The Ocean InfoHub Project and the development of the ODIS-architecture

IODEPO

Aug 22, 2021

CONTENTS

Introduction

Organizations are increasingly exposing data and resources on the Web. A popular approach to this is using web architecture to expose structured data on the web using the schema.org vocabulary. Doing this makes resources discoverable by a range of organizations leveraging this architecture to build indexes. These include major commercial indexes, large domain focused groups and community focused services.

The Ocean Data and Information System (ODIS) will provide a schema.org based interoperability layer and supporting technology to allow existing and emerging ocean data and information systems, from any stakeholder, to interoperate with one another. This will enable and accelerate more effective development and dissemination of digital technology and sharing of ocean data, information, and knowledge. As such, ODIS will not be a new portal or centralised system, but will provide a collaborative solution to interlink distributed systems for common goals. Together with global project partners and partners in the three regions, a process of co-design will enable a number of global and regional nodes to test the proof of concept for the ODIS.

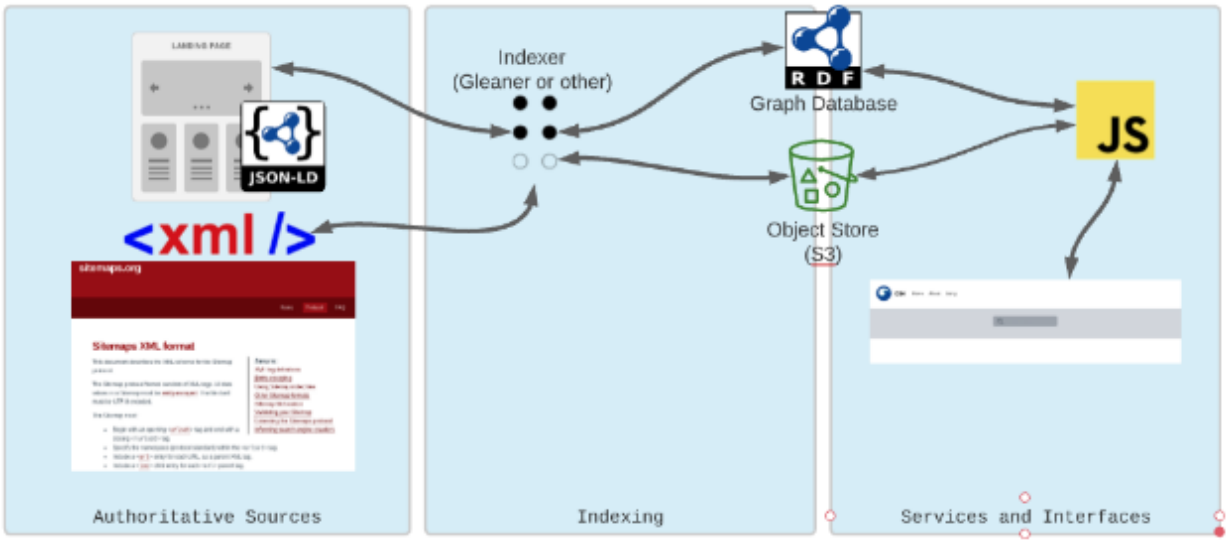
The ODIS-architecture development is being supported by the Ocean InfoHub Project, and it has been tested initially on IOC and partner databases. However, the system and standards are open for any institution or initiative that is interested in accessing the global data ecosystem to adopt and implement.

Guidance for the implementation of the ODIS-architecture

OIH is providing guidance on the various stages of such an architecture including authoring, publishing, indexing and interfaces.

The basics of this approach can be described as:

- Providers publish HTML pages for a resource. This may be a publication, course description, research instrument or other. The core themes for OIH are described in the Authoring section below.
- A HTML page then has a small JSON based snippet added to the HTML. This is described in the Including JSON-LD in your resource page in the Publishing resource below.
- If you wish a resource to be included in the OIH index, then you need to include it in a sitemap file. This is a small XML document that lists links to the resources you wish to be part of the index. This approach is shown in the sitemap.xml section of the Publishing resource.
- Once the above is done the publishing phase is over. At this point, OIH or other groups can now access and index your resources. OIH is using some existing software to index and generate the graph and expose a simple reference interface to them. This software is open and available and others are free to implement the approach with other software. Links to other software are at the repository.
- The OIH index/graph and a simple interface is current at a development site and in a later phase of OIH a production interface will be developed.



Additionally, software to aid in validating and checking the resources is under development and will be available at the repository. This will aid providers in expressing the information needed to address interfaces and services of interest to the community.

The result is a sustainable architecture to address discovery and access to various resources published by the community and a shared graph of these resources. That shared graph can be used by all members to link and discover across groups.

Key links to the OIH GitHub repository

Interested groups can review material addressing these stages at the OIH GitHub repository. Links and descriptions of these stages are described below.

Authoring Thematic Patterns

The ODIS OIH is working across five major thematic areas; Experts and Institutions, Documents, Projects, Training, Vessels. Examples of these thematic concepts are being hosted and developed with input from the community. Additionally, methods for validation and simple tooling for authoring and testing are hosted at this repository. Alongside these five thematic topics guidance on connecting services and spatial context on resources.

Publishing

Guidance on implementation the web architecture approach is also available. This includes approaches on leveraging robots.txt and sitemaps.xml file for expressing hosted resources to the net.

Indexing

The architecture approach is open and standards based. As such, many organizations will be able to leverage the authoring and publishing approaches above to index a providers resources. OIH will be providing reference implementations of

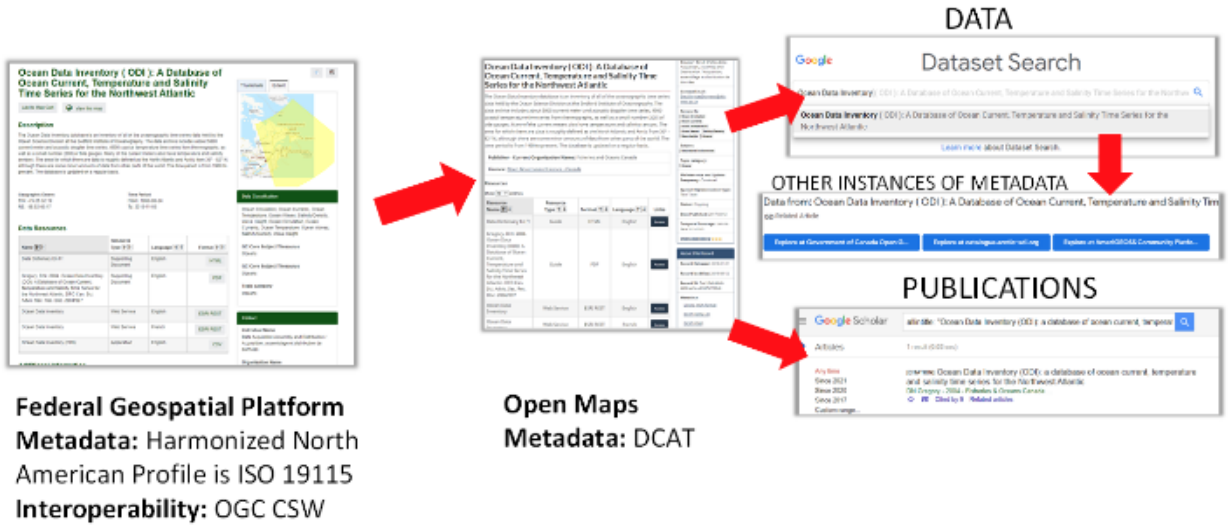
software that can generate the index.

Interfaces and Services

During the development of the OIH a basic reference implementation for an interface has been generated. This is a development site meant to test and exercise the above elements. It serves to demonstrate how others could also implement this approach and how future interfaces could be developed.

An example of the value of implementing a lightweight can be seen with the Government of Canada:

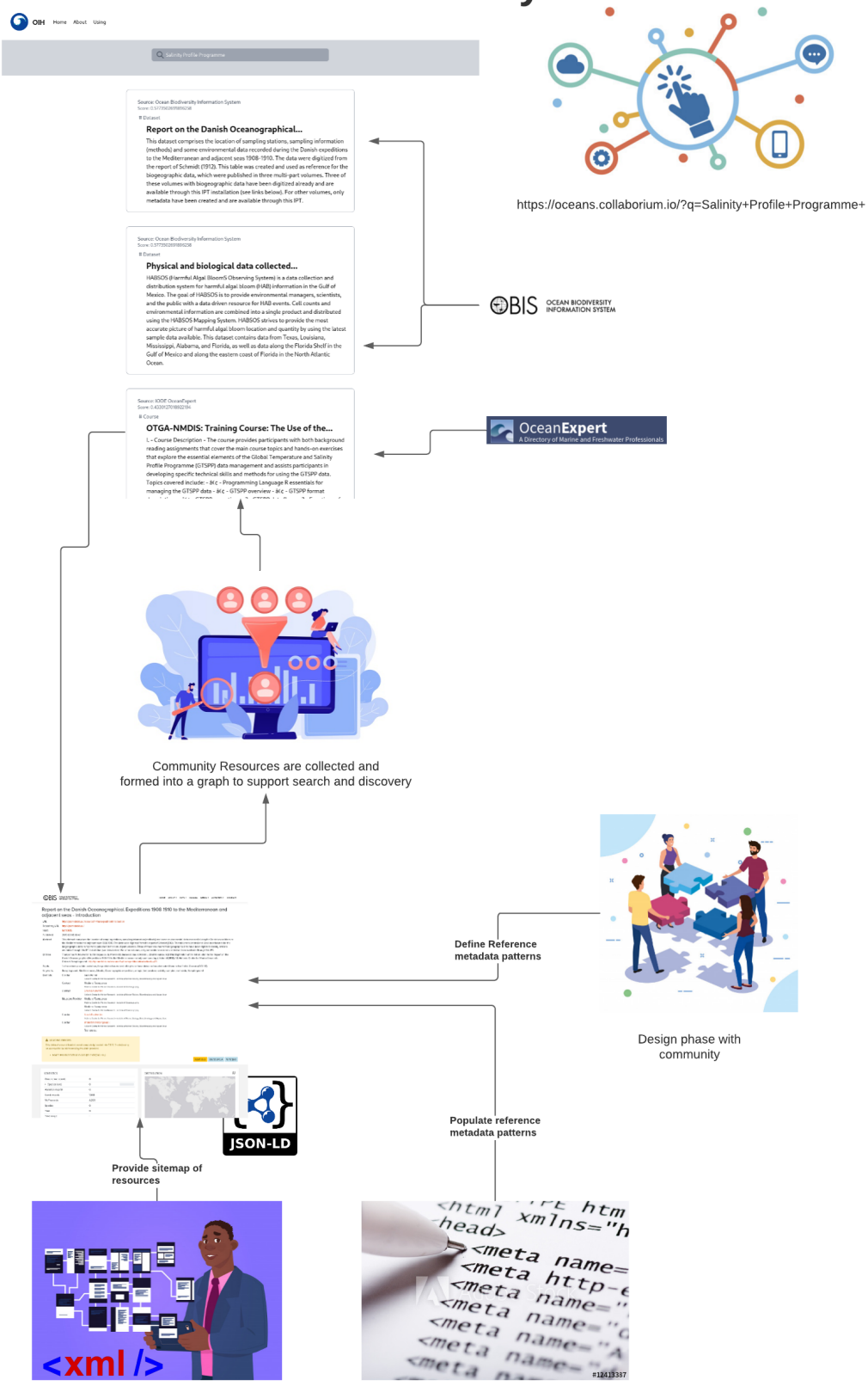
- The Federal Geospatial Platform is a intra-governmental data catalogue implementing the Harmonized North American Profile of ISO 19115 (HNAP), with content exposed externally via OGC CSW (Catalogue Services Web).
- This content is harvested by the public facing Open Maps platform, which includes a catalogue component that is fed in part by the Federal Geospatial Platform. DCAT-based metadata is derived from the original ISO 19115 based metadata. As this markup is recognized by web crawlers such as those hosted by Google, content is harvested and is subsequently visible through Google Dataset Search. Furthermore, the cited publication for the data is also link via a complementary link to Google Scholar.



Info-graphic

The following is a simple overview info-graphic of the Ocean Info Hub activity flow.

The Ocean Info Hub Activity Flow



Part I

Introduction

CHAPTER

ONE

STRUCTURED DATA ON THE WEB

1.1 About

Structured data on the web is a way to provide semantics and linked data in an approachable manner. This approach expresses concepts in JSON-LD which is a JavaScript notation popular among developers which easily expresses concepts (terms) and links to related resources (things). This structured data on the web approach has been popularized by the large commercial search providers like Google, Bing, Yandex and others via schema.org As described at schema.org: "Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond."

The popularity of leveraging the schema.org approach in the earth sciences can be attributed to both this ease of developer adoption and also to its foundational use of web architecture. A web architecture foundation aids adoption by the operations side as well as the developer side. It also takes advantage of the scale and resilience of the web.

The broad nature of schema.org even scopes to the concepts of Datasets. It is the existence of schema.org/Dataset that was a focus of several EarthCube projects (Project 418, Project 419 and the Resource Registry) from which spun up the ESIP Science on Schema work.

Additionally, Google leveraged schema.org/Dataset to develop and populate the Google Data Set Search and provides guidance to developers to facilitate this.

1.2 Web architecture approach

OIH is focused on leveraging the web architecture as the foundation for this approach. There are several key reasons for this vs approaches like OAI-PMH or others.

A key point is that in the processes of establishing a web presence, a standard step for groups, they have already begun to build the infrastructure needed for structured data on the web. Setting up special servers or establishing and maintaining special APIs to support harvesting is not required.

Also, a large collection of tooling already exists around JSON that is directly usable in JSON-LD. That scale extends to the use of schema.org patterns which have become common in the commercial web. Allowing us to bring those same patterns and the tooling to the science community.

Additionally, this approach keeps the metadata and its representation a product of the data providers. The actor in the life cycle most aware of needed edits, new records or other events. That same record then serves multiple consumers able to generate various value add products. This benefits the provider by facilitating multiple and varied discovery vectors for their holdings.

Another key factor is the web native and semantic nature of this representation of metadata. Traditional metadata, such as ISO, by itself does not express a web referenceable instance of concepts. In doing this, structured data on the web allow connections to be made and discovered by people and machines across many holdings. This aids in both serendipitous discovery and can also be leveraged to aid discovery via semantic relations.

1.3 Terminology

A CSV file is a text file containing spreadsheet information following a data model that is encoded using a convention of rows and commas defining columns.

A JSON-LD fle is a text file containing graph information following the RDF data model that is encoded using a convention based on JSON syntax.

JSON-LD is a way to serialize RDF that uses JSON notation. It is really no different then than RDF-XML, turtle, n-triples, etc. There are several ways to represent the RDF data model in text files (and some emerging binary ones like CBOR and parquet patterns).

Schema.org is a vocabulary for describing things similar to DCAT, FOAF, Dublin Core. It does this by using RDF as the underlying data model to represent this "ontology".

The confusion comes from the collision of outcomes. JSON-LD came about, partly, to allow the use of the RDF data model by a broader audience. This is done by leveraging a more popular notation for the data model, JSON, in the form of JSON-LD. Schema.org also wanted to advance the use of structured [meta]data by making it easier to use and connecting structured data to web pages. At the start, there were three approaches; RDFa, microformats and JSON-LD, to putting schema.org in web pages. However, the JSON-LD approach to incorporating this structured data has grown in popularity far beyond the others. As the popularity of both JSON-LD and schema.org grew, they often got conflated with each other.

The term "structured data on the web" is perhaps a more neutral way to discuss the use of vocabularies encoding in JSON-LD used in web pages. However, the phrase "schema.org" is starting to become the term for "structured data on the web using JSON-LD as a serialization". Even in cases where you combine other vocabularies such as DCAT with JSON-LD with no schema.org involved, it seems the way to convey this is to say: "We will use the schema.org 'pattern' with DCAT".

It is arguably not the best or most accurate communications strategy. It can conflate data models, serialization and vocabularies. However, it is concise and ubiquitous and not likely to change.

1.4 Intellectual Merit

OIH leverages structured data o the web patterns in the form of of Schema.org and JSON-LD encoding. This means that much of what is done to address OIH implementation by providers also is available both to existing commercial indexing approaches as well as emerging community practices

Additionally, both the publishing and indexing approaches are based on several web architecture patterns. Meaning that existing organization skills are leveraged and staff experience is enhanced. This helps to address both the sustainability of the OIH connection and the efficiency of organizational operation.

1.5 Broader Impacts

By leveraging existing technology and approaches a larger community is enabled to engage and make more samples discoverable and usable.

The nature of structured data on the web also provides the ability to apply semantic context to samples. This means richer discovery and information about samples, the past uses and potential future uses is more readily available.

Simplified architecture also means easier development of tools and interfaces to present the data. Allowing the presentation of samples and their information in a manner aligned with a given community's needs. A simplified architecture aids sustainability from both a technical and financial perspective.

CHAPTER

TWO

PERSONAS

2.1 About

The OIH system can be viewed as involving three personas. These are briefly presented here.


Publisher

A key persona whose activities are covered in detail in *Publishing patterns for OIH*


Aggregator

Leverages web architecture to retrieve structured data on the web and generate usable indexes.


User

The end user of the publishing and aggregation activities. May leverage the web for discovery or tools such as Jupyter for analytics and visualization.

2.2 Persona: Publisher

In OIH the Publisher is the one authoring the JSON-LD documents and publishing them to the web. Details on this persona can be found in the *Publisher* section. Additionally, this persona would be leveraging this encoding described in the *JSON-LD Foundation* section and the profiles described in the *Thematic Patterns*.

2.3 Persona: Aggregator

In OIH the Aggregator is a person or organization who is indexing resources on the web using the structured data on the web patterns described in this documentation.Details on the approach used by OIH and potential alternatives can be found in the *Aggregator* section.

2.4 Persona: User

The user is the individual or community who wished to leverage the indexes generated as a result of the publishing and aggregation activities. The user may be using the developed knowledge graph or some web interface built on top of the knowledge graph or other index.

User tools may be web sites or scientific notebooks. Some examples of these user experiences are described in the *User* section.

CHAPTER

THREE

PUBLISHER

3.1 About

This page describes the publishing process for structured data on the web approach OIH will use.

Note many software packages you are using might already implement this approach. See the section: *Existing support in software* at the bottom of this document.

See also:

We also recommend reviewing the document: Schema.org for Research Data Managers: A Primer

3.1.1 Architecture Implementation

The Ocean Info Hub (OIH) will leverage structured data on the web and web architecture patterns to expose metadata about resources of interest to the community. The primary tasks include:

- Authoring JSON-LD documents (https://json-ld.org/) aligned with ODIS OIH guidance to express the structured metadata for a resource. This step will require experience with using the existing metadata resources within an organization. So any necessary skills needed to access or query existing facility data systems will be needed to assemble the information to populate the JSON-LD data graph. The JSON-LD documents need to be generated using the tools/languages at the previous reference or through other means.
- Within the system architecture of the site, a JSON-LD document needs to be placed into the HTML DOM as a SCRIPT tag within the HEAD tag of each published resource. The SCRIPT tag pattern is:

```
<script type="application/ld+json">JSON_LD content</script>
```

- Additionally these resources that are marked up with these tags and JSON-LD documents should be expressed in an XML sitemap file. This should follow the guidance at https://www.sitemaps.org/. It should also include a lastmod node element as described at https://www.sitemaps.org/protocol.html which should indicate the date the resource metadata was last updated and published to the web.
- The process of aligning the JSON-LD is iterative at this stage as the OIH profile is evolved. To aid this we can leverage existing validation tools including JSONSchema, W3C SPARQL and more to communicate structure changes. These tools exist and need only be implemented using knowledge of command line environments. The results will then indicate revisions needed in the JSON-LD. OIH will provide the necessary templates for the tools to use against the authored JSON-LD documents.

Information on the sources, standards and vocabularies to be used can be found at: https://github.com/iodepo/odis-arch/tree/schema-dev/docs

3.1.2 Including JSON-LD in your resource page

To provide detailed and semantically described details on a resource, OIH uses a JSON-LD snippet or *data graph*. This small document provides details on the resource. It can also express any explicate connections to other resources an author may wish to express. The semantic nature of the document also means that connections may later be discovered through graph queries.

Pages will need a JSON-LD data graph placed in it via a typed script tag/element in the document head element like the following.

```
<script type="application/ld+json"></script>
```

An example data graph can be seen below. However, check the various thematic sections for more examples for a given thematic area.

```
{
    "@context": {
        "@vocab": "https://schema.org/",
        "endDate": {
            "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
        },
        "startDate": {
            "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
        }
    },
    "@id": "https://foo.org/url/to/metadata/representation",
    "@type": "Course",
    "description": "In this course you will get an introduction to the main tools and␣
↪ideas in the data scientist's toolbox...",
    "hasCourseInstance": {
        "@type": "CourseInstance",
        "courseMode": [
            "MOOC",
            "online"
        ],
        "endDate": "2019-03-21",
        "startDate": "2019-02-15"
    }
}
```

This example is from the training and courses thematic section. To view all the types being developed reference the Thematic section.

These JSON-LD documents leverage schema.org as the primary vocabulary. The examples in the thematic section provide examples for the various type.

JSON-LD Tools and References

A key resource for JSON-LD can be found at JSON-LD. There is also an interactive *playground* hosted there. The JSON-LD Playground is useful when testing or exploring approaches for JSON-LD data graphs. It will catch basic errors of syntax and use. Note, it will not catch semantic issues such as using properties on types that are out of range. Tools like the Structured Data Testing Tool are better at that. Also the documents and validation material created here OIH will also allow for that sort of testing and feedback.

Providers may also wish to provide content negotiation for type application/ld+json for these resources. Some indexers, like Gleaner, will attempt to negotiate for the specific serialization and this will likely lighten the load on the servers going forward.

Validation With SHACL or ShEx

To help facilitate the interconnection of resource, some application focused validation will be developed. Note, this validation does not limit what can be in the graphs.Rather, it simply provides insight on to how well a given graph can be leveraged for a specific application. For this project, the application will be the OIH search portal.

Some initial development work for this can be found in the *validation directory*

Validation Tools and References

- SHACL playground
- Schemarama
- Schimatos.org
  - demo
- Comparing ShEx and SHACL

Validation Leveraging JSON Schema

We have been exploring the potential to use JSON Schema combined with various on-line JSON editors (JSON Schema driven) to provide a potential approach to a more visual editing workflow. The workflow presented here is very ad hoc but exposes a potential route a group might take to develop a usable tool. Such a tool might, for example, leverage the Electron app dev environment to evolve this approach in a more dedicated tool/manner.

Use a JSON-LD document (Example) one could load this into something like the JSONschema.net tool.

