequent owner(s) (each and all, an "owner") of an original work of authorship and/or a database (each, a "Work").

Certain owners wish to permanently relinquish those rights to a Work for the purpose of contributing to a commons of creative, cultural and scientific works ("Commons") that the public can reliably and without fear of later claims of infringement build upon, modify, incorporate in other works, reuse and redistribute as freely as possible in any form whatsoever and for any purposes, including without limitation commercial purposes. These owners may contribute to the Commons to promote the ideal of a free culture and the further production of creative, cultural and scientific works, or to gain reputation or greater distribution for their Work in part through the use and efforts of others.

For these and/or other purposes and motivations, and without any expectation of additional consideration or compensation, the person associating CC0 with a Work (the "Affirmer"), to the extent that he or she is an owner of Copyright and Related Rights in the Work, voluntarily elects to apply CC0 to the Work and publicly distribute the Work under its terms, with knowledge of his or her Copyright and Related Rights in the Work and the meaning and intended legal effect of CC0 on those rights.

1. **Copyright and Related Rights.** A Work made available under CC0 may be protected by copyright and related or neighboring rights ("Copyright and Related Rights"). Copyright and Related Rights include, but are not limited to, the following:

   i. the right to reproduce, adapt, distribute, perform, display, communicate, and translate a Work;

   ii. moral rights retained by the original author(s) and/or performer(s);

   iii. publicity and privacy rights pertaining to a person's image or likeness depicted in a Work;

   iv. rights protecting against unfair competition in regards to a Work, subject to the limitations in paragraph 4(a), below;

   v. rights protecting the extraction, dissemination, use and reuse of data in a Work;

   vi. database rights (such as those arising under Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, and under any national implementation thereof, including any amended or successor version of such directive); and

   vii. other similar, equivalent or corresponding rights throughout the world based on applicable law or treaty, and any national implementations thereof.

2. **Waiver.** To the greatest extent permitted by, but not in contravention of, applicable law, Affirmer hereby overtly, fully, permanently, irrevocably and unconditionally waives, abandons, and surrenders all of Affirmer's Copyright and Related Rights and associated claims and causes of action, whether now known or unknown (including existing as well as future claims and causes of action), in the Work (i) in all territories worldwide, (ii) for the maximum duration

provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "Waiver"). Affirmer makes the Waiver for the benefit of each member of the public at large and to the detriment of Affirmer's heirs and successors, fully intending that such Waiver shall not be subject to revocation, rescission, cancellation, termination, or any other legal or equitable action to disrupt the quiet enjoyment of the Work by the public as contemplated by Affirmer's express Statement of Purpose.

3. **Public License Fallback.** Should any part of the Waiver for any reason be judged legally invalid or ineffective under applicable law, then the Waiver shall be preserved to the maximum extent permitted taking into account Affirmer's express Statement of Purpose. In addition, to the extent the Waiver is so judged Affirmer hereby grants to each affected person a royalty-free, non transferable, non sublicensable, non exclusive, irrevocable and unconditional license to exercise Affirmer's Copyright and Related Rights in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "License"). The License shall be deemed effective as of the date CC0 was applied by Affirmer to the Work. Should any part of the License for any reason be judged legally invalid or ineffective under applicable law, such partial invalidity or ineffectiveness shall not invalidate the remainder of the License, and in such case Affirmer hereby affirms that he or she will not (i) exercise any of his or her remaining Copyright and Related Rights in the Work or (ii) assert any associated claims and causes of action with respect to the Work, in either case contrary to Affirmer's express Statement of Purpose.

4. **Limitations and Disclaimers.**

   a. No trademark or patent rights held by Affirmer are waived, abandoned, surrendered, licensed or otherwise affected by this document.

   b. Affirmer offers the Work as-is and makes no representations or warranties of any kind concerning the Work, express, implied, statutory or otherwise, including without limitation warranties of title, merchantability, fitness for a particular purpose, non infringement, or the absence of latent or other defects, accuracy, or the present or absence of errors, whether or not discoverable, all to the greatest extent permissible under applicable law.

   c. Affirmer disclaims responsibility for clearing rights of other persons that may apply to the Work or any use thereof, including without limitation any person's Copyright and Related Rights in the Work. Further, Affirmer disclaims responsibility for obtaining any necessary consents, permissions or other rights required for any use of the Work.

   d. Affirmer understands and acknowledges that Creative Commons is not a party to this document and has no duty or obligation with respect to this CC0 or use of the Work.

CHAPTER

THIRTEEN

DISCLAIMER

This communication reflects the views of the authors and neither NIH or any Associated Partners are liable for any use that may be made of the information contained herein.

CHAPTER

FOURTEEN

FAIR COOKBOOK CODE OF CONDUCT

14.1 CFDE Code of Conduct

All members of the CFDE are expected to agree with the following code of conduct. We will enforce this code as needed. We expect cooperation from all members to help ensuring a safe environment for everybody.

14.2 The Quick Version

The Consortium is dedicated to providing a harassment-free experience for everyone, regardless of gender, gender identity and expression, age, sexual orientation, disability, physical appearance, body size, race, or religion (or lack thereof). We do not tolerate harassment of Consortium members in any form. Sexual language and imagery is generally not appropriate for any venue, including meetings, presentations, or discussions.

14.3 The Less Quick Version

Harassment includes offensive verbal comments related to gender, gender identity and expression, age, sexual orientation, disability, physical appearance, body size, race, religion, sexual images in public spaces, deliberate intimidation, stalking, following, harassing photography or recording, sustained disruption of talks or other events, inappropriate physical contact, and unwelcome sexual attention.

Members asked to stop any harassing behavior are expected to comply immediately.

If you are being harassed, notice that someone else is being harassed, or have any other concerns, please contact our Code of Conduct Team immediately. If you would prefer to remain anonymous, you can email that address using a free single-anonymous email service like Anonymouse

We expect members to follow these guidelines at any Consortium event.

Original source and credit: http://2012.jsconf.us/#/about & The Ada Initiative. Please help by translating or improving: http://github.com/leftlogic/confcodeofconduct.com. This work is licensed under a Creative Commons Attribution 3.0 Unported License