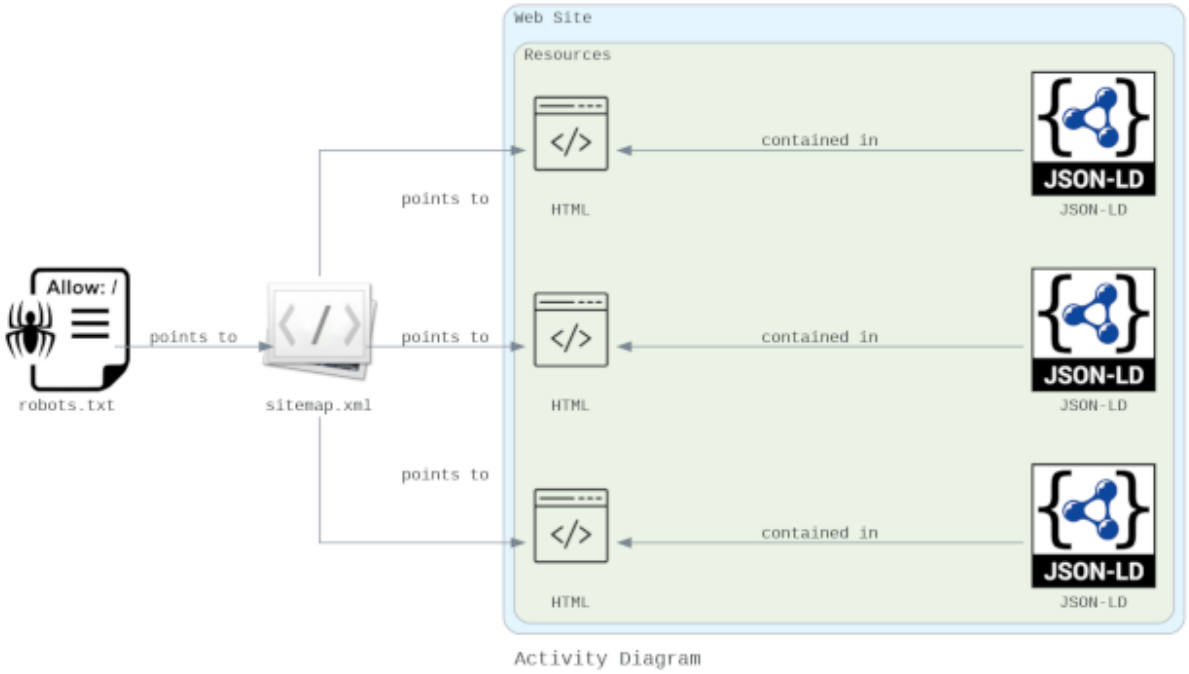
The results of the above can then been loaded into the online JSON-Editor at https://json-editor.github.io/json-editor/. (Ref: https://github.com/json-editor/json-editor)

The results of this then can be loaded into https://json-ld.org/playground/ to validate that we have well formed JSON-LD.

Though this workflow is rather crude and manual it exposes a route to a defined workflow based around established schema that leverages other tools and software libraries to generate a workable tool.

3.2 Basics

The basic activity can be seen in the following diagram:



3.2.1 Elements in detail

robots.txt

OPTIONAL: Providers may decide to generate or modify their robots.txt file to provide guidance to the aggregators. The plan is to use the Gleaner software (gleaner.io) as well as some Python based notebooks and a few other approaches in this test.

Gleaner uses an agent string of EarthCube_DataBot/1.0 and this can be used a robots.txt file to specify alternative sitemaps and guidance. This also allows a provider to provide guidance to Google and other potential indexers both for allow and disallow directives.

```
Sitemap: http://samples.earth/sitemap.xml

User-agent: *
Crawl-delay: 4
Allow: /

User-agent: Googlebot
Disallow: /id

User-agent: EarthCube_DataBot/1.0
Allow: /
Sitemap: https://example.org/sitemap.xml
```

sitemap.xml

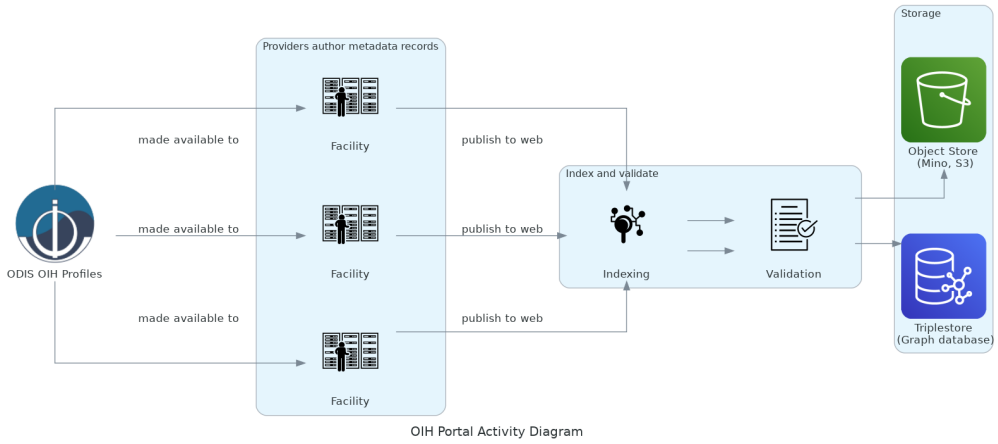
Providers will need to expose a set of resource landing pages using a sitemap.xml file. As noted above, providers can expose a sitemap file to just the target agent to avoid indexing test pages by commercial providers. You may wish to do this during testing or for other reasons. Otherwise, a sitemap.xml file exposed in general from somewhere in your site is perfectly fine.

Information on the sitemap structure can be found at sitemaps.org.

It is encouraged to use the sitemap lastmod node to provide guidance to indexers on page updates. Additionally indexers may test ways to evaluate additions and removals from the sitemap URL set to manage new or removed resources.

```
<?xml version="1.0" encoding="UTF-8"?>
<sitemapindex xmlns="http://www.sitemaps.org/schemas/site0.9">
   <sitemap>
      <loc>http://samples.earth/sitemap_websites_sampleseaxml</loc>
      <lastmod>2004-10-01T18:23:17+00:00</lastmod>
   </sitemap>
   <sitemap>
      <loc>http://samples.easitemap_doclouds_igsndatagraphs.xml</loc>
      <lastmod>2005-01-01</lastmod>
   </sitemap>
</sitemapindex>
```

3.3 Full Workflow



OIH Portal Activity Diagram

The architecture defines a workflow for objects seen in the above diagram.

The documents flow from; authoring, publishing and indexing to storage for the objects and the resulting graph. These resources are then ready for use in search and other functions.

Moving left to right we can review the image.

1. Providers are engaged in the process of developing the OIH example documents. These provide a *profile* to follow to represent the semantic metadata. Note, these are not limiters, simply guidance on minimum and recommend elements to address the functional goals of the OIH portal.
2. Providers use these documents to generate the JSON-LD data graphs.These can be either static documents or generated and placed in pages dynamically with Javascript or server side templates. These are the existing web pages for the resoruces, not enhanced with the semantic metadata snippets in the HTML source.
3. These are published to the web and referenced in the sitemap.xml document that is also made available. At this point this material is available to anyone who may wish to index it and provide discovery for these resources.
4. OIH Portal will then index and validate these resources on a recurring bases to maintain a current index. This index will include both the JSON-LD objects and the graph they form. This graph can be used for search, connections and other value add services for the community. The graph is also directly available to the community for them to use in support of services they may wish to provide.

3.4 Existing support in software

Many content management approaches and packages may already have support for the structured data on the web pattern.

A collection of starting points for their support follows.

- WordPress
- Drupal
- CKAN
- DSpace
- ERDDAP (native support)
- OPeNDAP (native support)
- GeoNode
  - schema.org issue ref

CHAPTER

FOUR

JSON-LD FOUNDATION

4.1 Introduction

This document provide a very brief introduction to the JSON-LD serialization format.The JSON-LD website has some detailed material and videos in their documentation section.

The material here is just a brief introduction. For this page we will be using a simplified version of a CreativeWork document. All the types used by OIH are defined by Schema.org types. In this case it is CreativeWork.

At the Schema.org site you will find extensive details on what the various types mean and the full range of their properties. For OIH we are defining only a few of these properties as of interest in the *Thematic section*. You are free to use additional properties to describe your resources. It will not cause any issues, however, the OIH interfaces may not leverage them. However, if you feel others would, or you use them yourself, it's encouraged to add them.

We will use the following simple JSON-LD document to show the various features of the format.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": "CreativeWork",
    "@id": "https://example.org/id/XYZ",
    "name": "Name or title of the document",
    "description": "Description of the creative work to aid in searching",
    "url":  "https://www.sample-data-repository.org/creativework/report.pdf"
}
```

```
<graphviz.dot.Digraph at 0x7effbc0ac220>
```

Note: A small note on nomenclature. In Schema.org, as in ontologies, the class or type names of Things will be uppercase. So, for example, in the above JSON-LD data graph, this is describing a resource of type `CreativeWork`. So the type CreativeWork will start with an uppercase letter. The property `name` is a property of the type and properties like this will be lowercase.

4.2 The Context

The context is where the terms used in a document are connected to definitions and identifiers for them. If you wish to dive into the details of the context check out the W3 JSON-LD 1.1 Recommendations Context section.

The context part of this document is highlighted below.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": "CreativeWork",
    "@id": "https://example.org/id/XYZ",
    "name": "Name or title of the document",
    "description": "Description of the creative work to aid in searching",
    "url":  "https://www.sample-data-repository.org/creativework/report.pdf"
}
```

Note: This @context section will be the same for all the documents described in OIH documentation with the exception of the spatial patterns.

As justed noted, for the spatial patterns we add in the OGC context to all us to use terms from that vocabulary. Below we can see the addition of the geosparql context in line 4 and the use of the vocabulary, using the defined geosparql: prefix in lines 9, 11 and 15.

If we wanted to use other vocabularies like DCAT or FOAF, we would add them to the context with a prefix and then the vocabulary namespace. We could then use terms from that vocabulary in our document following the same prefix:term pattern.

```
{
    "@context": {
      "@vocab": "https://schema.org/",
      "geosparql": "http://www.opengis.net/ont/geosparql#"
    },
    "@id": "https://example.org/id/XYZ",
    "@type": "Dataset",
    "name": "Data set name",
    "geosparql:hasGeometry": {
        "@type": "http://www.opengis.net/ont/sf#Point",
        "geosparql:asWKT": {
            "@type": "http://www.opengis.net/ont/geosparql#wktLiteral",
            "@value": "POINT(-76 -18)"
        },
        "geosparql:crs": {
            "@id": "http://www.opengis.net/def/crs/OGC/1.3/CRS84"
        }
    }
}

```

4.3 Graph

The next section we will discuss is the graph part of the document. This is where the properties and values of our resource are described. First though, let's visit a couple special properties in our document.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": "CreativeWork",
    "@id": "https://example.org/id/XYZ",
    "name": "Name or title of the document",
    "description": "Description of the creative work to aid in searching",
    "url":  "https://www.sample-data-repository.org/creativework/report.pdf"
}
```

4.3.1 Node identifiers (@id)

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": "CreativeWork",
    "@id": "https://example.org/id/XYZ",
    "name": "Name or title of the document",
    "description": "Description of the creative work to aid in searching",
    "url":  "https://www.sample-data-repository.org/creativework/report.pdf"
}
```

The first special property is the @id property. This is the identifier for the top level node in the graph and is typically the identifier for the record.

Note: It should be noted this is the not the ID for the object being described but rather the record itself. If you are describing a dataset with a DOI, for example, the @id is not that DOI. Rather it is the ID, potentially the URL, for the metadata record about that dataset. Your dataset ID would be included in the metadata record using the the identifier property.

It's good practice to ensure all your records have an @id property. If there is no value then the resource is identified by what is known as a blank node. Such identifiers do not allowing use in a Linked Open Data approach and are generally not recommended.

The @id should be the URL for the metadata record itself. Not the HTML page the record is in. However, these might be the same if use use content negotiation to select between HTML and JSON-LD representations of the record.

4.3.2 Type identifiers (@type)

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": "CreativeWork",
    "@id": "https://example.org/id/XYZ",
    "name": "Name or title of the document",
    "description": "Description of the creative work to aid in searching",
    "url":  "https://www.sample-data-repository.org/creativework/report.pdf"
}
```

The next property to focus on is the @type property. This describes the type of record we are describing.

Note: In Schema.org and in most vocabularies, types will be named with a capitol letter. Properties on these types will be all lower case. So, CreateWork, as a type, starts with a upper case C. Then, name, as a property on the CreateWork type, starts with a lower case n.

For OIH these type for the various thematic profiles are defined in the documentation for the types.

4.3.3 Other properties

At this point we can look at the other properties for our type.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": "CreativeWork",
    "@id": "https://example.org/id/XYZ",
    "name": "Name or title of the document",
    "description": "Description of the creative work to aid in searching",
    "url":  "https://www.sample-data-repository.org/creativework/report.pdf"
}
```

As noted, we are using Schema.org type for OIH. In this case, as mentioned, this is type CreativeWork. So any of the properties seen at the Schema.org site can be used. The key properties of value to the OIH implementation can then be found, for this type, in the *Documents thematic type*.

4.3.4 Domain and range

The domain of a property identifies the type of object it can be applied to.So if the domain of a property like schema.org/knowsAbout is Person and Organization. Then that property can only be used on those types. For example, it would not be correct to use knowsAbout on a resource of type Dataset.

The range of a property identifies the type of object the property can point to. In the case of knowAbout, we see its range as Text, Thing or URL. This means the property can point to a Text, Thing or URL object.

In schema.org, the domain will be identified as "Used on these types", and the range will be identified as "Values expected to be one of these types". You can see this at the schema.org/knowsAbout page.

4.3.5 Thing and DataType

The Thing and Datatype types are two special types we should mention. Thing is the upper level and most generic type in schema.org. Everything in schema.org is descended from Thing. So when knowsAbout says its range includes Thing, it means you can use any type in schema.org as the value of that property.

DataType is the basic data type Thing in schema.org and is a subclass of rdfs:Class. A DataType includes things like Integers, Strings, DateTime, etc. So, using again knowsAbout, we see the range includes not only Thing by also the DataTypes Text and URL, where URL is actually a sub-type of Text.

Part II

Profiles

CHAPTER

FIVE

THEMATIC PATTERNS

5.1 Introduction

These thematic patterns are managed by OIH and the community to add in the discovery and use of ocean related resources. The patterns are simple examples of Schema.org types, with a focus on the properties and type relations of value to the Ocean InfoHub and the community it engages.

These "profiles" provide both a starting point for new users and a catalysis for discussion and extension with the community.

5.2 Thematic Profiles

These profiles represent reference implementation of schema.org Types related to the identified ODIS thematic areas. They provide a set of minimal elements and notes on more detailed elements.

These are not final and will evolve with community input. As this process moves forward we will implement versioning the profiles to provide stable implementations providers can reliably leverage in their workflows.

5.2.1 Core Profiles

Six key categories of interest:

1. *Experts and Institutions*
2. *Documents*
3. *Spatial Maps*
4. *Projects*
5. *Training*
6. *Vessels*

5.2.2 Supporting Profiles

In support of these five thematic types above, these cross cutting types and properties were selected for attention. They represent some key patterns people may wish to leverage when describing their resources.

1. *Spatial Geometry*
2. *Services*
3. *Term Lists*
4. *Languages*
5. *Linking to Principles*
6. *Identifier Patterns*

See also:

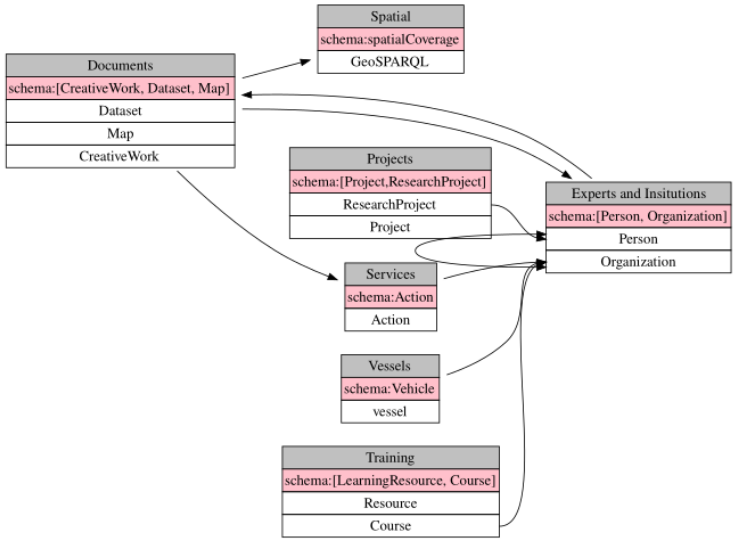
For OIH the focus is on generic documents which can scope reports, data and other resources. In those cases where the resources being described are of type Dataset you may wish to review patterns developed for GeoScience Datasets by the ESIP Science on Schema community.

See also:

For OIH the focus is on generic documents which can scope reports, data and other resources. In those cases where the resources being described are life sciences resources such as datasets, software, and training materials. we recommend following patterns developed by Bioschemas.

5.3 Example Connections

The following figure provides an overview of potential connections between the various thematic types in OIH. These are not *all* the relations but simply some to demonstrate the concept as well as give some guidance on the connection of value in query results.


Spatial
schema:spatialCoverage
GeoSPARQL
Documents
schema:[CreativeWork, Dataset, Map]
Dataset
Map
CreativeWork
Projects
schema:[Project,ResearchProject]
ResearchProject
Project
Experts and Insitutions
schema:[Person, Organization]
Person
Organization
Services
schema:Action
Action
Vessels
schema:Vehicle
vessel
Training
schema:[LearningResource, Course]
Resource
Course

CHAPTER

SIX

EXPERTS AND INSTITUTIONS

6.1 About

Expert: A person who has a deep understanding of a particular subject area.

Institution: A group of people working together to provide a particular service.

6.2 Example: Person Graph

The following graph present a basic record we might present about a person.We can break down some of the key attributes of this record.

OIH is using schema.org/Person for this type. Any of the properties of Person seen there are valid to use in such a record.

At the core though there are a few key items OIH is looking for in a Person record.

```
{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@id": "https://example.org/id/x",
  "@type": "Person",
  "name": "Jane Doe",
  "jobTitle": "Professor",
  "telephone": "(425) 123-4567",
  "url": "http://www.janedoe.com",
  "knowsAbout": [
    {
      "@type": "Text",
      "description": "Invasive species in brackish water"
    },
    {
      "@type": "URL",
      "url": "https://www.wikidata.org/wiki/Q183368"
    },
    {
      "@id": "https://example.org/id/course/x",
      "@type": "Course",
      "description": "In this course ...",
      "url": "URL to the course"
    }
  ],
```

(continues on next page)

(continued from previous page)

```
  "identifier": {
    "@id": "https://orcid.org/0000-0002-2257-9127",
    "@type": "PropertyValue",
    "propertyID": "https://registry.identifiers.org/registry/orcid",
    "url": "https://orcid.org/0000-0002-2257-9127",
    "description": "Optional description of this record..."
  },
  "nationality": [
    {
      "@type": "Country",
      "name": "Fiji"
    },
    {
      "@type": "DefinedTerm",
      "url": "https://unece.org/trade/cefact/unlocode-code-list-country-and-territory
↪",
      "inDefinedTermSet": "UN/LOCODE Code List by Country and Territory",
      "name": "Fiji",
      "termCode": "FJ"
    }
  ],
  "knowsLanguage" :{
    "@type": "Language",
    "name": "Spanish",
    "alternateName": "es"
  }
}
```

```
<graphviz.dot.Digraph at 0x7f52306adb20>
```

6.2.1 Details: Authoritative Reference

For each profile there are a few key elements we need to know about. One key element is what the authoritative reference or canonical identifier is for a resource.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/x",
    "@type": "Person",
    "identifier": {
        "@id": "https://orcid.org/0000-0002-2257-9127",
        "@type": "PropertyValue",
        "description": "Optional description of this record...",
        "propertyID": "https://registry.identifiers.org/registry/orcid",
        "url": "https://orcid.org/0000-0002-2257-9127"
    }
}
```

```
<graphviz.dot.Digraph at 0x7f5229229ca0>
```

6.2.2 Details: nationality

Nationality provide connections to languages a person is connected with. The property, schema.org/nationality, is used to present that. In the OIH we need to state what the semantics of nationality are for our use case.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/x",
    "@type": "Person",
    "nationality": [
        {
            "@type": "Country",
            "name": "Fiji"
        },
        {
            "@type": "DefinedTerm",
            "inDefinedTermSet": "UN/LOCODE Code List by Country and Territory",
            "name": "Fiji",
            "termCode": "FJ",
            "url": "https://unece.org/trade/cefact/unlocode-code-list-country-and-
 ↪territory"
        }
    ]
}
```

```
<graphviz.dot.Digraph at 0x7f52292298e0>
```

Note: The visual above demonstrates an issue that can be seen in several of the graph. Where we don't use an @id the graph will be represented as a "blank node".These will be uniquely identified in the graph, however, in the construction of the visual this is a common blank node and results in the double arrows pointing to an underscore. This is a visualization issue and not a proper representation of the graph structure.

6.2.3 Details: knowsLanguage

Knows about provide connections to languages a person is connected with. The property, schema.org/knowsLanguage, is used to present that. Multiple languages can be expressed using the JSON array [] syntax.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/x",
    "@type": "Person",
    "knowsLanguage": {
        "@type": "Language",
        "alternateName": "es",
        "name": "Spanish"
    }
}
```

```
<graphviz.dot.Digraph at 0x7f52292b05e0>
```

6.2.4 Details: Knows About

Knows about provide connections to resources a person is connected with. The property, schema.org/knowsAbout, can connect a Person or Organization to Text, URL or any Thing type.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/x",
    "@type": "Person",
    "knowsAbout": [
        {
            "@type": "Text",
            "description": "Invasive species in brackish water"
        },
        {
            "@type": "URL",
            "url": "https://www.wikidata.org/wiki/Q183368"
        },
        {
            "@id": "https://example.org/id/course/x",
            "@type": "Course",
            "description": "In this course ...",
            "url": "URL to the course"
        }
    ]
}
```

```
<graphviz.dot.Digraph at 0x7f5229229790>
```

6.3 Example: Institution Graph

Here we have an example of an data graph for type schema.org/Organization.For the identifier we are using the a GRID, but this could also be something like a ROR.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://index.example.org/id/org/x",
    "@type": "Organization",
    "address": {
        "@type": "PostalAddress",
        "addressLocality": "Paris, France",
        "postalCode": "F-75002",
        "streetAddress": "38 avenue de l'Opera"
    },
    "email": "secretariat(at)example.org",
    "name": "Organization X",
```

(continues on next page)

(continued from previous page)

```
15      "description": "Description of org ...",
16      "telephone": "( 33 1) 42 68 53 00",
17      "url": "https://example.org/",
18      "member": [
19          {
20              "@id": "https://example.org/id/org/1",
21              "@type": "Organization",
22              "name": "Organization A",
23              "description": "Org A is a potential parent organization of Org X"
24          },
25          {
26              "@id": "https://orcid.org/0000-0002-2257-9127",
27              "@type": "Person"
28          }
29      ],
30      "identifier": {
31          "@id": "https://grid.ac/institutes/grid.475727.4",
32          "@type": "PropertyValue",
33          "description": "UN Department of Economic and Social Affairs Sustainable␣
 ↪Development",
34          "propertyID": "https://registry.identifiers.org/registry/grid",
35          "url": "https://grid.ac/institutes/grid.475727.4"
36      }
37  }
```

6.3.1 One the property membership

Line 18-29 show the inclusion of a schema.org/membership property. There are issues to note here both for consumers (aggregators) and providers (publishers). The Person type is show connected simply on a type and id. This provides the cleanest connection. If a member is added by type and id, as in the case of the "Organization A" link, there is the problem of additional triples being added. Here, the name and description properties are going to add triples to the OIH KG. In so doing, we run the risk or adding potentially un-authoritative information. The aggregator doesn't know if triples here are or are not provided by an actor authoritative for those properties. This could be addresses with framing or validation workflows, or ignored. The prov elements stored could be leveraged to later track down sources, but don't provide further information on the issue of authority.

It is recommended that best practice is to attempt to link only on ids (with a type in all cases) where possible. If you are connecting with a type, do not provide additional properties. In cases where such an id can not be provided, you may wish to fill out basic properties you can provide with confidence.

```
<graphviz.dot.Digraph at 0x7f52296175e0>
```

6.3.2 Details: Authoritative Reference

For each profile there are a few key elements we need to know about. One key element is what the authoritative reference or canonical identifier is for a resource.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://index.example.org/id/org/x",
```

(continues on next page)

(continued from previous page)

```
    "@type": "Organization",
    "identifier": {
        "@id": "https://grid.ac/institutes/grid.475727.4",
        "@type": "PropertyValue",
        "description": "UN Department of Economic and Social Affairs Sustainable␣
↪Development",
        "propertyID": "https://registry.identifiers.org/registry/grid",
        "url": "https://grid.ac/institutes/grid.475727.4"
    }
}
```

```
<graphviz.dot.Digraph at 0x7f52292b08e0>
```

6.4 References

- schema:Person
- scheme:Organization
- Science on Schema Repository
- https://oceanexpert.org/
  - Example page expert
  - Example page institution
  - Ocean Expert: refernce: Adam Leadbetter

CHAPTER

SEVEN

DOCUMENTS

7.1 About

Documents: These include datasets, reports or other documents

See also:

For OIH the focus is on generic documents which can scope reports, data and other resources. In those cases where the resources being described are of type Dataset you may wish to review patterns developed for GeoScience Datasets by the ESIP Science on Schema community.

7.2 Creative works (documents)

Documents will include maps, reports, guidance and other creative works. Due to this OIH will focus on a generic example of schema.org/CreativeWork and then provide examples for more focused creative work examples.

Load in JSON-LD Playground

Load in Structured Data Testing Tool

```
{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@type": "CreativeWork",
  "@id": "https://example.org/id/XYZ",
  "name": "Name or title of the document",
  "description": "Description of the creative work to aid in searching",
  "url": "https://www.sample-data-repository.org/creativework/report.pdf",
  "contributor": {
    "@type": "Organization",
    "@id": "http://www.foo.org/orgID",
    "legalName": "Some Institute"
  },
  "author": {
    "@id": "https://www.sample-data-repository.org/person/51317",
    "@type": "Person",
    "name": "Dr Uta Passow",
    "givenName": "Uta",
    "familyName": "Passow",
    "url": "https://www.sample-data-repository.org/person/51317"
  },
  "identifier": {
```

(continues on next page)

(continued from previous page)

```
    "@id": "https://doi.org/10.5066/F7VX0DMQ",
    "@type": "PropertyValue",
    "propertyID": "https://registry.identifiers.org/registry/doi",
    "value": "doi:10.5066/F7VX0DMQ",
    "url": "https://doi.org/10.5066/F7VX0DMQ"
  },
  "keywords": {
    "@type": "DefinedTerm",
    "inDefinedTermSet": {
      "@type": "DefinedTermSet",
      "name": "Name of the set",
      "description": "Description of the set",
      "url": "url for the set"
    },
    "termCode": "A code that identifies this DefinedTerm within a DefinedTermSet"
  },
  "provider": {
    "@id": "https://www.repositoryB.org",
    "@type": "Organization",
    "legalName": "Sample Data Repository Office",
    "name": "SDRO",
    "sameAs": "http://www.re3data.org/repository/r3dxxxxxxxx",
    "url": "https://www.sample-data-repository.org"
  },
  "license": "http://spdx.org/licenses/CC0-1.0",
  "publisher": {
    "@id": "https://www.publishingrus.org",
    "@type": "Organization",
    "legalName": "Some Institute"
  }
}
```

```
<graphviz.dot.Digraph at 0x7f1e1c10bb20>
```

7.2.1 Details: Authoritative Reference

For each profile there are a few key elements we need to know about. One key element is what the authoritative reference or canonical identifier is for a resource.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/XYZ",
    "@type": "CreativeWork",
    "identifier": {
        "@id": "https://doi.org/10.5066/F7VX0DMQ",
        "@type": "PropertyValue",
        "propertyID": "https://registry.identifiers.org/registry/doi",
        "url": "https://doi.org/10.5066/F7VX0DMQ",
        "value": "doi:10.5066/F7VX0DMQ"
    }
}
```

```
<graphviz.dot.Digraph at 0x7f1e14c86dc0>
```

7.2.2 Frame on publisher and provider

Our JSON-LD documents are graphs that can use framing to subset. In this case we can look closer at the author property which points to a type Person.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/XYZ",
    "@type": "CreativeWork",
    "provider": {
        "@id": "https://www.repositoryB.org",
        "@type": "Organization",
        "legalName": "Sample Data Repository Office",
        "name": "SDRO",
        "sameAs": "http://www.re3data.org/repository/r3dxxxxxxxx",
        "url": "https://www.sample-data-repository.org"
    },
    "publisher": {
        "@id": "https://www.publishingrus.org",
        "@type": "Organization",
        "legalName": "Some Institute"
    }
}
```

```
<graphviz.dot.Digraph at 0x7f1e14c6e8e0>
```

7.2.3 Frame on author type Person

Our JSON-LD documents are graphs that can use framing to subset. In this case we can look closer at the author property which points to a type Person.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/XYZ",
    "@type": "CreativeWork",
    "author": {
        "@id": "https://www.sample-data-repository.org/person/51317",
        "@type": "Person",
        "familyName": "Passow",
        "givenName": "Uta",
        "name": "Dr Uta Passow",
        "url": "https://www.sample-data-repository.org/person/51317"
    }
}
```

```
<graphviz.dot.Digraph at 0x7f1e14c86f10>
```

7.2.4 References

- For dataset we can use SOS Dataset
- OBPS group is using JericoS3 API (ref: https://www.jerico-ri.eu/)
  - Traditional knowledge points here
  - sounds like they use dspace
- For other document these are likely going to be some schema:CretiveWork with there being many subtypes we can explore. See also here Adam Leadbetter's work at Ocean best practices
  - This is a great start and perhaps helps to highlight why SHACL shapes are useful
  - https://irishmarineinstitute.github.io/erddap-lint/
  - https://github.com/earthcubearchitecture-project418/p419dcatservices/blob/master/CHORDS/DataFeed.jsonld *EMODnet (Coner Delaney)
  - ERDAP also
  - Are we talking links from schema.org that link to OGC and ERDAP services
  - Are these methods?
  - Sounds like may link to external metadata for interop they have developed in the community
- NOAA connected as well
  - Interested in OGC assets
  - ERDAP data platform

CHAPTER

EIGHT

SPATIAL MAPS

8.1 About

Maps: A map represented by a static file or document

A map in this context would be a static file or document of some sort. Map services like those described by an OGC Catalogue Service or other GIS service would be described as a service.

Note: In the current context, schema.org Map typically references maps a document. Here we are likely to reference a KML, Shapefile or GeoPackage. We may wish to then indicate the type of document it is through a mimetype via encoding.

The schema.org type Map only offers one special property beyond the parent CreativeWork. That is a mapType which is an enumeration of types that do not apply to OIH use cases. However, the use of the Map typing itself may aid in narrowing search requests later to a specific creative work.

Load in JSON-LD Playground

Load in Structured Data Testing Tool

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": "Map",
    "@id": "https://example.org/id/XYZ",
    "name": "Name or title of the document",
    "description": "Description of the map to aid in searching",
    "url": "https://www.sample-data-repository.org/creativework/map.pdf",
    "identifier": {
        "@id": "https://doi.org/10.5066/F7VX0DMQ",
        "@type": "PropertyValue",
        "propertyID": "https://registry.identifiers.org/registry/doi",
        "value": "doi:10.5066/F7VX0DMQ",
        "url": "https://doi.org/10.5066/F7VX0DMQ"
    },
    "keywords": [
        {
            "@id": "http://purl.org/dc/dcmitype/Image",
            "@type": "DefinedTerm",
            "inDefinedTermSet": "http://purl.org/dc/terms/DCMIType",
            "termCode": "Image",
            "name": "Image"
```

(continues on next page)

(continued from previous page)

```
        },
        "Region X",
        {
            "@id": "https://www.wikidata.org/wiki/Q350134",
            "@type": "URL",
            "url": "https://www.wikidata.org/wiki/Q350134",
            "name": "North Atlantic Ocean"
        }
    ]
}
```

```
<graphviz.dot.Digraph at 0x7fa130251b20>
```

8.1.1 Details: Authoritative Reference

For each profile there are a few key elements we need to know about. One key element is what the authoritative reference or canonical identifier is for a resource.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/XYZ",
    "@type": "Map",
    "identifier": {
        "@id": "https://doi.org/10.5066/F7VX0DMQ",
        "@type": "PropertyValue",
        "propertyID": "https://registry.identifiers.org/registry/doi",
        "url": "https://doi.org/10.5066/F7VX0DMQ",
        "value": "doi:10.5066/F7VX0DMQ"
    }
}
```

```
<graphviz.dot.Digraph at 0x7fa128dcbdc0>
```

8.1.2 Keywords

We can see three different approaches here to defining keywords.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/XYZ",
    "@type": "Map",
    "keywords": [
        {
            "@id": "http://purl.org/dc/dcmitype/Image",
            "@type": "DefinedTerm",
            "inDefinedTermSet": "http://purl.org/dc/terms/DCMIType",
            "name": "Image",
            "termCode": "Image"
```

(continues on next page)

(continued from previous page)

```
        },
        "Region X",
        {
            "@id": "https://www.wikidata.org/wiki/Q350134",
            "@type": "URL",
            "name": "North Atlantic Ocean",
            "url": "https://www.wikidata.org/wiki/Q350134"
        }
    ]
}
```

```
<graphviz.dot.Digraph at 0x7fa1291c0340>
```

8.1.3 References

- For dataset we can use SOS Dataset
- OBPS group is using JericoS3 API (ref: https://www.jerico-ri.eu/)
  - Traditional knowledge points here
  - sounds like they use dspace
- For other document these are likely going to be some schema:CretiveWork with there being many subtypes we can explore. See also here Adam Leadbetter's work at Ocean best practices
  - This is a great start and perhaps helps to highlight why SHACL shapes are useful
  - https://irishmarineinstitute.github.io/erddap-lint/
  - https://github.com/earthcubearchitecture-project418/p419dcatservices/blob/master/CHORDS/DataFeed.jsonld *EMODnet (Coner Delaney)
  - ERDAP also
  - Are we talking links from schema.org that link to OGC and ERDAP services
  - Are these methods?
  - Sounds like may link to external metadata for interop they have developed in the community
- NOAA connected as well
  - Interested in OGC assets
  - ERDAP data platform

CHAPTER

NINE

PROJECTS

9.1 About

Project: An enterprise (potentially individual but typically collaborative), planned to achieve a particular aim. Use properties from Organization, subOrganization/parentOrganization to indicate project sub-structures.

9.2 Research Project

This is what a basic research project data graph might look like. We have the full record below, but this shows some of the basics we would be looking for.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/resproj/X",
    "@type": "ResearchProject",
    "description": "Repo description ... ",
    "identifier": {
        "@id": "https://grid.ac/institutes/grid.475727.4",
        "@type": "PropertyValue",
        "description": "UN Department of Economic and Social Affairs Sustainable␣
↪Development",
        "propertyID": "https://registry.identifiers.org/registry/grid",
        "url": "https://grid.ac/institutes/grid.475727.4"
    },
    "legalName": "Example Data Repository",
    "name": "ExDaRepo",
    "url": "https://www.example-data-repository.org"
}
```

```
<graphviz.dot.Digraph at 0x7f7815ed8250>
```

9.2.1 Details: Authoritative Reference

For each profile there are a few key elements we need to know about. One key element is what the authoritative reference or canonical identifier is for a resource.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/resproj/X",
    "@type": "ResearchProject",
    "identifier": {
        "@id": "https://grid.ac/institutes/grid.475727.4",
        "@type": "PropertyValue",
        "description": "UN Department of Economic and Social Affairs Sustainable␣
↪Development",
        "propertyID": "https://registry.identifiers.org/registry/grid",
        "url": "https://grid.ac/institutes/grid.475727.4"
    }
}
```

```
<graphviz.dot.Digraph at 0x7f7815e86250>
```

9.3 Full Research Project

Here is what our full record looks like. We have added in several more nodes to cover things like funding source, policy connections, spatial area served and parent organization.

```
{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@type":     "ResearchProject",
  "@id": "https://example.org/id/resproj/X",
  "legalName": "Example Data Repository",
  "name": "ExDaRepo",
  "url": "https://www.example-data-repository.org",
  "description": "Repo description ... ",
  "logo": {
    "@type": "ImageObject",
    "url": "https://www.example-data-repository.org/logo.jpg"
  },
  "identifier": {
    "@id": "https://grid.ac/institutes/grid.475727.4",
    "@type": "PropertyValue",
    "description": "UN Department of Economic and Social Affairs Sustainable␣
↪Development",
    "propertyID": "https://registry.identifiers.org/registry/grid",
    "url": "https://grid.ac/institutes/grid.475727.4"
  },
  "contactPoint": {
    "@id": "https://www.example-data-repository.org/about-us",
    "@type": "ContactPoint",
    "name": "Support",
```

(continues on next page)

(continued from previous page)

```
    "email": "info@example-data-repository.org",
    "url": "https://www.example-data-repository.org/about-us",
    "contactType": "customer support"
  },
  "funder": {
    "@type": "FundingAgency",
    "@id": "https://dx.doi.org/10.13039/10000001",
    "legalName": "National Science Foundation",
    "alternateName": "NSF",
    "url": "https://www.nsf.gov/"
  },
  "ethicsPolicy": {
    "@type": "CreativeWork",
    "@id": "https://example.org/id/XYZ",
    "name": "Name or title of the document",
    "description": "Description of the creative work ",
    "url": "https://www.foo.org/creativework/ethicsPolicy.pdf"
  },
  "diversityPolicy": {
    "@type": "CreativeWork",
    "@id": "https://example.org/id/ABC",
    "name": "Name or title of the document",
    "description": "Description of the creative work",
    "url": "https://www.foo.org/creativework/diversityPolicy.pdf"
  },
  "areaServed": [
    {
      "@type": "Place",
      "geo": {
        "@type": "GeoCoordinates",
        "latitude": 39.3280,
        "longitude": 120.1633
      },
      "description": "Description of the area served"
    },
    {
      "@type": "Text",
      "description": "Textual description of area served"
    },
    {
      "@type": "AdministrativeArea",
      "geo": {
        "@type": "GeoCoordinates",
        "latitude": 39.3280,
        "longitude": 120.1633
      },
      "description": "Needs to be subset of Place, Review Place"
    }
  ],
  "parentOrganization": {
    "@type": "Organization",
    "@id": "http://www.someinstitute.edu",
    "legalName": "Some Institute",
    "name": "SI",
    "url": "http://www.someinstitute.edu",
    "address": {
      "@type": "PostalAddress",
```

(continues on next page)

(continued from previous page)

```
      "streetAddress": "234 Main St.",
      "addressLocality": "Anytown",
      "addressRegion": "ST",
      "postalCode": "12345",
      "addressCountry": "USA"
    }
  }
}
```

```
<graphviz.dot.Digraph at 0x7f7815ecabe0>
```

9.3.1 References

- https://schema.org/Project

CHAPTER

TEN

TRAINING

10.1 About

From https://schema.org/Course

Course: A description of an educational course which may be offered as distinct instances at which take place at different times or take place at different locations, or be offered through different media or modes of study. An educational course is a sequence of one or more educational events and/or creative works which aims to build knowledge, competence or ability of learners.

10.2 Basic Course style

```
{
    "@context": {
        "@vocab": "https://schema.org/",
        "endDate": {
            "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
        },
        "startDate": {
            "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
        }
    },
    "@id": "https://example.org/id/course/1",
    "@type": "Course",
    "description": "In this course you will get an introduction to the main tools and␣
↪ideas in the data scientist's toolbox...",
    "hasCourseInstance": [
        {
            "@type": "CourseInstance",
            "courseMode": [
                "MOOC1",
                "online"
            ],
            "endDate": "2019-03-21",
            "startDate": "2019-02-15",
            "attendee": {
                "@type": "Person",
                "name": "Jane Doe",
                "jobTitle": "Professor",
                "telephone": "(425) 123-4567",
                "url": "http://www.janedoe.com",
```

(continues on next page)

(continued from previous page)

```
                "identifier": {
                    "@id": "ID_value_string",
                    "@type": "PropertyValue",
                    "propertyID": "This can be text or URL for an ID like ORCID",
                    "url": "https://foo.org/linkToPropertyIDPage",
                    "description": "Optional description of the ID"
                }
            }
        },
        {
            "@type": "CourseInstance",
            "courseMode": [
                "MOOC2",
                "online"
            ],
            "endDate": "2019-05-21",
            "startDate": "2019-04-15"
        }
    ]
}
```

```
<graphviz.dot.Digraph at 0x7f282428aa60>
```

10.3 Simple Course

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/course/2",
    "@type": "Course",
    "courseCode": "F300",
    "name": "Physics",
    "provider": {
        "@type": "CollegeOrUniversity",
        "name": "University of Bristol",
        "url": {
            "@id": "/provider/324/university-of-bristol"
        }
    }
}
```

```
<graphviz.dot.Digraph at 0x7f28169e6670>
```

10.4 References

- RDA Education and Training on handling of research data IG
- DC Tabular Application Profiles (DC TAP) - Primer
- https://www.w3.org/TR/xmlschema11-2/
  - Use YYYY-MM-DDThh:mm:ss or YYYY-MM-DD
- http://www.marinetraining.eu/
  - Example page
- https://oceanexpert.org/
- Example page
- OCTO
- https://oceansummerschools.iode.org/
- https://www.openchannels.org/upcoming-events-list
- https://catalogue.odis.org/search/type=16
- https://clmeplus.org/

CHAPTER

ELEVEN

VESSELS

11.1 About

OIH is exploring how we might leverage schema.org to describe research vessels.Note that schema.org is a very broad vocabulary and as such specific concepts like research vessel is not well aligned.

In Schema.org the type Vehicle is described as a device that is designed or used to transport people or cargo over land, water, air, or through space. We have used this broad scoping to cover research vessels. We could go on to connect this type then to a descriptive property in a concept such at the WikiData entry for Research Vessel, Q391022. We may wish to leverage some of the approaches in *Keywords and Defined Terms*.

Our goal is to use schema.org as a simple upper level vocabulary that allows us to describe research vessels in a simple and then connect off to more detailed information on them.

So the goal here is to show how we can use schema.org as a discovery layer and link more directly to detailed institutional metadata records.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/X",
    "@type": "Vehicle",
    "name": "JOIDES Resolution",
    "identifier": {
        "@id": "https://example.org/id/vessel/X",
        "@type": "PropertyValue",
        "propertyID": "https://en.wikipedia.org/wiki/IMO_number",
        "url": "https://example.org/id/vessel/X",
        "description": "Vessel ID "
    },
    "additionalProperty": {
        "@id": "ID_value_string",
        "@type": "PropertyValue",
        "propertyID": "https://en.wikipedia.org/wiki/IMO_number",
        "url": "https://foo.org/linkToPropertyIDPage",
        "description": "Any additional properties for the vessel"
    },
    "subjectOf": {
        "@type": "DataDownload",
        "name": "external-metadata.xml",
        "description": "Metadata describing the vessel",
        "encodingFormat": [
            "application/xml",
```

(continues on next page)

(continued from previous page)

```
            "https://foo.org/ship01"
        ],
        "dateModified": "2019-06-12T14:44:15Z"
    }
}
```

```
<graphviz.dot.Digraph at 0x7fe73c71b370>
```

11.1.1 Details: Authoritative Reference

For each profile there are a few key elements we need to know about. One key element is what the authoritative reference or canonical identifier is for a resource.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/X",
    "@type": "Vehicle",
    "identifier": {
        "@id": "https://example.org/id/vessel/X",
        "@type": "PropertyValue",
        "description": "Vessel ID ",
        "propertyID": "https://en.wikipedia.org/wiki/IMO_number",
        "url": "https://example.org/id/vessel/X"
    }
}
```

```
<graphviz.dot.Digraph at 0x7fe72e281dc0>
```

11.1.2 Details: subjectOf

Like SOS, we are recommending the use of subjectOf to link a simple Schema.org type to a more detailed metadata description record. This allows us to use the easy discovery layer in Schema.org but connect to domain specific metadata records.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/X",
    "@type": "Vehicle",
    "subjectOf": {
        "@type": "DataDownload",
        "dateModified": "2019-06-12T14:44:15Z",
        "description": "Metadata describing the vessel",
        "encodingFormat": [
            "application/xml",
            "https://foo.org/ship01"
        ],
        "name": "external-metadata.xml"
    }
}
```

```
<graphviz.dot.Digraph at 0x7fe72e677880>
```

11.2 References

- ICES
- POGO
- EurOcean
- https://vocab.nerc.ac.uk/search_nvs/C17/
- SeaDataNet
- Marine Facilities Planner
- EuroFleets
- Identifiers to use include NOCD Code, Call Sign, ICES Shipcode, MMSI Code, IMO Code

CHAPTER

TWELVE

SPATIAL GEOMETRY

12.1 About

The primary OIH guidance will be to use the OGC GeoSPARQL vocabulary. The schema.org spatial types and propeties are not well defined and difficult at times to reliably translate to geometries.

12.2 Simple GeoSPARQL WKT

This is a simple example of how to embed a WKT string via GeoSPARQL into a record.Well Know Text (WKT) is a OGC standard referenced at: https://www.ogc.org/standards/wkt-crs. A more accessible example can be found at WikiPedia: Well-known text representation of geometry.

```
{
    "@context": {
      "@vocab": "https://schema.org/",
      "geosparql": "http://www.opengis.net/ont/geosparql#"
    },
    "@id": "https://example.org/id/XYZ",
    "@type": "Dataset",
    "name": "Data set name",
    "geosparql:hasGeometry": {
        "@type": "http://www.opengis.net/ont/sf#Point",
        "geosparql:asWKT": {
            "@type": "http://www.opengis.net/ont/geosparql#wktLiteral",
            "@value": "POINT(-76 -18)"
        },
        "geosparql:crs": {
            "@id": "http://www.opengis.net/def/crs/OGC/1.3/CRS84"
        }
    }
}
```

```
<graphviz.dot.Digraph at 0x7fa898398a90>
```

12.3 Classic Schema.org

Is is a simple example of the existing Schema.org pattern for a lat long value.This pattern is of little use other than perhaps to Google. There is the pending GeospatialGeometry which is a type Intangible (and not Place referenced by spatialCoverage). This will be a subtype of GeoShape. There are inconsistencies with Schema.org guidance for textual geometry representation and that of Well Known Text (WKT).

```
{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@id": "https://example.org/id/XYZ",
  "@type": "Dataset",
  "name": "Data set name",
  "spatialCoverage": {
    "@type": "Place",
    "geo": {
      "@type": "GeoCoordinates",
      "latitude": 39.3280,
      "longitude": 120.1633
    }
  }
}
```

```
<graphviz.dot.Digraph at 0x7fa898398c40>
```

12.4 Option review, SOS Issue 105

From the referenced SOS issue 105:

```
{
    "@context": {
        "@version": 1.1,
        "geoblob": {
            "@id": "http://example.com/vocab/json",
            "@type": "@json"
        },
        "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
        "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
        "xsd": "http://www.w3.org/2001/XMLSchema#",
        "description": "http://igsn.org/core/v1/description",
        "geosparql": "http://www.opengis.net/ont/geosparql#",
        "schema": "https://schema.org/"
    },
    "@id": "https://samples.earth/id/do/bqs2dn2u6s73o70jdup0",
    "@type": "http://igsn.org/core/v1/Sample",
    "description": "A fake ID for testing",
    "schema:subjectOf": [
        {
            "schema:url": "https://samples.earth/id/do/bqs2dn2u6s73o70jdup0.geojson",
            "@type": "schema:DigitalDocument",
            "schema:format": [
                "application/vnd.geo+json"
            ],
```

(continues on next page)

(continued from previous page)

```
            "schema:conformsTo": "https://igsn.org/schema/spatial.schema.json"
        }
    ],
    "geosparql:hasGeometry": {
        "@id": "_:N98e75cacc29f40deb555eb583cb162dc",
        "@type": "http://www.opengis.net/ont/sf#Point",
        "geosparql:asWKT": {
            "@type": "http://www.opengis.net/ont/geosparql#wktLiteral",
            "@value": "POINT(-76 -18)"
        },
        "geosparql:crs": {
            "@id": "http://www.opengis.net/def/crs/OGC/1.3/CRS84"
        }
    },
    "geoblob": {
        "type": "GeometryCollection",
        "geometries": [{
            "type": "Point",
            "coordinates": [-76, -18]
        }]
    },
    "schema:spatialCoverage": {
        "@type": "schema:Place",
        "schema:geo": {
          "@type": "schema:GeoCoordinates",
          "schema:latitude": -18,
          "schema:longitude": -76
        }
      }
}
```

```
<graphviz.dot.Digraph at 0x7fa891303eb0>
```

12.5 References

- GeoAPI at GitHub
- Science on Schema Issue 105
  - Leverages subjectOf to connect to a Thing / CreativeWork
- https://www.unsalb.org/
- https://www.un.org/geospatial/
- schema.org/spatial
- schema.org/GeospatialGeometry
- SOS patern follows:
  - spatialCoverage -> Place -> geo -> GeoCoordinates OR GeoShape
- Some groups are using GeoNode
  - schema.org issues
- ICAN & Schema.org

- OGC SELFIE
- Think broad
- Science on Schema spatial for dataset guidance

CHAPTER

THIRTEEN

SERVICES

13.1 About

This section will provide information on the service type. This is not one of the main OIH types. However, we will provide guidance here on describing services using schema.org.

It should be noted that this might be a simple link to an OpenAPI or some other descriptor document. Also, schema.org is not rich enough for complex descriptions and itself borrows from the Hydra vocabulary. It may be required to leverage Hydra if complex descriptions are needed.

The graph describes a service than can be invoked with:

```
curl --data-binary "@yourfile.jpg" -X POST https://us-central1-top-operand-112611.
↪cloudfunctions.net/function-1
```

This with POST a jpeg to the service and get back a simple text response with some information about the image.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
      },
    "@type": "Action",
    "@id": "https://us-central1-top-operand-112611.cloudfunctions.net/function-1",
    "result": {
      "@type": "DataDownload",
      "encodingFormat": "text/plain",
      "description": "a simple text result for the RGB counts"
    },
    "target": {
      "@type": "EntryPoint",
      "urlTemplate": "https://us-central1-top-operand-112611.cloudfunctions.net/
↪function-1",
      "httpMethod": "POST",
      "contentType": ["image/jpeg", "image/png"]
    },
    "object": {
      "@type": "ImageObject",
      "description": "A JPEG or PNG to analyze the RGB counts"
    }
  }
```

```
<graphviz.dot.Digraph at 0x7fca4016e370>
```

13.2 References

- https://schema.org/docs/actions.html
- https://schema.org/Action
- https://www.w3.org/TR/web-share/
- https://www.hydra-cg.com/spec/latest/core/

CHAPTER

FOURTEEN

KEYWORDS AND DEFINED TERMS

14.1 About

This section is looking at how the keywords could be connected with Defined Terms that point to external vocabularies that follow a vocabulary publishing patterns like at the W3C Best Practice Recipes for Publishing RDF Vocabularies.

The pattern breaks down a bit when attempting to connect with things like the Global Change Master Directory keywords.

A person could adapt the pattern to connect things like: EARTH SCIENCE > OCEANS > OCEAN CHEMISTRY. This does have a UUID (6eb3919b-85ce-4988-8b78-9d0018fd8089) but this is not a dereference-able PID.

14.2 Keywords

We can see three different approaches here to defining keywords.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/XYZ",
    "@type": "Map",
    "keywords": [
        {
            "@id": "http://purl.org/dc/dcmitype/Image",
            "@type": "DefinedTerm",
            "inDefinedTermSet": "http://purl.org/dc/terms/DCMIType",
            "name": "Image",
            "termCode": "Image"
        },
        "Region X",
        {
            "@id": "https://www.wikidata.org/wiki/Q350134",
            "@type": "URL",
            "name": "North Atlantic Ocean",
            "url": "https://www.wikidata.org/wiki/Q350134"
        }
    ]
}
```

```
<graphviz.dot.Digraph at 0x7fc97dc26b80>
```

14.3 Defined Terms

During generation of the structured data a provide may wish to either use or publish a set of controlled vocabulary terms or a similar set.

Within schema.org this could be done by leveraging the "DefinedTerm" amd "DefinedTermSet" types.

These types allow us both to define a set of terms and use a set of terms in describing a thing.

Note that DefinedTerm is an intangible and can connect to most types in Schema.org. So we can use them in places such as:

- CreativeWork -> keyword
- LearningResource -> teaches
- PropertyValue -> valueReference
- LearningResource -> competencyRequired
- CreativeWork -> learningResourceType

```
[
        {
                "@context": {
                        "@vocab": "https://schema.org/"
                }
        },
        {
                "@type": "DefinedTermSet",
                "@id": "http://openjurist.org/dictionary/Ballentine",
                "name": "Ballentine's Law Dictionary"
        },
        {
                "@type": "DefinedTerm",
                "@id": "http://openjurist.org/dictionary/Ballentine/term/calendar-year
↪",
                "name": "calendar year",
                "description": "The period from January 1st to December 31st,␣
↪inclusive, of any year.",
                "inDefinedTermSet": "http://openjurist.org/dictionary/Ballentine"
        },
        {
                "@type": "DefinedTerm",
                "@id": "http://openjurist.org/dictionary/Ballentine/term/schema",
                "name": "schema",
                "description": "A representation of a plan or theory in the form of␣
↪an outline or model.",
                "inDefinedTermSet": "http://openjurist.org/dictionary/Ballentine"
        }
]
```

14.4 References

- schema.org/DefinedTerm
- schema.org/DefinedTermSet

CHAPTER

FIFTEEN

LANGUAGES

15.1 About

JSON-LD fully support the identification of the language types.

Properties such as label, description, keyword etc can be extended in the context with a container language attribute notiation.

This will allow the use of standard language codes (fr, es, en, de, etc) to be used when describing these properties.

```
{
    "@context": {
      "vocab": "http://example.com/vocab/",
      "label": {
        "@id": "vocab:label",
        "@container": "@language"
      }
    },
    "@id": "http://example.com/queen",
    "label": {
      "en": "The Queen",
      "de": [ "Die Königin", "Ihre Majestät" ]
    }
  }
```

```
<graphviz.dot.Digraph at 0x7f580dc07b20>
```

In graph space the resulting triples from the above are:

```
<http://example.com/queen> <http://example.com/vocab/label> "Die Königin"@de .
<http://example.com/queen> <http://example.com/vocab/label> "Ihre Majestät"@de .
<http://example.com/queen> <http://example.com/vocab/label> "The Queen"@en .
```

with language encoding attributes in place. These can be used in searching and result filters.

Note, this can cause issues in query space since the concept of

```
"The Queen"
```

and

```
"The Queen"@en
```

are different and so care must be taken the creation of the SPARQL queries not to accidentally imposed implicate filters through the use of language types.

CHAPTER

SIXTEEN

LINKING TO DOCUMENTS AND RESOURCES

Leveraging the ability to link between resources can serve many goals. We may wish to demonstrate connections between people and courses they have taken or or organizations they are connected with. We may be wishing to link documents to people or organizations.

This section will review two key thematic profiles and some examples of how to express links from them to other resources. Our goal will be different in various cases. The two profiles are type CreativeWork and type Organization.

In the case of *Organization our purpose may be to express alignment to various principles and policies*. These might provide people with an understanding of the goals of an organization when they are searching for or assessing them.

In the case of *CreativeWork we are looking to express connections to the publisher and provider of the creative work*. This is mostly to connect these works with the responsible party associated with them but may also serve to connect to the principles they are associated with.

16.1 Organization link options

In the following section we will look at three different options for expressing links between an organization and resources that describe the policy and principles of the subject organization.

First we will see the full data graph. We have highlighted the sections we we will review here. Namely the subjectOF and publishingPrinciples predicates.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/org/x",
    "@type": "Organization",
    "address": {
        "@type": "PostalAddress",
        "addressLocality": "Paris, France",
        "postalCode": "F-75002",
        "streetAddress": "38 avenue de l'Opera"
    },
    "email": "secretariat(at)example.org",
    "name": "Organization X",
    "description": "Description of org ...",
    "telephone": "( 33 1) 42 68 53 00",
    "member": [
        {
            "@type": "Organization",
            "name": "Organization A",
```

(continues on next page)

(continued from previous page)

```
            "description": "Org A is a potential parent organization of Org X"
        }
    ],
    "identifier": {
        "@id": "https://grid.ac/institutes/grid.475727.4",
        "@type": "PropertyValue",
        "description": "UN Department of Economic and Social Affairs Sustainable Development",
        "propertyID": "https://registry.identifiers.org/registry/grid",
        "url": "https://grid.ac/institutes/grid.475727.4"
    },
    "subjectOf": {
        "@type": "CreativeWork",
        "@id": "http://purl.unep.org/sdg/SDGIO_00020173",
        "url": "http://www.ontobee.org/ontology/SDGIO?iri=http://purl.unep.org/sdg/SDGIO_00020173",
        "name": "UNSD SDG indicator code:C140c01",
        "description": "Number of countries making progress ... the oceans and their resources"
    },
    "publishingPrinciples": [
        {
            "@type": "CreativeWork",
            "@id": "https://sdgs.un.org/goals/goal14",
            "url": "https://sdgs.un.org/goals/goal14",
            "name": "Sustainable Development Goal 14",
            "description": "Conserve and sustainably use the oceans, seas and marine resources for sustainable development"
        },
        {
            "@type": "CreativeWork",
            "@id": "https://dx.doi.org/10.1038/sdata.2016.18",
            "url": "https://www.nature.com/articles/sdata201618",
            "name": "FAIR data principles",
            "description": "FAIR Principles definition as referenced from: Wilkinson, M. D. et al. The FAIR Guiding Principles for scientific data management and stewardship."
        }
    ]
}
```

```
<graphviz.dot.Digraph at 0x7f484d005b20>
```

16.1.1 subjectOf

Lets take a look at subjectOf. In this case we are using subjectOf to express a connection to a UN SDG. This, subjectOf, could also be used to connect documents describing the policy and principles of an organization or additional metadata for a creative work. When we look at subjectOf we can see we are allowed are allowed to use it on any type Thing, but must point to a CreativeWork or Event.

Note: Recall that in the case of OIH types, the type CourseInstance or EducationEvent are both subtype of Event. Given that we can use subjectOf to connect a Thing to these types as well. Also, Course is a subtype of CreativeWork, so we are good there too in the context of the range of subjectOf. Refernece thematic type *Training*

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/org/x",
    "@type": "Organization",
    "subjectOf": {
        "@id": "http://purl.unep.org/sdg/SDGIO_00020173",
        "@type": "CreativeWork",
        "description": "Number of countries making progress ... the oceans and their␣
↪resources",
        "name": "UNSD SDG indicator code:C140c01",
        "url": "http://www.ontobee.org/ontology/SDGIO?iri=http://purl.unep.org/sdg/
↪SDGIO_00020173"
    }
}
```


```
<graphviz.dot.Digraph at 0x7f4837b85dc0>
```

16.1.2 publishingPrinciples

Lets take a look at publishingPrinciples.


```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/org/x",
    "@type": "Organization",
    "publishingPrinciples": [
        {
            "@id": "https://sdgs.un.org/goals/goal14",
            "@type": "CreativeWork",
            "description": "Conserve and sustainably use the oceans, seas and marine␣
↪resources for sustainable development",
            "name": "Sustainable Development Goal 14",
            "url": "https://sdgs.un.org/goals/goal14"
        },
        {
            "@id": "https://dx.doi.org/10.1038/sdata.2016.18",
            "@type": "CreativeWork",
            "description": "FAIR Principles definition as referenced from: Wilkinson,␣
↪M. D. et al. The FAIR Guiding Principles for scientific data management and␣
↪stewardship.",
            "name": "FAIR data principles",
            "url": "https://www.nature.com/articles/sdata201618"
        }
    ]
}
```


```
<graphviz.dot.Digraph at 0x7f4837b85640>
```

16.2 SDG Linkage

The following provides an example of how Sustainable Development Goals (SDGs) could be linked to a Schema.org defined type using subjectOf.

```
{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@type": "CreativeWork",
  "@id": "https://example.org/id/XYZ",
  "name": "Name or title of the document",
  "description": "Description of the resource to aid in searching",
  "distribution": {
    "@type": "DataDownload",
    "contentUrl": "https://www.sample-data-repository.org/dataset/472032.tsv",
    "encodingFormat": "text/tab-separated-values"
  },
  "subjectOf": {
    "@type": "CreativeWork",
    "@id": "http://purl.unep.org/sdg/SDGIO_00020173",
    "url": "http://www.ontobee.org/ontology/SDGIO?iri=http://purl.unep.org/sdg/SDGIO_
↪00020173",
    "name": "UNSD SDG indicator code:C140c01",
    "description": "Number of countries making progress ... the oceans and their␣
↪resources"
  },
  "maintainer" : {
     "@type" : "Organization",
     "@id": "https://ror.org/050bms902",
     "description": "UN Department of Economic and Social Affairs Sustainable␣
↪Development"
  }
}
```

```
<graphviz.dot.Digraph at 0x7f4837f7fee0>
```

16.3 Creative work link options

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/XYZ",
    "@type": "CreativeWork",
    "identifier": {
        "@id": "https://doi.org/10.5066/F7VX0DMQ",
        "@type": "PropertyValue",
        "propertyID": "https://registry.identifiers.org/registry/doi",
        "url": "https://doi.org/10.5066/F7VX0DMQ",
        "value": "doi:10.5066/F7VX0DMQ"
    },
    "subjectOf": {
        "@type": "CreativeWork",
```

(continues on next page)

(continued from previous page)

```
        "@id": "http://purl.unep.org/sdg/SDGIO_00020173",
        "url": "http://www.ontobee.org/ontology/SDGIO?iri=http://purl.unep.org/sdg/SDGIO_00020173",
        "name": "UNSD SDG indicator code:C140c01",
        "description": "Number of countries making progress ... the oceans and their resources"
    },
    "publishingPrinciples": {
        "@type": "CreativeWork",
        "@id": "http://purl.unep.org/sdg/SDGIO_00020173",
        "url": "http://www.ontobee.org/ontology/SDGIO?iri=http://purl.unep.org/sdg/SDGIO_00020173",
        "name": "FAIR data principles",
        "description": "Number of countries making progress ... the oceans and their resources"
    },
    "citation": {
        "@type": "CreativeWork",
        "@id": "http://purl.unep.org/sdg/SDGIO_00020173",
        "url": "http://www.ontobee.org/ontology/SDGIO?iri=http://purl.unep.org/sdg/SDGIO_00020173",
        "name": "UNSD SDG indicator code:C140c01",
        "description": "Number of countries making progress ... the oceans and their resources"
    },
    "provider": {
        "@id": "https://www.sample-data-repository.org",
        "@type": "Organization",
        "legalName": "Sample Data Repository Office",
        "name": "SDRO",
        "sameAs": "http://www.re3data.org/repository/r3dxxxxxxxxx",
        "url": "https://www.sample-data-repository.org"
    },
    "publisher": {
        "@id": "https://www.sample-data-repository.org"
    }
}
```

```
<graphviz.dot.Digraph at 0x7f4837beb6a0>
```

16.4 Refs

- SDGs
- SDG targets
- SDG indicators

CHAPTER

SEVENTEEN

IDENTIFIER

17.1 About

When using an identifier section, we are suggesting to use a PropertyValue node. In this we can then present several elements in context. These are highlighted below.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": "Map",
    "@id": "https://example.org/id/XYZ",
    "name": "Name or title of the document",
    "description": "Description of the map to aid in searching",
    "url": "https://www.sample-data-repository.org/creativework/map.pdf",
    "identifier": {
        "@id": "https://doi.org/10.5066/F7VX0DMQ",
        "@type": "PropertyValue",
        "propertyID": "https://registry.identifiers.org/registry/doi",
        "value": "doi:10.5066/F7VX0DMQ",
        "url": "https://doi.org/10.5066/F7VX0DMQ"
    },
    "keywords": {
        "@type": "DefinedTerm",
        "inDefinedTermSet": {
            "@type": "DefinedTermSet",
            "name": "Name of the set",
            "description": "Description of the set",
            "url": "url for the set"
        },
        "termCode": "A code that identifies this DefinedTerm within a DefinedTermSet"
    }
}
```

17.2 Properties of interest

propertyID : A commonly used identifier for the characteristic represented by the property, e.g. a manufacturer or a standard code for a property. propertyID can be (1) a prefixed string, mainly meant to be used with standards for product properties; (2) a site-specific, non-prefixed string (e.g. the primary key of the property or the vendor-specific id of the property), or (3) a URL indicating the type of the property, either pointing to an external vocabulary, or a Web resource that describes the property (e.g. a glossary entry). Standards bodies should promote a standard prefix for the identifiers of properties from their standards.

value : The value of the quantitative value or property value node. For PropertyValue, it can be 'Text;', 'Number', 'Boolean', or 'StructuredValue'.

url (technically type URL) : URL of the item.

17.3 Frame and view "identifier" section

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/XYZ",
    "@type": "Map",
    "identifier": {
        "@id": "https://doi.org/10.5066/F7VX0DMQ",
        "@type": "PropertyValue",
        "propertyID": "https://registry.identifiers.org/registry/doi",
        "url": "https://doi.org/10.5066/F7VX0DMQ",
        "value": "doi:10.5066/F7VX0DMQ"
    }
}
```

```
<graphviz.dot.Digraph at 0x7fb6d455eb80>
```

Part III

Aggregation

CHAPTER

EIGHTEEN

AGGREGATOR

18.1 Intoduction

This section introduces the the OIH approach to indexing. Currently, OIH is using the Gleaner software to do the indexing and leverages the Gleaner IO gleaner-compose Docker Compose files for the server side architecture. For more information on Docker Compose files visit the Overiew of Docker Compose. The gleaner-compose repository holds Docker compose files that can set up various environments that Gleaner needs.

The figure below gives a quick overview of the various compose options for setting up the supporting architecture for Gleaner. A fully configured system where all the indexing and data services are running and exposing services to the net, a total of five containers are run. In many case you may run fewer than this.

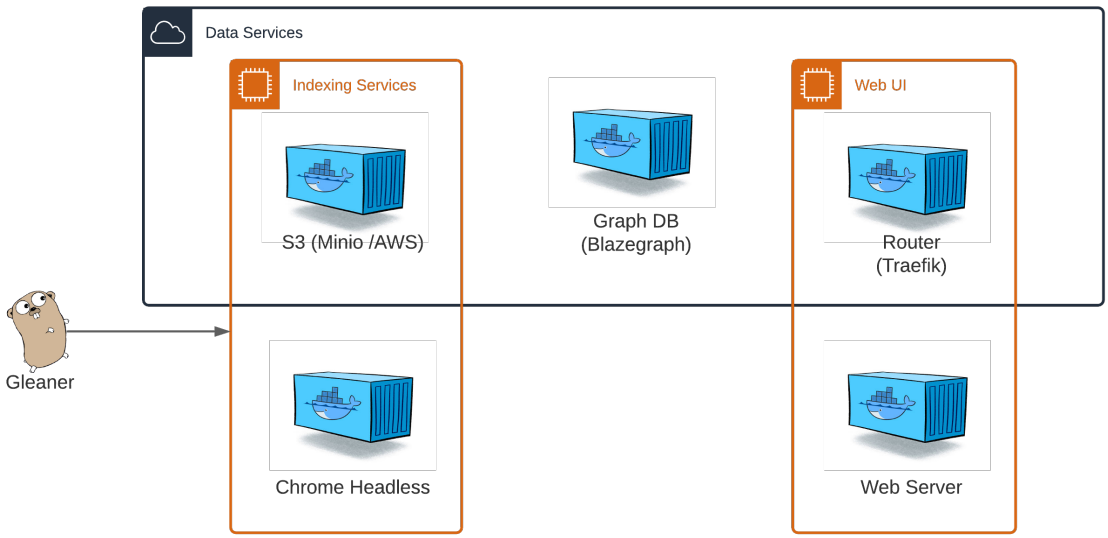


Fig. 1: The various compose options for Gleaner

18.1.1 Container overview

- S3 (Minio / AWS): This is the only container that is required in all cases to run. Gleaner needs an S3 compatible object store. By default we use the Minio object store
- Chrome Headless: In cases where providers place the JSON-LD documents into the pages with Javascript, we need to render the page before reading and accessing the DOM. This is done using Chrome Headless
- Graph data base: Gleaner extracts JSON-LD documents from resources. These JSON-LD documents are representations of the RDF data mode. To queries on them at scale, it easiest to load the triples into a compatible graph database. Sometimes we call this a triplestore. For OIH we use the Blazgraph triplestore.
- Router: If we wish to deploy this setup onto the net, we will route to route all the services through a single domain. To do this network routing we use Traefik. This router is not required for local use and alternative routers like Caffdy or nginx are also valid options.
- Web Server: If you wish to serve a web UI for the index, then you can also leverage this setup to serve that. Again, this is optional and your web site may be hosted elswhere and simply call to the index in compliance with CORS settings. There is an example web server that leverages the object store available in this setup.

18.1.2 Gleaner

As mentioned Gleaner is a single binary app (ie, one file). It can be run on Linux, Mac OS X or Windows. It does not need to be run on the same machine as the supporting services as it can connect to them over the network. So, for example, they could be hosted in commercial cloud services or on remote servers.

You can download and compile the code from the previously mentioned github repository or the releases page.

A single configuration file provides the settings Gleaner needs. Additionally, a local copy of the current schema.org context file should be downloaded and available to the app. This file is needed for many operations and access it over the net is slow and often rate limited depending on the source. You can download the file at the Schema.org for Developers page.

This setup show in the above figure is the typical setup for Gleaner and is detailed in the *Quick Start* section.

18.1.3 Indexing Services

This is the basic indexing service requirements. At a minimum we need the object store and the Chrome headless containers scoped in the *Indexing Sevices* box above. More details on this set can be found in *Indexing Servives*.

18.1.4 Data Services

A more expanded set of services is defined in the *Data Services* section. This section discussion a setup more designed to address a server setup tht will support indexing and also present the resulting indexes to the broader internet.

18.1.5 Web UI

As mentioned, if you wish to serve a web UI for the index, then you can leverage this setup to serve that. Again, this is optional and your web site can be hosted elsewhere and simply call to the index in compliance with CORS settings. Some tailes on this can be found in the *Interfaces* section.

18.1.6 Alternatives

Note, the Gleaner ecosystem is not a requirement. OIH follows the structured data on the web and data on the web best practices patterns. Being web architecture based, there are many open source tools and scripting solutions you might use. You may wish to explore the *Alternative Approaches* section for more on this.

What follows is a bit more detail on the setup used by Gleaner. Experienced users will see where they can swap out elements for their own preference. Like a different triplestore, or wish to leverage a commercial object store? Simply modify the architecture to do so.

18.2 ODIS Catalog as Index Source

Before we discuss indexing source a key question is what source will be indexed. OIH is not a web crawl in that it doesn't move from source to source based on the content of those sources.

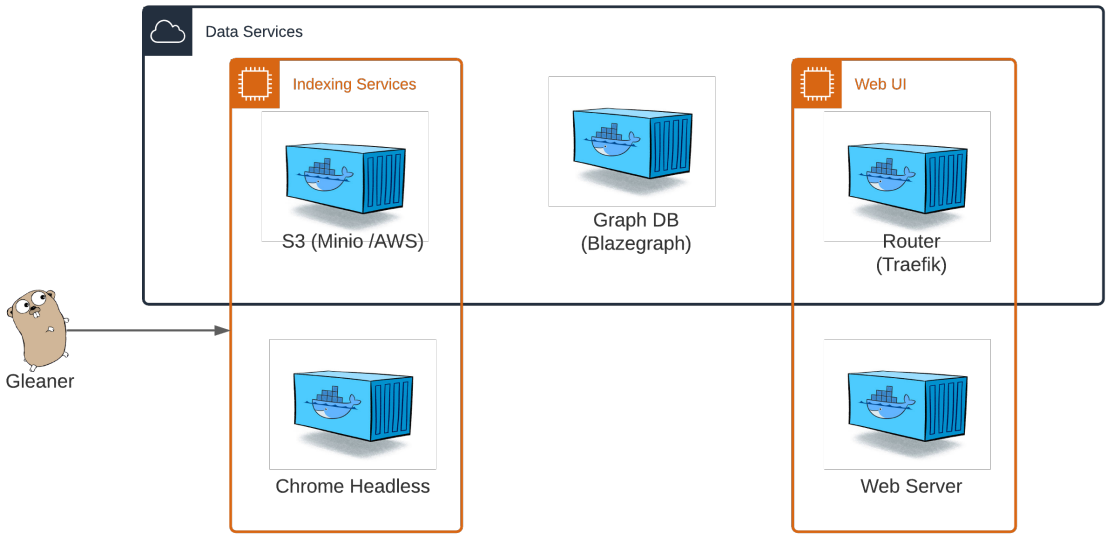
Rather, the OIH index is based on a list of sources selected ahead of time. At this time that set of sources if based on those partners engaged in the development phase of OIH. As the work moves to a more routine operation the sources will come from the ODIS Catalog.

The ODIS Catalog will then act as a curated source of domains for inclusion in the Ocean InfoHub. This will provide a level of curation and vetted of sources and ensure sources are aware of the technical requirements for inclusion in the OIH index.

CHAPTER

NINETEEN

INDEXING WITH GLEANER



19.1 Gleaner (app)

The Gleaner applications performs the retrieval and loading of JSON-LD documents from the web following structured data on the web patterns. Gleaner is available for Linux, Mac OS X and Windows.

While Gleaner is a stand alone app, it needs to interact with an object store to support data storage and other operations. These dependencies are met within the Gleaner Indexing Services or Data Service Docker compose files.

Warning: This documentation is in development. The primary testing environments are Linux and other UNIX based platforms such as Mac OS X. If you are on Windows, there may be some issues. If you can use a Linux subsystem on Windows, you may experience better results. We will test with Windows eventually and update documentation as needed.

19.1.1 Quick Start steps

This quick start guide is focused on setting up and testing Gleaner in a local environnement. It is similar to how you might run Gleaner in a production environment but lacks the routing and other features likely desired for such a situation.

Note: This documentation assumes a basic understanding of Docker and experience with basic Docker activities like starting and stopping containers. It also assumes an understanding of using a command line interface and editing configuration files in the YAML format.

Command

From this point down, the documentation will attempt to put all commands you should issue in this admonition style box.

In the end, this is the table of applications and config files you will need. In this guide we will go through downloading, setting them up and running Gleaner to index documents from the web.

Table 1: Required Applications and Their Config Files

Gleaner	Docker	Minio Client
config.yaml	`setenv.sh`	`load2blaze.sh`
schemaorg-current-https.jsonld	gleaner-DS-NoRouter.yml	

Grab Gleaner and the support files we need

We will need to get the Gleaner binary for your platform and also the Gleaner configuration file template. To do this, visit the Gleaner Releases page and pick the release *Ocean InfoHubdev rc1*. Under the *Assets* drop down you should see the files we need. Get:

- Gleaner for your platform
- Gleaner config template: template_v2.0.yaml
- Gleaner indexing service compose file: gleaner-IS.yml
- Helper environment setup script: setenvIS.sh

For this demonstration, we will be running on linux, so this would look something like:

Command

```
curl -L -O https://github.com/earthcubearchitecture-project418/gleaner/releases/
 ↪download/2.0.25/gleaner
curl -L -O https://github.com/earthcubearchitecture-project418/gleaner/releases/
 ↪download/2.0.25/gleaner-IS.yml
curl -L -O https://github.com/earthcubearchitecture-project418/gleaner/releases/
 ↪download/2.0.25/setenvIS.sh
curl -L -O https://github.com/earthcubearchitecture-project418/gleaner/releases/
 ↪download/2.0.25/template_v2.0.yaml
```

Note: You can download these with any tool you wish or through the browser. Above we downloaded used the command line curl tool. For GitHub, be sure to add the -L to inform curl to follow redirects to the object to download.

Command

You may need to change the permission on your gleaner file to ensure it can be run. On Linux this would look something like the following.

```
chmod 755 gleaner
```

We then need to visit Schema.org for Developers to pull down the appropriate JSON-LD context. For this work we will want to pull down the *schemaorg-current-https* in JSON-LD format.It also should work to do something similar to the following:

Command

```
curl -O https://schema.org/version/latest/schemaorg-current-https.jsonld
```

About the compose file(s)

The above steps have collected the resources for the indexer. We now want to set up the services that Gleaner will use to perform the indexing. To do that we use Docker or an appropriate run time alternative like Podman or others. For this example, we will assume you are using the Docker client.

As noted, a basic understanding of Docker and the ability to issue Docker cli commands to start and stop containers is required. If you are new do Docker, we recommend you visit and read: Get Started with Docker.

We need to select the type of services we wish to run. The various versions of these Docker compose file can be found in the Gleaner-compose deployment directory.

Why pick one over the other?

> Choose Gleaner IS if you simply wish to retrieve the JSON-LD into a data warehouse to use in your own workflows
>
> Choose Gleaner DS if you wish to build out a graph and want to use the default contains used by Gleaner.

Note: We wont look at this file in detail here since there will hopefully be no required edits. You can see the file in detail in the Index Services section.

Edit environment variables setup script

We have Docker and the appropriate compose file. The compose files require a set of environment variables to be populated to provide the local hosts information needed to run. You can set these yourself or use or reference the setenv.sh file in the Gleaner-compose repository in theGleaner-compose deployment directory. You may also need to visit information about permissions at Post-installation steps for Linux if you are having permission issues.

Let's take a look at the script.

```
1 #!/bin/bash
2
3 # Object store keys
4 export MINIO_ACCESS_KEY=worldsbestaccesskey
5 export MINIO_SECRET_KEY=worldsbestsecretkey
```

(continues on next page)

(continued from previous page)

```

# local data volumes
export GLEANER_BASE=/tmp/gleaner/
mkdir -p ${GLEANER_BASE}
export GLEANER_OBJECTS=${GLEANER_BASE}/datavol/s3
export GLEANER_GRAPH=${GLEANER_BASE}/datavol/graph
```

You may wish to edit file to work better with your environment. By default it will attempt to use localhost to resolve with and host local runtime data in a /tmp/gleaner directory.

Spin up the containers

Load our environment variables to the shell:

Command

```
source setenv.sh
```

Then start the containers:

Command

```
docker-compose -f gleaner-IS.yml up -d
```

If all has gone well, you should be able to see your running containers with

Command

```
docker ps
```

and see results similar to:

```
CONTAINER ID        IMAGE                                COMMAND                  CREATED␣
 ↪            STATUS              PORTS                     NAMES
c4b7097f5e06        nawer/blazegraph                     "docker-entrypoint.s…"   8␣
 ↪seconds ago       Up 7 seconds        0.0.0.0:9999->9999/tcp   test_triplestore_1
ca08c24963a0        minio/minio:latest                   "/usr/bin/docker-ent…"   8␣
 ↪seconds ago       Up 7 seconds        0.0.0.0:9000->9000/tcp   test_s3system_1
24274eba0d34        chromedp/headless-shell:latest       "/headless-shell/hea…"   8␣
 ↪seconds ago       Up 7 seconds        0.0.0.0:9222->9222/tcp   test_headless_1
```

Edit Gleaner config file

We have all the files we need and we have our support services running. The next and final step is to edit our Gleaner configuration file. This will let Gleaner know the location of the support services, the JSON-LD context file and the locations of the resources we wish to index.

Let's take a look at the full configuration file first and then break down each section.

```
---
minio:
  address: 0.0.0.0
  port: 9000
  accessKey: worldsbestaccesskey
  secretKey: worldsbestsecretkey
  ssl: false
  bucket: gleaner
gleaner:
  runid: oih # this will be the bucket the output is placed in...
  summon: true # do we want to visit the web sites and pull down the files
  mill: true
context:
  cache: true
contextmaps:
- prefix: "https://schema.org/"
  file: "./jsonldcontext.json"  # wget http://schema.org/docs/jsonldcontext.jsonld
- prefix: "http://schema.org/"
  file: "./jsonldcontext.json"  # wget http://schema.org/docs/jsonldcontext.jsonld
summoner:
  after: ""       # "21 May 20 10:00 UTC"
  mode: full  # full || diff:  If diff compare what we have currently in gleaner to␣
↪sitemap, get only new, delete missing
  threads: 1
  delay: 0  # milliseconds (1000 = 1 second) to delay between calls (will FORCE␣
↪threads to 1)
  headless: http://0.0.0.0:9222  # URL for headless see docs/headless
millers:
  graph: true
  #geojson: false
sitegraphs:
- name: aquadocs
  url: https://oih.aquadocs.org/aquadocs.json
  headless: false
  pid: https://www.re3data.org/repository/aquadocs
  properName: AquaDocs
  domain: https://aquadocs.org
sources:
- name: samplesearth
  url: https://samples.earth/sitemap.xml
  headless: false
  pid: https://www.re3data.org/repository/samplesearth
  properName: Samples Earth (DEMO Site)
  domain: https://samples.earth
- name: marinetraining
  url: https://www.marinetraining.eu/sitemap.xml
  headless: false
  pid: https://www.re3data.org/repository/marinetraining
  properName: Marine Training EU
  domain: https://marinetraining.eu/
```

(continues on next page)

(continued from previous page)

```
- name: marineie
  url: http://data.marine.ie/geonetwork/srv/eng/portal.sitemap
  headless: true
  pid: https://www.re3data.org/repository/marineie
  properName: Marine Institute Data Catalogue
  domain: http://data.marine.ie
- name: oceanexperts
  url: https://oceanexpert.org/assets/sitemaps/sitemapTraining.xml
  headless: false
  pid: https://www.re3data.org/repository/oceanexpert
  properName: OceanExpert UNESCO/IOC Project Office for IODE
  domain: https://oceanexpert.org/
# - name: obis
#   url: https://obis.org/sitemap/sitemap_datasets.xml
#   headless: false
#   pid: https://www.re3data.org/repository/obis
#   properName: Ocean Biodiversity Information System
#   domain: https://obis.org
```

Object store

```
minio:
  address: 0.0.0.0
  port: 9000
  accessKey: worldsbestaccesskey
  secretKey: worldsbestsecretkey
  ssl: false
  bucket: gleaner
```

The minio section defines the IP and port of the object store. For this case, we are using minio and these are the IP and port from our docker compose steps above. Note, if you were to use Ceph or AWS S3, this section is still labeled minio. You simply need to update the property values.

Gleaner

```
gleaner:
  runid: oih # this will be the bucket the output is placed in...
  summon: true # do we want to visit the web sites and pull down the files
  mill: true
```

This passes a few high level concpets.

- runid:
- summon
- mill

Context sections

```
context:
  cache: true
contextmaps:
- prefix: "https://schema.org/"
  file: "./jsonldcontext.json"  # wget http://schema.org/docs/jsonldcontext.jsonld
- prefix: "http://schema.org/"
  file: "./jsonldcontext.json"  # wget http://schema.org/docs/jsonldcontext.jsonld
```

Comments for the context sections

Summoner section

```
summoner:
  after: ""       # "21 May 20 10:00 UTC"
  mode: full  # full || diff:  If diff compare what we have currently in gleaner to␣
↪sitemap, get only new, delete missing
  threads: 1
  delay: 0  # milliseconds (1000 = 1 second) to delay between calls (will FORCE␣
↪threads to 1)
  headless: http://0.0.0.0:9222  # URL for headless see docs/headless
```

Comments for the summoner sections

Millers section

```
millers:
  graph: true
  #geojson: false
```

Comments for the miller sections

Site graphs section

```
sitegraphs:
- name: aquadocs
  url: https://oih.aquadocs.org/aquadocs.json
  headless: false
  pid: https://www.re3data.org/repository/aquadocs
  properName: AquaDocs
  domain: https://aquadocs.org
```

Comments for the sitegrpah sections

Sources section

```
sources:
- name: samplesearth
  url: https://samples.earth/sitemap.xml
  headless: false
  pid: https://www.re3data.org/repository/samplesearth
  properName: Samples Earth (DEMO Site)
  domain: https://samples.earth
- name: marinetraining
  url: https://www.marinetraining.eu/sitemap.xml
  headless: false
  pid: https://www.re3data.org/repository/marinetraining
  properName: Marine Training EU
  domain: https://marinetraining.eu/
- name: marineie
  url: http://data.marine.ie/geonetwork/srv/eng/portal.sitemap
  headless: true
  pid: https://www.re3data.org/repository/marineie
  properName: Marine Institute Data Catalogue
  domain: http://data.marine.ie
- name: oceanexperts
  url: https://oceanexpert.org/assets/sitemaps/sitemapTraining.xml
  headless: false
  pid: https://www.re3data.org/repository/oceanexpert
  properName: OceanExpert UNESCO/IOC Project Office for IODE
  domain: https://oceanexpert.org/
# - name: obis
#   url: https://obis.org/sitemap/sitemap_datasets.xml
#   headless: false
#   pid: https://www.re3data.org/repository/obis
#   properName: Ocean Biodiversity Information System
#   domain: https://obis.org
```

Comments for the sources sections

Run gleaner

For this example we are going to run Gleaner directly. In a deployed instance you may run Gleaner via a script or cron style service. We will document that elsewhere.

We can do a quick test of the setup.

Command

```
./gleaner -cfg template_v2.0 -setup
```

For now, we are ready to run Gleaner. Try:

Command

```
./gleaner -cfg template_v2.0
```

Note: Leave the suffix like .yaml off the name of the config file. The config system can also read json and other formats. So simply leave the suffix off and let the config code inspect the contents.

19.1.2 Load results to a graph and test

You have set up the server environment and Gleaner and done your run. Things look good but you don't have a graph you can work with yet. You need to load the JSON-LD into the triplestore in order to start playing.

Minio Object store

To view the object store you could use your browser and point it on the default minio port at 9000. This typically something like localhost:9000.

If you wish to continue to use the command line you can use the Minio client at Minio Client Quickstart guide.

Once you have it installed and working, you can write an entry for our object store with:

Command

```
./mc alias set minio http://0.0.0.0:9000 worldsbestaccesskey worldsbestsecretkey
```

Load Triplestore

We now want to load these objects, which are JSON-LD files holding RDF based graph data, into a graph database. We use the term, triplestore, to define a graph database designed to work with the RDF data model and provide SPARQL query support over that graph data.

- Simple script loading
- Nabu
- Try out a simple SPARQL query

19.2 References

The following are some reference which may provide more information on the various technologies used in this approach.

- Google: Understanding how structured data works
- Google Dataset Search By the Numbers
- Google Dataset Search: Building a search engine for datasets in an open Web ecosystem
- W3C SPARQL
- SHACL
- Triplestores

CHAPTER

TWENTY

INDEXING SERVICES

Gleaner can not run alone and relies on a couple of Open Container Initiative (OCI) containers to support it. For this document, we will assume you are using Docker but this will work with Podman or other OCI compliant orchestration environments. These Gleaner Indexing Services are necessary to use Gleaner. The exception to this would be if you are using a 3rd party objects store like AWS S3 or Wasabi.

- Object Store An S3 compliant object store supporting S3 APIs including S3Select. For open source this is best satisfied with the Minio Object Store. For commercial cloud AWS S3 or hosted Ceph services will work.
- Headless Chrome (technically optional) This is only needed where you expect the sources you index to use Javascript to include the JSON-LD in the pages. If you know your sources do not use this publishing pattern and rather include the JSON-LD in the static page, then you don't need this container running.

IS represent the minimum required services to support Gleaner. With IS you have an object store in the form of Minio and a headless chrome server in the form of chromedp/headless-shell.

As shown in the figure below, and support the basic harvesting of resources with Gleaner and loading the JSON-LD objects into Minio.

It does not result in these objects ending up in a graph / triplestore. You would use this option if you intend to work on the JSON-LD objects yourself. Perhaps loading them into a alternative graphdb like Janus or working on them with python tooling.

Gleaner Indexing Services (IS) Environment Variables The Docker Compose file used to launch the Gleaner IS has a set of configurable elements that can be set and passed to the orchestration system with environment variables.

These can be set manually or through the command line. A simple script to set the variables could look like:

```
#!/bin/bash

# domains
export GLEANER_ADMIN_DOMAIN=admin.local.dev
export GLEANER_OSS_DOMAIN=oss.local.dev
export GLEANER_GRAPH_DOMAIN=graph.local.dev
export GLEANER_WEB_DOMAIN=web.local.dev
export GLEANER_WEB2_DOMAIN=web2.local.dev

# Object store keys
export MINIO_ACCESS_KEY=worldsbestaccesskey
export MINIO_SECRET_KEY=worldsbestsecretkey

# local data volumes
export GLEANER_BASE=/tmp/gleaner/
export GLEANER_TRAEFIK=${GLEANER_BASE}/config
export GLEANER_OBJECTS=${GLEANER_BASE}/datavol/s3
export GLEANER_GRAPH=${GLEANER_BASE}/datavol/graph
```

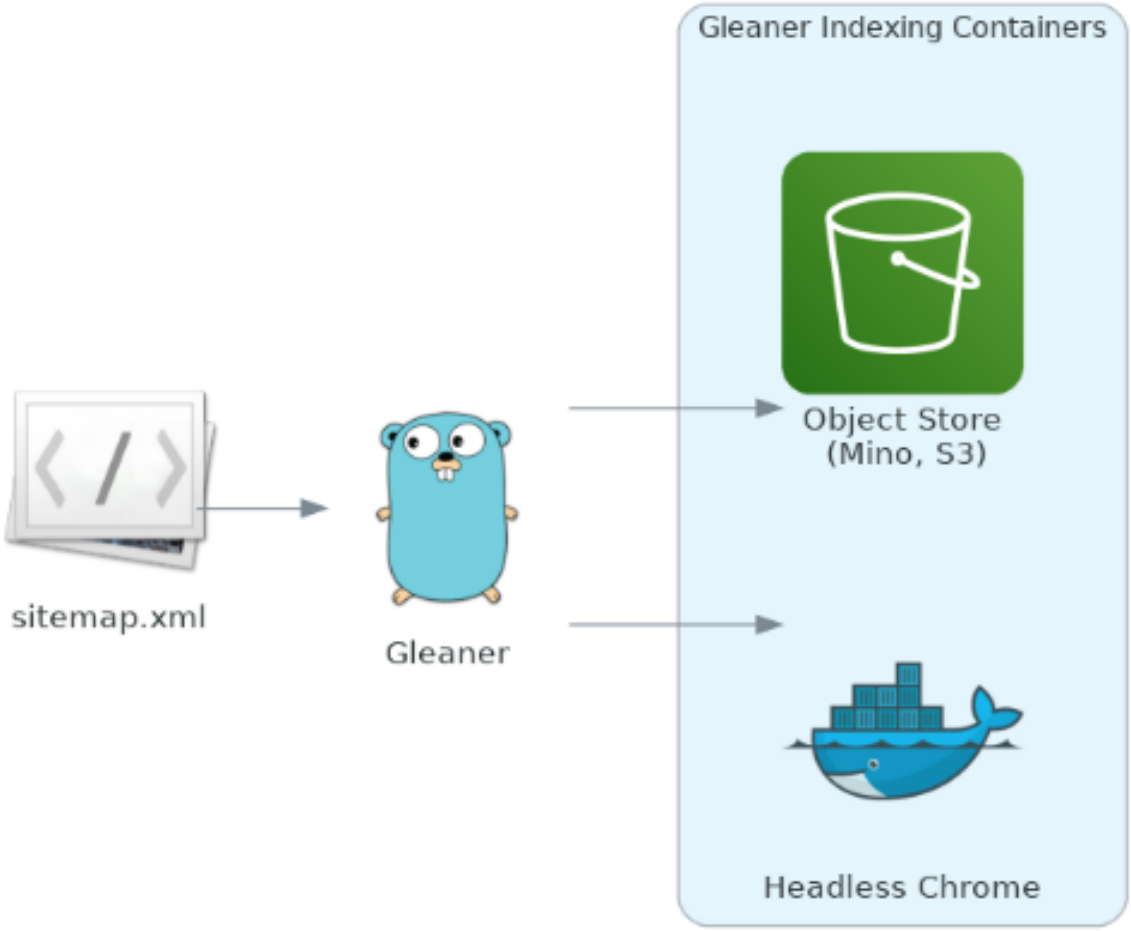


Fig. 1: Basic Gleaner Indexing Service Activity Workflow

The actual services can be deployed via a Docker Compose file (also works with Podman). An example of that file and details about it follow.

– Break down the compose files here link to them

CHAPTER

TWENTYONE

DATA SERVICES

The typical functional goal of this work is the development and use of a Graph that can be accessed via a triplestore (Graph Database). To do that we need a set of additional containers to support this and expose these services on the web through a single domain with https support.

- Object Store An S3 compliant object store supporting S3 APIs including S3Select. For open source this is best satisfied with the Minio Object Store. For commercial cloud AWS S3 or hosted Ceph services will work.
- Graph Database
- Web Router (technically optional)

21.1 Gleaner Data Services (DS)

If you wish to work with a triplestore and wish to use the default app used by OIH you can use the compose file that sets up the Gleaner Data Services environment.

This adds the Blazegraph triplestore to the configuration along with the object store.

The details of the OIH data services are found in the *Data Services* section.

Typically, a user would wish to run the full Gleaner DS stack which supports both the indexing process and the serving of the resulting data warehouse and graph database capacity.

Combined, these would then look like the following where the indexing and data services shared a common object store.

21.1.1 Object store pattern

Within in the object store the following digital object pattern is used.This is based on the work of the RDA Digital Fabric working group.

At this point the graph and data warehouse (object store) can be exposed to the net for use by clients such as jupyter notebooks or direct client calls to the S3 object APIs and SPARQL endpoint.

Gleaner Data Services (DS) Environment Variables The Docker Compose file used to launch the Gleaner DS has a set of configurable elements that can be set and passed to the orchestration system with environment variables.

These can be set manually or through the command line. A simple script to set the variables could look like:

– Environment Var settings script

The actual services can be deployed via a Docker Compose file (also works with Podman). An example of that file and details about it follow.

Let's take a look at this.

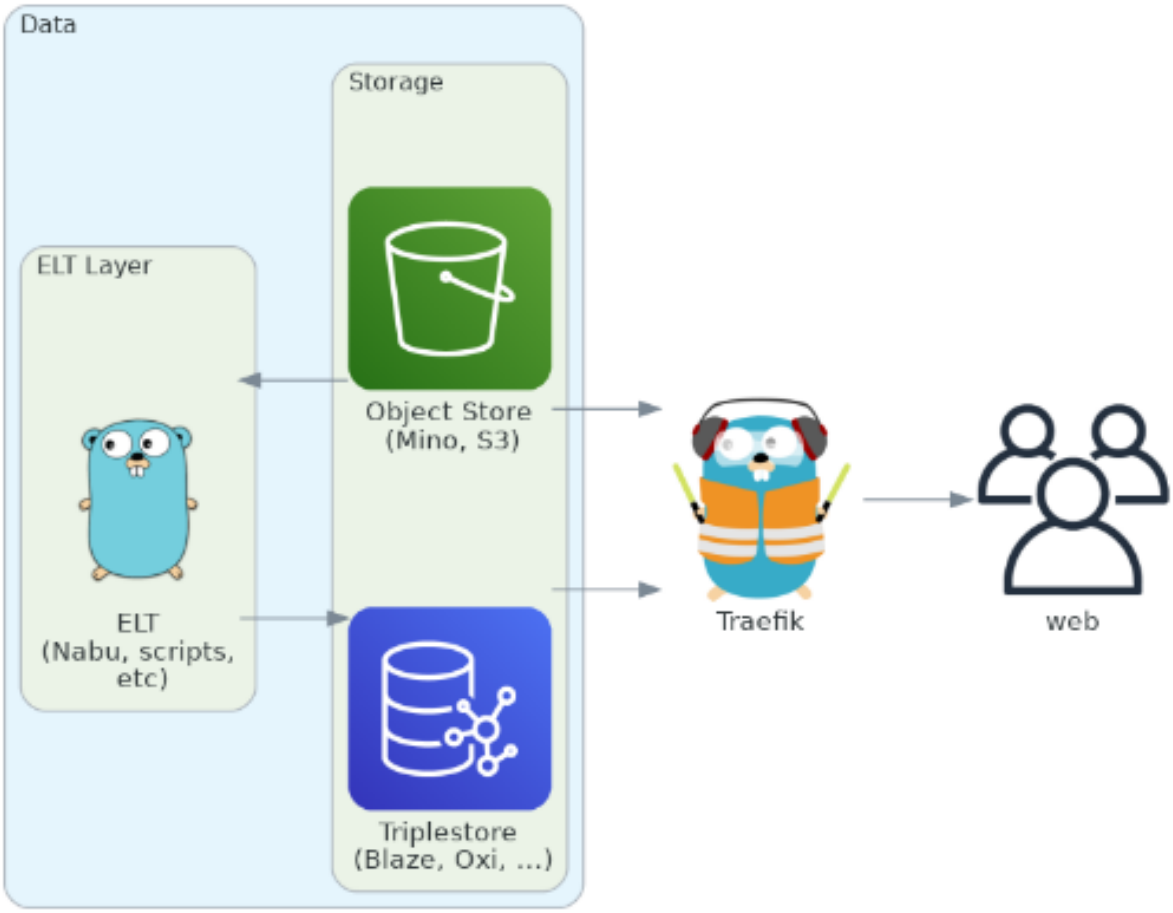


Fig. 1: Gleaner Data Service Activity Workflow

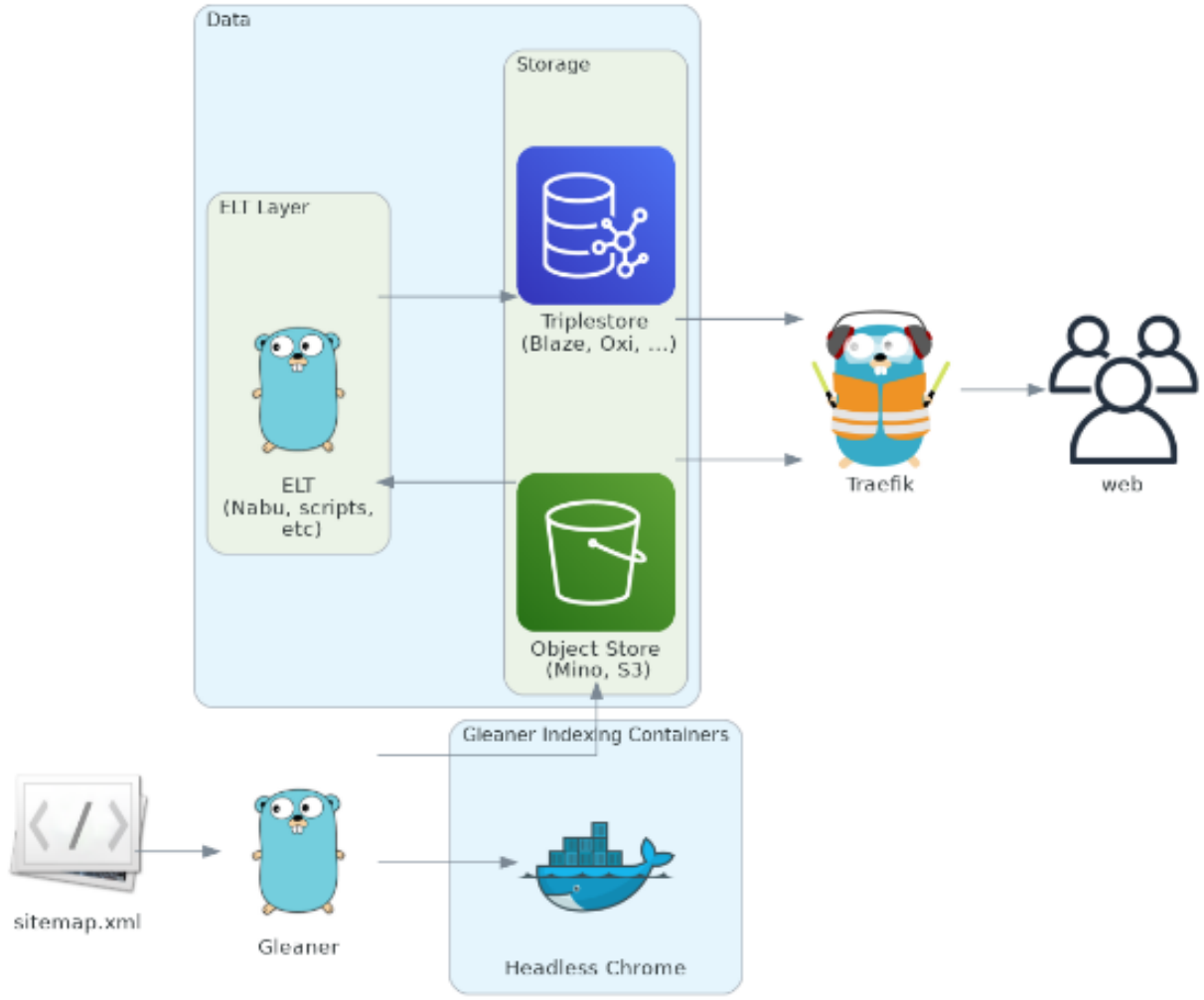


Fig. 2: Gleaner Indexing and Data Service Combined

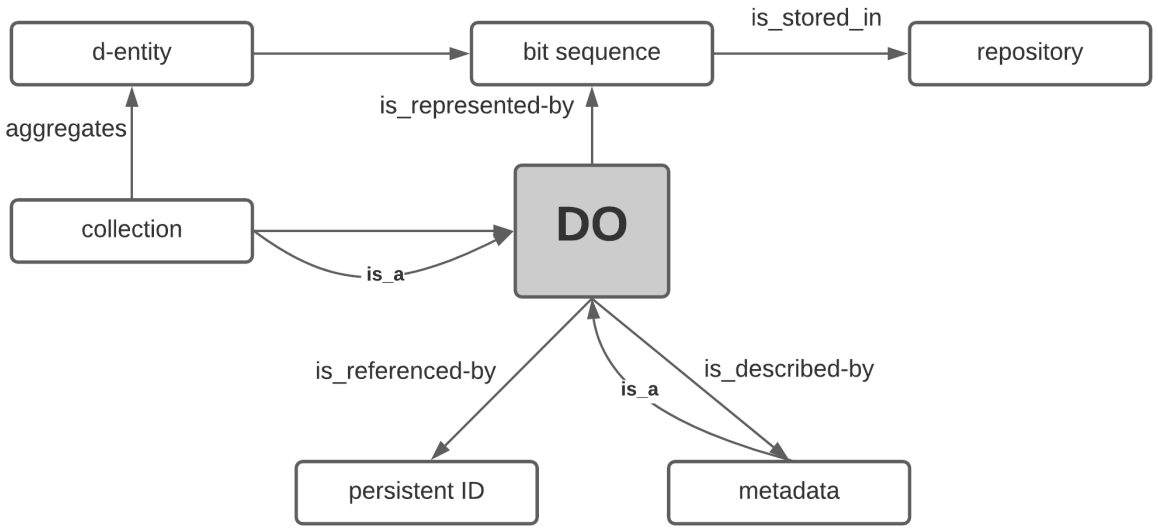


Figure 1: Research Data Alliance Digital Object Cloud Pattern

Fig. 3: Gleaner Digital Object Pattern

```
#!/bin/bash

# domains
export GLEANER_ADMIN_DOMAIN=admin.local.dev
export GLEANER_OSS_DOMAIN=oss.local.dev
export GLEANER_GRAPH_DOMAIN=graph.local.dev
export GLEANER_WEB_DOMAIN=web.local.dev
export GLEANER_WEB2_DOMAIN=web2.local.dev

# Object store keys
export MINIO_ACCESS_KEY=worldsbestaccesskey
export MINIO_SECRET_KEY=worldsbestsecretkey

# local data volumes
export GLEANER_BASE=/tmp/gleaner/
export GLEANER_TRAEFIK=${GLEANER_BASE}/config
export GLEANER_OBJECTS=${GLEANER_BASE}/datavol/s3
export GLEANER_GRAPH=${GLEANER_BASE}/datavol/graph

```

– Break down the compose file here

```
version: '3'

# ${GLEANER_ADMIN_DOMAIN}
# ${GLEANER_OSS_DOMAIN}
# ${GLEANER_GRAPH_DOMAIN}
# ${GLEANER_WEB_DOMAIN}
# ${GLEANER_WEB2_DOMAIN}
```

(continues on next page)

(continued from previous page)

```
# ${MINIO_ACCESS_KEY}
# ${MINIO_SECRET_KEY}
#
# ${GLEANER_TRAEFIK}
# ${GLEANER_OBJECTS}
# ${GLEANER_GRAPH}

services:
  triplestore:
    image: nawer/blazegraph
    environment:
      JAVA_XMS: 2g
      JAVA_XMX: 8g
      JAVA_OPTS: -Xmx6g -Xms2g --XX:+UseG1GC
    ports:
      - 9999:9999
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.triplestore.entrypoints=http"
      - "traefik.http.routers.triplestore.rule=Host(`${GLEANER_GRAPH_DOMAIN}`)"
      - "traefik.http.middlewares.triplestore-https-redirect.redirectscheme.
↪scheme=https"
      - "traefik.http.routers.triplestore.middlewares=triplestore-https-redirect"
      - "traefik.http.routers.triplestore-secure.entrypoints=https"
      - "traefik.http.routers.triplestore-secure.rule=Host(`${GLEANER_GRAPH_DOMAIN}`)"
      - "traefik.http.routers.triplestore-secure.tls=true"
      - "traefik.http.routers.triplestore-secure.tls.certresolver=http"
      - "traefik.http.routers.triplestore-secure.service=triplestore"
      - "traefik.http.middlewares.triplestore-secure.headers.
↪accesscontrolallowmethods=GET,OPTIONS,PUT,POST"
      - "traefik.http.middlewares.triplestore-secure.headers.
↪accesscontrolalloworigin=*"
      - "traefik.http.middlewares.triplestore-secure.headers.accesscontrolmaxage=200"
      - "traefik.http.middlewares.triplestore-secure.headers.addvaryheader=true"
      - "traefik.http.middlewares.triplestore-secure.headers.
↪accesscontrolallowcredentials=true"
      - "traefik.http.middlewares.triplestore-secure.headers.
↪accesscontrolallowheaders=Authorization,Origin,Content-Type,Accept"
      - "traefik.http.middlewares.triplestore-secure.headers.customresponseheaders.
↪Access-Control-Allow-Headers=Authorization,Origin,Content-Type,Accept"
      - "traefik.http.routers.triplestore-secure.middlewares=triplestore-secure@docker
↪"
      - "traefik.http.services.triplestore.loadbalancer.server.port=9999"
      - "traefik.docker.network=traefik_default"
    volumes:
      - ${GLEANER_GRAPH}:/var/lib/blazegraph
    networks:
      - traefik_default

  s3system:
    image: minio/minio:latest
    ports:
      - 9000:9000
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.s3system.entrypoints=http"
      - "traefik.http.routers.s3system.rule=Host(`${GLEANER_OSS_DOMAIN}`)"
```

(continues on next page)

(continued from previous page)

```
      - "traefik.http.middlewares.s3system-https-redirect.redirectscheme.scheme=https"
      - "traefik.http.routers.s3system.middlewares=s3system-https-redirect"
      - "traefik.http.routers.s3system-secure.entrypoints=https"
      - "traefik.http.routers.s3system-secure.rule=Host(`${GLEANER_OSS_DOMAIN}`)"
      - "traefik.http.routers.s3system-secure.tls=true"
      - "traefik.http.routers.s3system-secure.tls.certresolver=http"
      - "traefik.http.routers.s3system-secure.service=s3system"
      - "traefik.http.services.s3system.loadbalancer.server.port=9000"
      - "traefik.docker.network=traefik_default"
    volumes:
      - ${GLEANER_OBJECTS}:/data
    environment:
      - MINIO_ACCESS_KEY=${MINIO_ACCESS_KEY}
      - MINIO_SECRET_KEY=${MINIO_SECRET_KEY}
    networks:
      - traefik_default
    command: ["server", "/data"]

  features:
    image: fils/grow-general:latest
    ports:
      - 8080:8080
    environment:
      - S3ADDRESS=s3system:9000
      - S3BUCKET=sites
      - S3PREFIX=domain
      - DOMAIN=https://${GLEANER_WEB_DOMAIN}/
      - S3KEY=${MINIO_ACCESS_KEY}
      - S3SECRET=${MINIO_SECRET_KEY}
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.features.entrypoints=http"
      - "traefik.http.routers.features.rule=Host(`${GLEANER_WEB_DOMAIN}`, `${GLEANER_
↪WEB2_DOMAIN}`)"
      - "traefik.http.middlewares.features-https-redirect.redirectscheme.scheme=https"
      - "traefik.http.routers.features.middlewares=features-https-redirect"
      - "traefik.http.routers.features-secure.entrypoints=https"
      - "traefik.http.routers.features-secure.rule=Host(`${GLEANER_WEB_DOMAIN}`,`$
↪{GLEANER_WEB2_DOMAIN}`)"
      - "traefik.http.routers.features-secure.tls=true"
      - "traefik.http.routers.features-secure.tls.certresolver=http"
      - "traefik.http.routers.features-secure.service=features"
      - "traefik.http.services.features.loadbalancer.server.port=8080"
      - "traefik.docker.network=traefik_default"
      - "traefik.http.middlewares.features.headers.accesscontrolallowmethods=GET,
↪OPTIONS,PUT,POST"
      - "traefik.http.middlewares.features.headers.accesscontrolalloworigin=*"
      - "traefik.http.middlewares.features.headers.accesscontrolmaxage=100"
      - "traefik.http.middlewares.features.headers.addvaryheader=true"
      - "traefik.http.middlewares.features-secure.headers.accesscontrolallowheaders=*"
      - "traefik.http.middlewares.features-secure.headers.customresponseheaders.
↪Access-Control-Allow-Headers=*"
    networks:
      - traefik_default

networks:
  traefik_default:
```

(continues on next page)

(continued from previous page)

```
111
```

NOTE: DS also needs the object -> graph sync (via Nabu) NOTE: Should also add in (here or to the side) the ELT local Data Lake to Data Warehouse path (ala CSDCO VaultWalker)

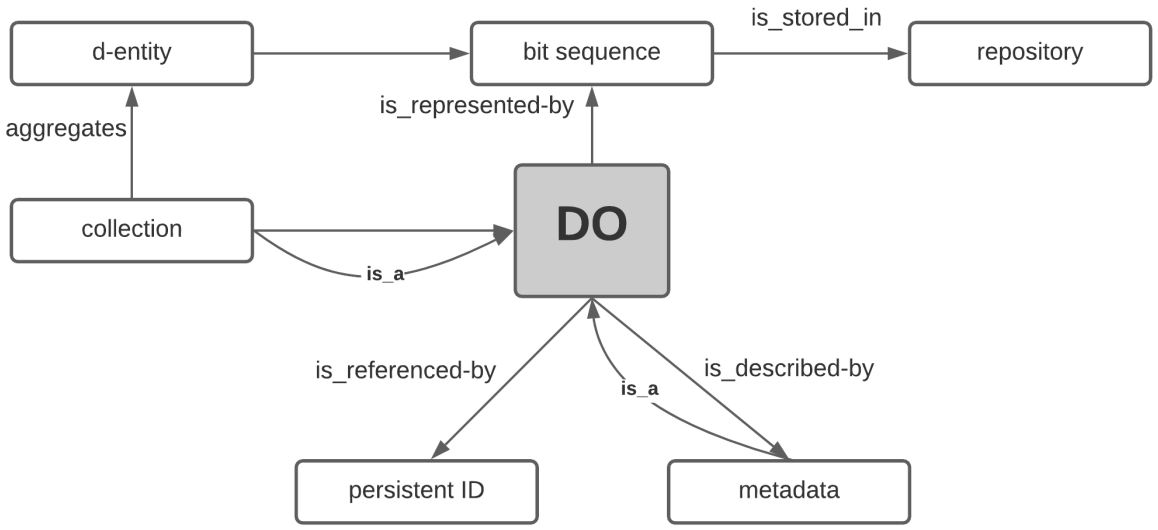


Figure 1: Research Data Alliance Digital Object Cloud Pattern

CHAPTER

TWENTYTWO

INTERFACES

22.1 About

In the end the goal is to provide use of the generated index. There are several possible used for an index.

- Web UI such as the reference client at oceans.collaborium.io
  - A variation on this is the development of web components that can be easily included in domain sites to perform operations on the OIH index
- graph access via SPARQL
- access to the graph and objects via workflows like Jupyter notebooks

22.2 Gleaner Web UI (WUI)

The user of the index may take several forms. A user may be a software developer creating a web based interface to the generated index. It may also be an end user accessing the index (indexes) through notebooks or special clients.

Those wishing to run a web site can augemnt the compose files to run their preferred web server, object server (to serve files from the object store) or software such as node or others to support their deployment pattern.

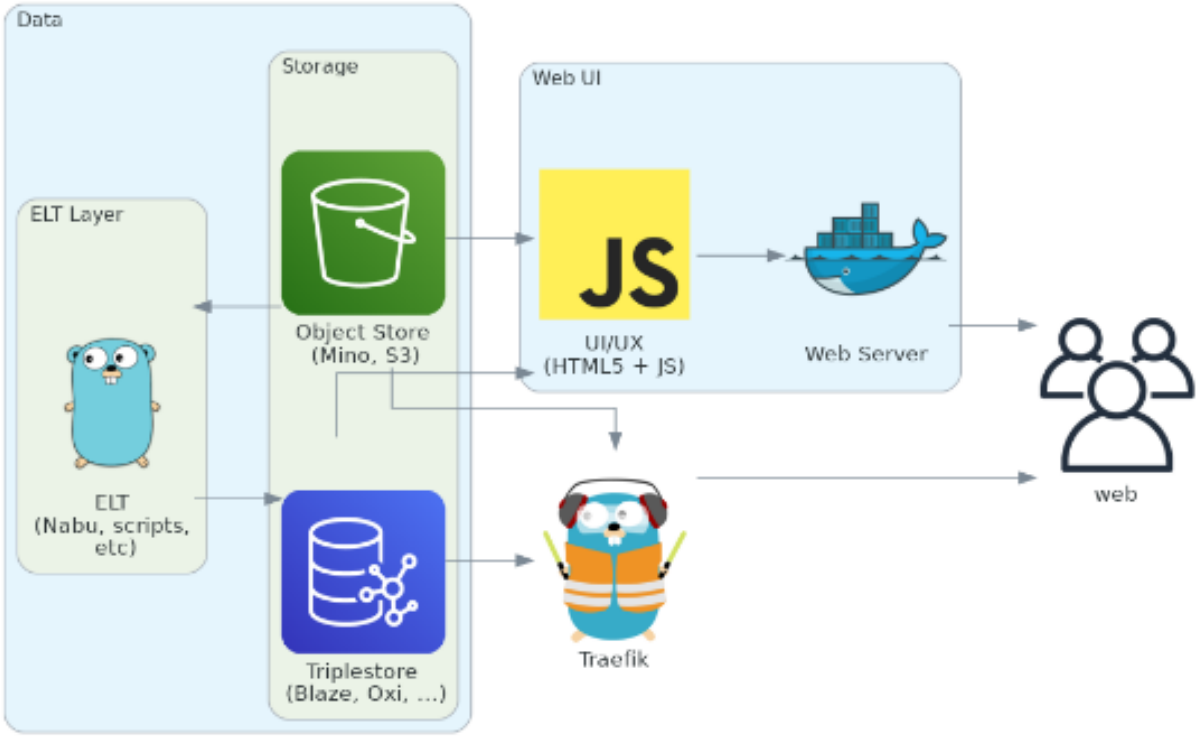


Fig. 1: Gleaner Optional Web UI

CHAPTER

TWENTYTHREE

GRAPH FIRST APPROACH

23.1 About

During the early adopters meetings and in discussion with others an alternative publication pattern came up. This is the pattern where it is not possible to update the web resources with the metadata content. This may be due to access or technical issues. Regardless, what was possible was to generate the metadata in bulk locally and make the resulting document available.

This approach is not ideal since it is a non-standard pattern and makes the data and information more obscure to other users. However, it is one the OIH architecture can adapt to and is preferable to the option of excluding those partners in this activity.

As such, we are making some changes to allow for this pattern. This means documenting the published graph structure based on the existing thematic patterns and some updates in the indexing workflow to obtain and integrate these graphs into the OIH graph.

> **Warning:** Anti-pattern: Using the approach here is not in alignment with Google guidance nor with W3C patterns for structured data on the web.
>
> It is documented here for edge cases where this is the minimum viable approach. The hope is it could act as a gateway to a more standards aligned implementation later.

23.2 Graph Only

There are cases where it is only possible to generate the graph based on the metadata. Access to the HTML pages is either difficult or the process of inserting the data into the pages is not supportable.

For this case the goal is to create a simple graph in JSON-LD. To do this we need a collection approach that is valid for a range of Things.

For this it is proposed to use ItemList which can be used on a list of type Thing, ie anything type in the Schema.org vocabulary.

This would define a ListItem with item of any type. Below is an example for a CreativeWork (map) and a Course. Once you are in a "item" any of the details from the other thematic type descriptions can be used.

```
{
  "@context": "https://schema.org/",
  "@type": ["ItemList", "CreativeWork"],
  "name": "Resource collection for site X",
  "author": "Creator of the list",
```

(continues on next page)

(continued from previous page)

```
  "itemListOrder": "https://schema.org/ItemListUnordered",
  "numberOfItems": 2,
  "itemListElement": [
    {
      "@type": "ListItem",
      "item": {
           "@id": "ID_for_this_metadata_record1",
           "@type": "Map",
            "@id": "https://example.org/id/XYZ",
            "name": "Name or title of the document",
            "description": "Description of the map to aid in searching",
            "url":  "https://www.sample-data-repository.org/creativework/map.pdf"
      }
    },
    {
      "@type": "ListItem",
      "item": {
           "@id": "ID_for_this_metadata_record2",
            "@type": "Course",
            "courseCode": "F300",
            "name": "Physics",
            "provider": {
                "@type": "CollegeOrUniversity",
                "name": "University of Bristol",
                "url": {
                    "@id": "/provider/324/university-of-bristol"
                }
            }
        }
    }
  ]
}
```

In the case of schema:Dataset one might use schema:DataCatlogue for the following approach. However, since OIH is addressing a wide range of types a more generic collection of Things or CreativeWorks approach is needed.

23.3 Item Catalogue Page

It's not hard to generate a simple HTMl page based on the structured metadata file. This doesn't alter the content of the graph, just builds an automated HTMl page around it.

23.4 Publishing and referencing

23.5 Testing

Since we are now dealing with a graph that is pulled as a complete entity there are a few thoughts.

1. How do ensure a connection between a record in the list and a resolvable URL? Do we need to:
    1. ensure each record has a IRI it is subject of
    2. in the case where IRI is or can be URL, do a validation of at least a 200 on it

2. How do we publish this?
    1. entry in robots.txt (might be able due to reasons above?)
    2. published and provided to OIH
3. Need guidance on format and structure

CHAPTER

TWENTYFOUR

PROV

24.1 About

This is the start of some discussion on issues around prov tracking in OIH. This may take two paths. One would be the prov tracking indexers might do and the other prov that providers would encode to provide specific prov the community requests.

24.2 Gleaner Prov

The Gleaner application generates a prov graph of the activity of accessing and indexing provider resources. The main goal of this prov is to connect an indexed URL to the digital object stored in the object store. This digital object should be the JSON-LD data graph presented by the provider.

By contrast, the authoritative reference in the various profiles will connect the the data graph ID, or in the absence of that the data graph URL or the referenced resources URL by gleaner, to another reference. This may be an organization ID or a PID of the connected resource.

```
{
    "@context": {
        "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
        "prov": "http://www.w3.org/ns/prov#",
        "rdfs": "http://www.w3.org/2000/01/rdf-schema#"
    },
    "@graph": [
        {
            "@id": "https://www.re3data.org/repository/obis",
            "@type": "prov:Organization",
            "rdf:name": "Ocean Biodiversity Information System",
            "rdfs:seeAlso": "https://obis.org"
        },
        {
            "@id": "https://obis.org/dataset/9381239f-3d64-48b4-80c9-b9ebb674edc2",
            "@type": "prov:Entity",
            "prov:wasAttributedTo": {
                "@id": "https://www.re3data.org/repository/obis"
            },
            "prov:value": "https://obis.org/dataset/9381239f-3d64-48b4-80c9-
↪b9ebb674edc2"
        },
        {
            "@id": "https://gleaner.io/id/collection/
↪7c1eaa1aaed95861330109026c42e57a31ecae55",
```

(continues on next page)

(continued from previous page)

```
            "@type": "prov:Collection",
            "prov:hadMember": {
                "@id": "https://obis.org/dataset/9381239f-3d64-48b4-80c9-b9ebb674edc2"
            }
        },
        {
            "@id": "urn:gleaner:milled:obis:7c1eaa1aaed95861330109026c42e57a31ecae55",
            "@type": "prov:Entity",
            "prov:value": "7c1eaa1aaed95861330109026c42e57a31ecae55.jsonld"
        },
        {
            "@id": "https://gleaner.io/id/run/7c1eaa1aaed95861330109026c42e57a31ecae55
↪",
            "@type": "prov:Activity",
            "prov:endedAtTime": {
                "@value": "2021-04-20",
                "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
            },
            "prov:generated": {
                "@id":
↪"urn:gleaner:milled:obis:7c1eaa1aaed95861330109026c42e57a31ecae55"
            },
            "prov:used": {
                "@id": "https://gleaner.io/id/collection/
↪7c1eaa1aaed95861330109026c42e57a31ecae55"
            }
        }
    ]
}
```

```
{
    "@context": {
        "@vocab": "https://schema.org/",
        "prov": "http://www.w3.org/ns/prov#"
    },
    "@id": "https://gleaner.io/id/run/7c1eaa1aaed95861330109026c42e57a31ecae55",
    "@type": "prov:Activity",
    "prov:endedAtTime": {
        "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
        "@value": "2021-04-20"
    },
    "prov:generated": {
        "@id": "urn:gleaner:milled:obis:7c1eaa1aaed95861330109026c42e57a31ecae55",
        "@type": "prov:Entity",
        "prov:value": "7c1eaa1aaed95861330109026c42e57a31ecae55.jsonld"
    },
    "prov:used": {
        "@id": "https://gleaner.io/id/collection/
↪7c1eaa1aaed95861330109026c42e57a31ecae55",
        "@type": "prov:Collection",
        "prov:hadMember": {
            "@id": "https://obis.org/dataset/9381239f-3d64-48b4-80c9-b9ebb674edc2",
            "@type": "prov:Entity",
            "prov:value": "https://obis.org/dataset/9381239f-3d64-48b4-80c9-
↪b9ebb674edc2",
            "prov:wasAttributedTo": {
```

(continues on next page)

(continued from previous page)

```
                "@id": "https://www.re3data.org/repository/obis",
                "@type": "prov:Organization",
                "http://www.w3.org/1999/02/22-rdf-syntax-ns#name": "Ocean␣
↪Biodiversity Information System",
                "http://www.w3.org/2000/01/rdf-schema#seeAlso": "https://obis.org"
            }
        }
    }
}
```

24.3 Nano Prov

This is a basic nanoprov example. Note, this is a draft and the ID connections and examples have not been made yet.

```
{
    "@context": {
        "gleaner": "https://voc.gleaner.io/id/",
        "np": "http://www.nanopub.org/nschema#",
        "prov": "http://www.w3.org/ns/prov#",
        "xsd": "http://www.w3.org/2001/XMLSchema#"
    },
    "@set": [
        {
            "@id": "gleaner:nanopub/XID",
            "@type": "np:NanoPublication",
            "np:hasAssertion": {
                "@id": "gleaner:nanopub/XID#assertion"
            },
            "np:hasProvenance": {
                "@id": "gleaner:nanopub/XID#provenance"
            },
            "np:hasPublicationInfo": {
                "@id": "gleaner:nanopub/XID#pubInfo"
            }
        },
        {
            "@id": "gleaner:nanopub/XID#assertion",
            "@graph": {
                "@id": "DataSetURI",
                "@type": "schema:Dataset",
                "description": "This is where you would put corrections or annotations
↪",
                "identifier": [
                    {
                        "@type": "schema:PropertyValue",
                        "name": "GraphSHA",
                        "description": "A SHA256 sha stamp on the harvested data␣
↪graph from a URL",
                        "value": "{{SHA256 HASH HERE}}"
                    },
                    {
                        "@type": "schema:PropertyValue",
                        "name": "ProviderID",
                        "description": "The id provided with the data graph by the␣
↪provider",
```

(continues on next page)

(continued from previous page)

```
                        "value": "{{re3 or URL noted in config}}"
                    },
                    {
                        "@type": "schema:PropertyValue",
                        "name": "URL",
                        "description": "The URL harvested by gleaner",
                        "value": "{{The URL the JSON-LD came from}}"
                    }
                ]
            }
        },
        {
            "@id": "gleaner:nanopub/XID#provenance",
            "@graph": {
                "@id": "URIforprovondataset",
                "prov:wasGeneratedAtTime": {
                    "@value": "dateDone",
                    "@type": "xsd:dateTime"
                },
                "prov:wasDerivedFrom": {
                    "@id": "IDHERE"
                },
                "prov:wasAttributedTo": {
                    "@id": "IDHERE"
                }
            }
        },
        {
            "@id": "gleaner:nanopub/XID#pubInfo",
            "@graph": {
                "@id": "IDHERE",
                "prov:wasAttributedTo": {
                    "@id": "gleaner:tool/gleaner"
                },
                "prov:generatedAtTime": {
                    "@value": "2019-10-23T14:38:00Z",
                    "@type": "xsd:dateTime"
                }
            }
        }
    ]
}
```

```
<graphviz.dot.Digraph at 0x7f3439ea0070>
```

24.4 Refs

Nanopubs Guidance

CHAPTER

TWENTYFIVE

ALTERNATIVES

25.1 Options

While Gleaner will be used during initial OIH development it is not the only or required approach. The web architecture foundation means there are many other tools that can be used and might be leveraged in a production environment including:

- Extrunct
- BioSchemas Tools
- LDSpider
- Squirrel
- Nutch (Apache)
- Laundromat
- DataArchiver
- OD Archiver

These different tools may better fit into the workflow and available skill sets for a group. Distinct from these is DataONE Plus which is a "Search as a Service" offering from DataONE.

Part IV

Tooling

CHAPTER

TWENTYSIX

TOOLING

26.1 About

The tooling section is a collection of tools, scripts, notebooks and other software that could be of use to the various personas of Ocean InfoHub

26.1.1 OpenRefine

In this section you will find some details around the Open Refine project and how to use it to generate JSON-LD documents.

26.1.2 Notebooks

In this section you will find some Jupyter Notebooks that demonstrate working with JSON-LD in various ways. These can be copied and used locally to explore or implement workflows.

26.2 On-line tooling

Schema.org Validator

json-ld playgroud

SHACL Playground

json-lint

f-uji

26.3 Dev

json-ld

f-uji dev

jq

jello (python) and Practical JSON at the command line using jello

Python Extruct

26.4 OpenRefine

26.4.1 About

Some examples of using OpenRefine to generate a list of things. This is done as an example for those who may have a more manual workflow and wish to explore some tools to help automating that.

As the output from the template export in OpenRefine is an array it give us a chance to look at both DataFeed and ItemList.

In the wonderfully open world of Schema.org thee is also Series and its subtypes of CreativeWorkSeries and EventSeries.

Since the majority of what we work with are not data sets, we will focus on the ItemList and CreativeWorkSeries. The comparison to RSS is valid and casting to RSS is likely easy should it be desired.

Note, in the context of OIH this also raises the option of leveraging these approaches for the publishing and indexing of resources that align with this model.

26.4.2 Generic Template

About

The following are the templates sections for the OpenRefine export template command.

There are four sections

PREFIX

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": ["ItemList", "CreativeWork"],
    "name": "Creative work list",
    "author": "Author of the list",
    "about": {
      "@type": "Course"
    },
    "itemListElement": [
```

ROW TEMPLATE

```
{
    "@context": {
        "@vocab": "https://schema.org/",
        "endDate": {
            "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
        },
        "startDate": {
            "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
        }
    },
    "@id":  {{jsonize(cells["ID"].value)}},
```

(continues on next page)

(continued from previous page)

```
    "@type":{{jsonize(cells["type"].value)}},
    "description":  {{jsonize(cells["description"].value)}},
    "name" : {{jsonize(cells["name"].value)}},
    "hasCourseInstance": {
        "@type":  {{jsonize(cells["CourseInstance"].value)}},
        "courseMode":  {{jsonize(cells["courseMode"].value)}},
        "endDate":  {{jsonize(cells["enddata"].value)}},
        "startDate":{{jsonize(cells["startdate"].value)}}
    },
    "provider": {
        "@type": "CollegeOrUniversity",
        "name": {{jsonize(cells["provider.name"].value)}},
        "url": {
            "@id":  {{jsonize(cells["provider.url"].value)}}
        }
    }
}
```

ROW SEP

```
,
```

SUFFIX

```
 ]
}
```

NOTES

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": "Course",
    "courseCode": "F300",
    "name": "Physics",
    "provider": {
        "@type": "CollegeOrUniversity",
        "name": "University of Bristol",
        "url": {
            "@id": "/provider/324/university-of-bristol"
        }
    }
}


    {
      "ID" : {{jsonize(cells["ID"].value)}},
```

(continues on next page)

(continued from previous page)

```
      "type" : {{jsonize(cells["type"].value)}},
      "description" : {{jsonize(cells["description"].value)}},
      "name" : {{jsonize(cells["name"].value)}},
      "provider.name" : {{jsonize(cells["provider.name"].value)}},
      "provider.url" : {{jsonize(cells["provider.url"].value)}},
      "CourseInstance" : {{jsonize(cells["CourseInstance"].value)}},
      "courseMode" : {{jsonize(cells["courseMode"].value)}},
      "enddata" : {{jsonize(cells["enddata"].value)}},
      "startdate" : {{jsonize(cells["startdate"].value)}}
    }
```

26.4.3 Sargassum Project Template

About

The following is a simple proof of concept page. It uses data from the Sargassum Projects page and is presented here simply as an example of this workflow.

If we visit the page referenced above, we can see some map interfaces. In the first, we are able to view and export the data in the map as a table. This will be downloaded as a CSV file. We can load this .csv file directly into OpenRefine and from there use the templating exporterto generate a valid JSON-LD document.

The following are the templates sections for the OpenRefine export template command.

There are four sections

PREFIX

```
{
     "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": ["ItemList", "ResearchProject"],
    "name": "Sargassum Information Hub Projects",
    "author": "Sargassum Information Hub",
    "about": {
      "@type": "ResearchProject"
    },
    "itemListElement": [
```

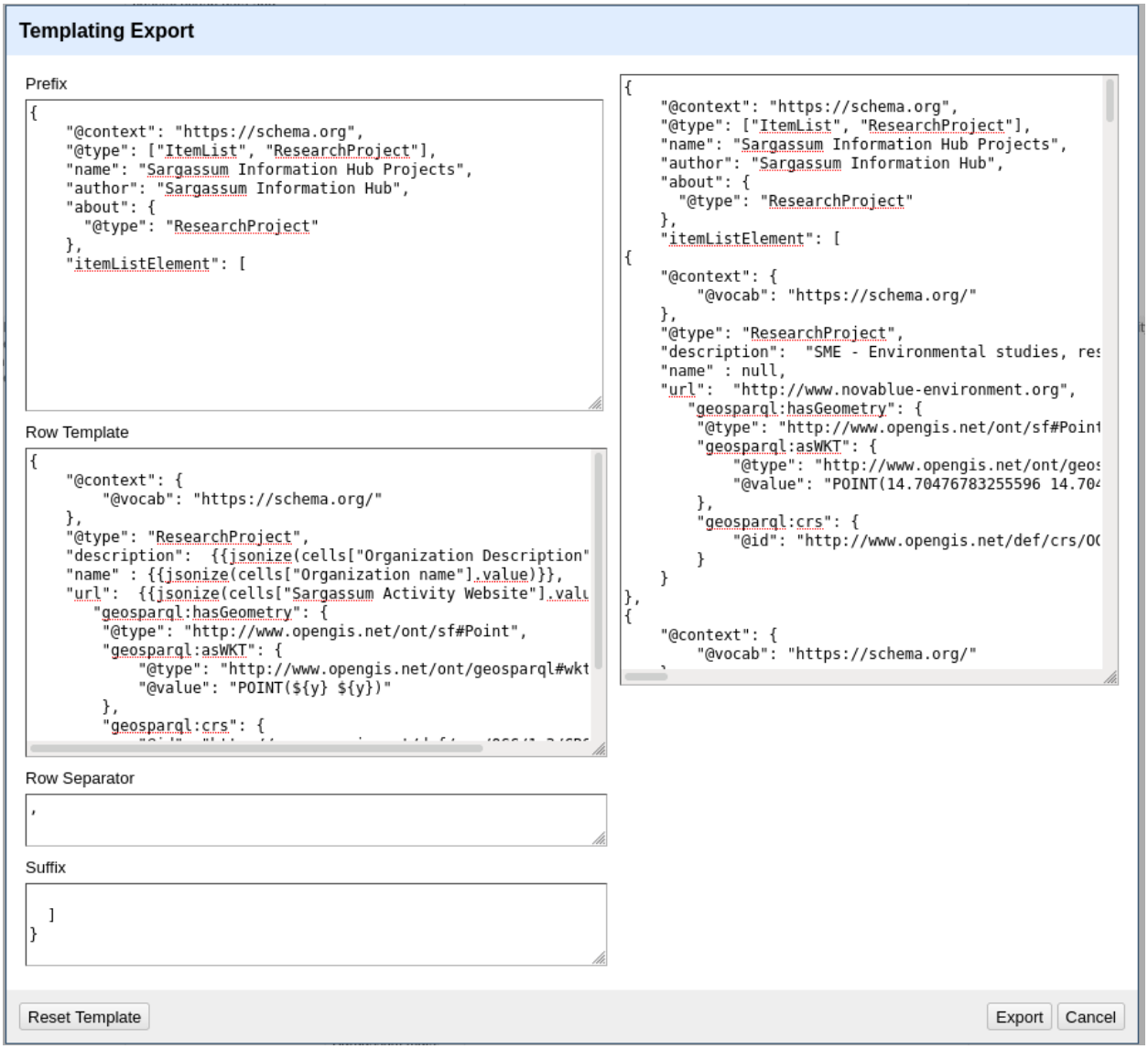


Fig. 1: A view of the template export in OpenRefine with the following sections inserted

ROW TEMPLATE

Note

In the following row template the entries such as:

```
  {{jsonize(cells["Organization Description"].value)}}
```

will present a value with quotes around it.

Where

```
${x}
```

will present the value from the noted colum, x, with no quotes around it.

When generating the JSON we may or may not need to include quotes depending on where we are in the serialization.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": "ResearchProject",
    "description":  {{jsonize(cells["Organization Description"].value)}},
    "name" : {{jsonize(cells["Organization name"].value)}},
    "url":  {{jsonize(cells["Sargassum Activity Website"].value)}},
       "geosparql:hasGeometry": {
        "@type": "http://www.opengis.net/ont/sf#Point",
        "geosparql:asWKT": {
            "@type": "http://www.opengis.net/ont/geosparql#wktLiteral",
            "@value": "POINT(${y} ${y})"
        },
        "geosparql:crs": {
            "@id": "http://www.opengis.net/def/crs/OGC/1.3/CRS84"
        }
    }
}
```

ROW SEP

```
,
```

SUFFIX

```
 ]
}
```

NOTES

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": "Course",
    "courseCode": "F300",
    "name": "Physics",
    "provider": {
        "@type": "CollegeOrUniversity",
        "name": "University of Bristol",
        "url": {
            "@id": "/provider/324/university-of-bristol"
        }
    }
}
```

```
    {
      "ID" : {{jsonize(cells["ID"].value)}},
      "type" : {{jsonize(cells["type"].value)}},
      "description" : {{jsonize(cells["description"].value)}},
      "name" : {{jsonize(cells["name"].value)}},
      "provider.name" : {{jsonize(cells["provider.name"].value)}},
      "provider.url" : {{jsonize(cells["provider.url"].value)}},
      "CourseInstance" : {{jsonize(cells["CourseInstance"].value)}},
      "courseMode" : {{jsonize(cells["courseMode"].value)}},
      "enddata" : {{jsonize(cells["enddata"].value)}},
      "startdate" : {{jsonize(cells["startdate"].value)}}
    }
```

26.5 OIH Notebooks

26.5.1 About

Notebooks related to the OIH project. These are early documents and they and this documentation will expand. These notebooks are developed and run using the Jupyter Data Science notebooks server (https://hub.docker.com/r/jupyter/datascience-notebook). The will likely run on Google Colab or Binder but additional installs may be required. We will attempt to document all requirements in the notebooks.

26.5.2 OIH Graph

Some analysis of the OIH graphs

- https://stackoverflow.com/questions/39274216/visualize-an-rdflib-graph-in-python
- https://networkx.org/documentation/stable/reference/algorithms/link_analysis.html

```
!pip install -q SPARQLWrapper
!pip -q install pydotplus
!pip -q install graphviz
!pip -q install pydotplus
!pip -q install mimesis
!pip -q install minio
!pip -q install s3fs
!pip -q install SPARQLWrapper
!pip -q install boto3
!pip -q install 'fsspec>=0.3.3'
!pip -q install rdflib  # !pip install -q -e git+https://github.com/RDFLib/rdflib.git
 ↪#egg=rdflib
!pip -q install rdflib-jsonld
!pip -q install PyLD==2.0.2
!pip -q install kglab
```

```
ERROR: pip's dependency resolver does not currently take into account all the␣
 ↪packages that are installed. This behaviour is the source of the following␣
 ↪dependency conflicts.
graph-notebook 2.1.4 requires networkx==2.4, but you have networkx 2.5.1 which is␣
 ↪incompatible.
aiobotocore 1.3.3 requires botocore<1.20.107,>=1.20.106, but you have botocore 1.21.
 ↪10 which is incompatible.
ERROR: pip's dependency resolver does not currently take into account all the␣
 ↪packages that are installed. This behaviour is the source of the following␣
 ↪dependency conflicts.
graph-notebook 2.1.4 requires networkx==2.4, but you have networkx 2.5.1 which is␣
 ↪incompatible.
boto3 1.18.4 requires botocore<1.22.0,>=1.21.4, but you have botocore 1.20.106 which␣
 ↪is incompatible.
```

```
from SPARQLWrapper import SPARQLWrapper, JSON
import pandas as pd
import dask, boto3
import dask.dataframe as dd
import numpy as np
import json
import geopandas
import matplotlib.pyplot as plt
import shapely
import kglab
```

```
#@title
def get_sparql_dataframe(service, query):
    """
    Helper function to convert SPARQL results into a Pandas data frame.
    """
    sparql = SPARQLWrapper(service)
    sparql.setQuery(query)
```

(continues on next page)

(continued from previous page)

```
    sparql.setReturnFormat(JSON)
    result = sparql.query()

    processed_results = json.load(result.response)
    cols = processed_results['head']['vars']

    out = []
    for row in processed_results['results']['bindings']:
        item = []
        for c in cols:
            item.append(row.get(c, {}).get('value'))
        out.append(item)

    return pd.DataFrame(out, columns=cols)
```

Some inspection queries for OIH Graph

```
oihgraph = "https://graph.collaborium.io/blazegraph/namespace/oihdev/sparql"
```

```
rp3 = """
prefix prov: <http://www.w3.org/ns/prov#>
PREFIX con: <http://www.ontotext.com/connectors/lucene#>
PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
PREFIX con-inst: <http://www.ontotext.com/connectors/lucene/instance#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <https://schema.org/>
PREFIX schemaold: <http://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT DISTINCT ?g  ?s  ?wat ?orgname ?domain ?type ?score ?name ?url ?lit ?
 ↪description ?headline
WHERE
{

   ?lit bds:search "coral" .
   ?lit bds:matchAllTerms "false" .
   ?lit bds:relevance ?score .
   ?s ?p ?lit .

   graph ?g {
    ?s ?p ?lit .
    ?s rdf:type ?type .
    OPTIONAL { ?s schema:name ?name .   }
    OPTIONAL { ?s schema:headline ?headline .   }
    OPTIONAL { ?s schema:url ?url .   }
    OPTIONAL { ?s schema:description ?description .    }
  }
   ?sp prov:generated ?g  .
   ?sp prov:used ?used .
   ?used prov:hadMember ?hm .
   ?hm prov:wasAttributedTo ?wat .
   ?wat rdf:name ?orgname .
   ?wat rdfs:seeAlso ?domain
```

(continues on next page)

(continued from previous page)

```
}
ORDER BY DESC(?score)
LIMIT 30
OFFSET 0
"""

dfrp3 = get_sparql_dataframe(oihgraph, rp3)
dfrp3.head(10)
```

```
                                                   g       s  \
0  urn:gleaner:milled:obis:13392d707024cdd4e509d6...  t12576
1  urn:gleaner:milled:obis:18d1180a74c200d06f9114...  t13860
2  urn:gleaner:milled:obis:24bac898cda34444176ec4...  t16445
3  urn:gleaner:milled:obis:24d453e3a4ea6d1f117e5c...  t16471
4  urn:gleaner:milled:obis:2524f94920efb8f87029bf...  t16581
5  urn:gleaner:milled:obis:2bf98aa888d856b8706176...  t18254
6  urn:gleaner:milled:obis:2d67f3625478df2c1520ae...  t18644
7  urn:gleaner:milled:obis:30c1e35e0d4a09e3aafd38...  t19399
8  urn:gleaner:milled:obis:384060928d22941acabcb6...  t21027
9  urn:gleaner:milled:obis:3c03fa0cea67f703cdf249...  t21600

                                      wat  \
0  https://www.re3data.org/repository/obis
1  https://www.re3data.org/repository/obis
2  https://www.re3data.org/repository/obis
3  https://www.re3data.org/repository/obis
4  https://www.re3data.org/repository/obis
5  https://www.re3data.org/repository/obis
6  https://www.re3data.org/repository/obis
7  https://www.re3data.org/repository/obis
8  https://www.re3data.org/repository/obis
9  https://www.re3data.org/repository/obis

                               orgname            domain  \
0  Ocean Biodiversity Information System  https://obis.org
1  Ocean Biodiversity Information System  https://obis.org
2  Ocean Biodiversity Information System  https://obis.org
3  Ocean Biodiversity Information System  https://obis.org
4  Ocean Biodiversity Information System  https://obis.org
5  Ocean Biodiversity Information System  https://obis.org
6  Ocean Biodiversity Information System  https://obis.org
7  Ocean Biodiversity Information System  https://obis.org
8  Ocean Biodiversity Information System  https://obis.org
9  Ocean Biodiversity Information System  https://obis.org

                         type score  \
0  https://schema.org/Dataset   1.0
1  https://schema.org/Dataset   1.0
2  https://schema.org/Dataset   1.0
3  https://schema.org/Dataset   1.0
4  https://schema.org/Dataset   1.0
5  https://schema.org/Dataset   1.0
6  https://schema.org/Dataset   1.0
7  https://schema.org/Dataset   1.0
8  https://schema.org/Dataset   1.0
```

(continues on next page)

(continued from previous page)

```
9  https://schema.org/Dataset   1.0

                                                  name  \
0  Coral Reef Evaluation and Monitoring Project F...
1  Coral Reef Evaluation and Monitoring Project D...
2  Coral Reef Evaluation and Monitoring Project F...
3  Coral Reef Evaluation and Monitoring Project F...
4  Interacciones entre Corales y CÃ©spedes algale...
5  Coral Reef Evaluation and Monitoring Project F...
6  Coral Reef Evaluation and Monitoring Project D...
7  Coral Reef Evaluation and Monitoring Project D...
8                  Nematoda from Kenya and Zanzibar
9  Coral Reef Evaluation and Monitoring Project F...

                                                   url    lit  \
0  https://obis.org/dataset/b91d89db-79d6-4bd3-84...  coral
1  https://obis.org/dataset/46005357-02b8-4f17-b0...  coral
2  https://obis.org/dataset/d4ec17b8-fc96-49b9-b7...  coral
3  https://obis.org/dataset/36bca81c-6d77-4fd4-a9...  coral
4  https://obis.org/dataset/e39be6ef-3c91-4e97-ba...  coral
5  https://obis.org/dataset/431f96f7-521c-4182-ae...  coral
6  https://obis.org/dataset/d88a91c1-2685-4afa-9a...  coral
7  https://obis.org/dataset/b856037f-bbdf-45da-9b...  coral
8  https://obis.org/dataset/aa9787d6-c4db-4fde-8e...  Coral
9  https://obis.org/dataset/c170a0a3-c669-436b-a1...  coral

                                         description headline
0  The purpose of the Coral Reef Evaluation and M...     None
1  The purpose of the Coral Reef Evaluation and M...     None
2  The purpose of the Coral Reef Evaluation and M...     None
3  The purpose of the Coral Reef Evaluation and M...     None
4  Para el componente denominado â??Interacciã³n ...     None
5  The purpose of the Coral Reef Evaluation and M...     None
6  The purpose of the Coral Reef Evaluation and M...     None
7  The purpose of the Coral Reef Evaluation and M...     None
8  Data on the species and trophic composition of...     None
9  The purpose of the Coral Reef Evaluation and M...     None
```

```
import rdflib
from rdflib.extras.external_graph_libs import rdflib_to_networkx_multidigraph
from rdflib.extras.external_graph_libs import rdflib_to_networkx_digraph
import networkx as nx
import matplotlib.pyplot as plt
import gzip

with gzip.open('./data/oceanexperts_graph.nq.gz', 'rb') as f:
    file_content = f.read()

g = rdflib.Graph()
g.parse(data = file_content, format="nquads")

G = rdflib_to_networkx_digraph(g)
# G = rdflib_to_networkx_multidigraph(result)

# # Plot Networkx instance of RDF Graph
# pos = nx.spring_layout(G, scale=2)
```

(continues on next page)

(continued from previous page)

```
# edge_labels = nx.get_edge_attributes(G, 'r')b
# #nx.draw_networkx_edge_labels(G, pos, labels=edge_labels)
# nx.draw_networkx_edge_labels(G, pos)
# nx.draw(G, with_labels=True)
```

```
pr = nx.pagerank(G,alpha=0.9)
# for key, value in pr.items():
#     print(key, ' : ', value)
```

```
import pandas as pd
prdf = pd.DataFrame.from_dict(pr, orient='index')
```

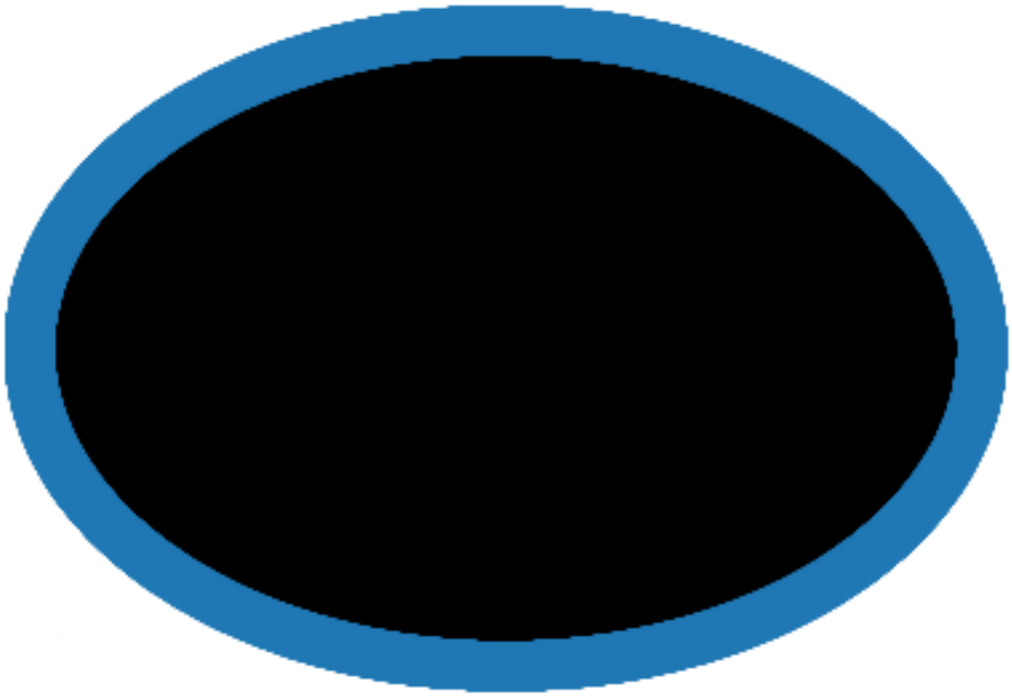
```
prdf.dtypes
```

```
0    float64
dtype: object
```

```
prdf.sort_values(by=0,ascending=False, inplace=True,)
prdf.head(20)
```

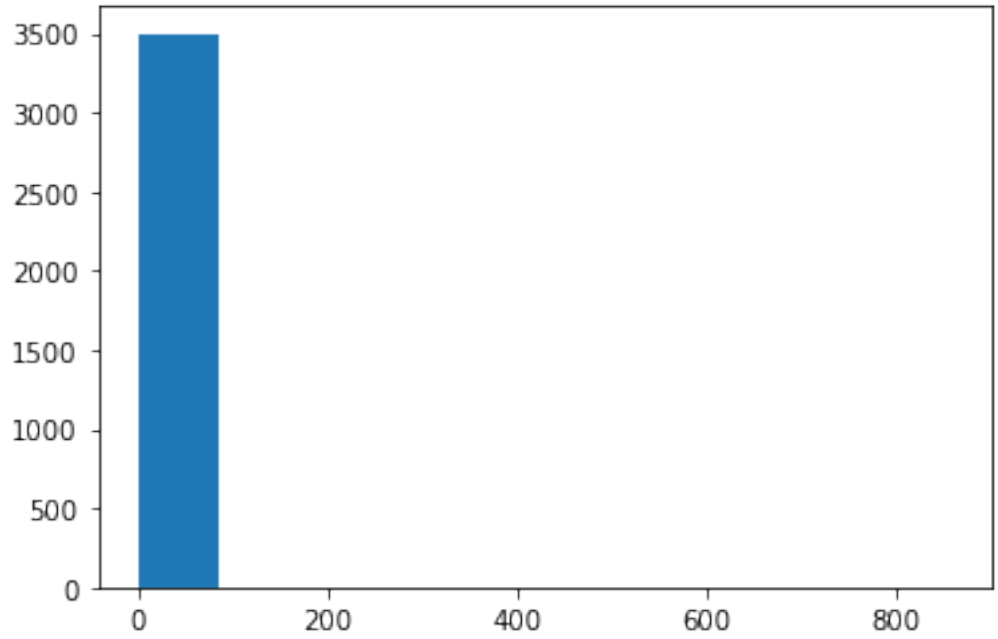
	0
https://schema.org/Place	0.058482
https://schema.org/CourseInstance	0.018446
https://schema.org/Course	0.016571
UNESCO/IOC Project Office for IODE Wandelaarka...	0.007863
Russia	0.007497
UNESCO/IOC Project Office for IODE Wandelaarka...	0.006203
Wandelaarkaai 7 8400 Oostende Belgium	0.003761
Belgium	0.002931
RV Professor Logachev Russia	0.002624
UNESCO / IOC Project Office for IODE Wandelaar...	0.002360
IOC Science and Communication Centre on Harmfu...	0.001830
Instituto de Investigaciones Marinas y Costera...	0.001812
"Ocean Valley", Pragathi Nagar (BO),...	0.001812
Kenya Marine and Fisheries Research Institute,...	0.001548
Calle 25 No. 2-55, Playa Salguero, Rodadero S...	0.001542
Institute of Oceanography and Environment Univ...	0.001271
Australia	0.001271
, Colombia	0.001271
Qingdao China	0.001018
Wandelaarkaai 7 Oostende Belgium	0.001018

```
nx.draw_circular(G, with_labels = False)
plt.show() # display
```



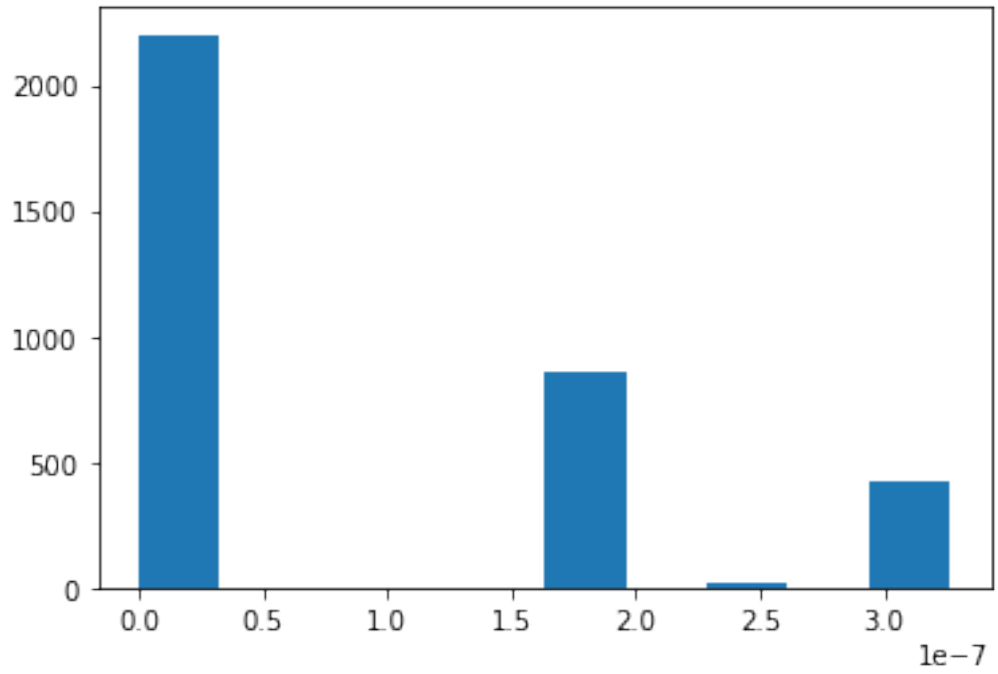
```
plt.hist([v for k,v in nx.degree(G)])
```

```
(array([3.499e+03, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 2.000e+00,
        0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00]),
 array([  1. ,  86.5, 172. , 257.5, 343. , 428.5, 514. , 599.5, 685. ,
        770.5, 856. ]),
 <BarContainer object of 10 artists>)
```




```
plt.hist(nx.centrality.betweenness_centrality(G).values())
```

```
(array([2200.,    0.,    0.,    0.,    0.,  858.,    0.,   20.,    0.,
         424.]),
 array([0.00000000e+00, 3.26437344e-08, 6.52874689e-08, 9.79312033e-08,
        1.30574938e-07, 1.63218672e-07, 1.95862407e-07, 2.28506141e-07,
        2.61149876e-07, 2.93793610e-07, 3.26437344e-07]),
 <BarContainer object of 10 artists>)
```



26.5.3 OIH Queries

What follows are some example SPARQL queries used in OIH for the test interface

Setup and inits

Installs

```
%%capture
#@title
!pip install -q SPARQLWrapper
!pip install -q cython
!pip install -q cartopy
!pip install -q geopandas
!pip install -q contextily==1.0rc2
!pip install pyshacl
!pip install 'PyLD>=2.0.3'
!pip install flatten_json
!pip install 'fsspec>=0.3.3'
!pip install s3fs
!pip install boto3
!pip install -q kglab
```

Imports

```
from SPARQLWrapper import SPARQLWrapper, JSON
import pandas as pd
import dask, boto3
import dask.dataframe as dd
import numpy as np
import json
import geopandas
import matplotlib.pyplot as plt
import shapely
import kglab

oih = "https://graph.collaborium.io/blazegraph/namespace/oihdev/sparql"
oihad = "https://graph.collaborium.io/blazegraph/namespace/aquadocs/sparql"
oihobps = "https://graph.collaborium.io/blazegraph/namespace/obps/sparql"
oihlocal = "http://192.168.86.45:49158/blazegraph/namespace/oih/sparql"
```

Functions

```
#@title
def get_sparql_dataframe(service, query):
    """
    Helper function to convert SPARQL results into a Pandas data frame.
    """
    sparql = SPARQLWrapper(service)
    sparql.setQuery(query)
    sparql.setReturnFormat(JSON)
```

(continues on next page)

(continued from previous page)

```
    result = sparql.query()

    processed_results = json.load(result.response)
    cols = processed_results['head']['vars']

    out = []
    for row in processed_results['results']['bindings']:
        item = []
        for c in cols:
            item.append(row.get(c, {}).get('value'))
        out.append(item)

    return pd.DataFrame(out, columns=cols)
```

```
rq_main = """prefix prov: <http://www.w3.org/ns/prov#>
        PREFIX con: <http://www.ontotext.com/connectors/lucene#>
        PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
        PREFIX con-inst: <http://www.ontotext.com/connectors/lucene/instance#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX schema: <https://schema.org/>
        PREFIX schemaold: <http://schema.org/>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

        SELECT DISTINCT ?g  ?s  ?wat ?orgname ?domain ?type ?score ?name ?url ?lit ?
↪description ?headline
        WHERE
        {
           ?lit bds:search "coral" .
           ?lit bds:matchAllTerms "false" .
           ?lit bds:relevance ?score .
           graph ?g {
            ?s ?p ?lit .
            ?s rdf:type ?type .
            OPTIONAL { ?s schema:name ?name .   }
            OPTIONAL { ?s schema:headline ?headline .   }
            OPTIONAL { ?s schema:url ?url .   }
            OPTIONAL { ?s schema:description ?description .    }
          }
           ?sp prov:generated ?g  .
           ?sp prov:used ?used .
           ?used prov:hadMember ?hm .
           ?hm prov:wasAttributedTo ?wat .
           ?wat rdf:name ?orgname .
           ?wat rdfs:seeAlso ?domain
        }
        ORDER BY DESC(?score)
        LIMIT 30
        OFFSET 0
        """
```

```
df = get_sparql_dataframe(oih, rq_main)
df.head(5)
```

```
                                                  g       s  \
0  urn:gleaner:milled:obis:13392d707024cdd4e509d6...  t12576
```

(continues on next page)

(continued from previous page)

```
1  urn:gleaner:milled:obis:18d1180a74c200d06f9114...  t13860
2  urn:gleaner:milled:obis:24bac898cda34444176ec4...  t16445
3  urn:gleaner:milled:obis:24d453e3a4ea6d1f117e5c...  t16471
4  urn:gleaner:milled:obis:2524f94920efb8f87029bf...  t16581

                                       wat  \
0  https://www.re3data.org/repository/obis
1  https://www.re3data.org/repository/obis
2  https://www.re3data.org/repository/obis
3  https://www.re3data.org/repository/obis
4  https://www.re3data.org/repository/obis

                                 orgname            domain  \
0  Ocean Biodiversity Information System  https://obis.org
1  Ocean Biodiversity Information System  https://obis.org
2  Ocean Biodiversity Information System  https://obis.org
3  Ocean Biodiversity Information System  https://obis.org
4  Ocean Biodiversity Information System  https://obis.org

                          type score  \
0  https://schema.org/Dataset   1.0
1  https://schema.org/Dataset   1.0
2  https://schema.org/Dataset   1.0
3  https://schema.org/Dataset   1.0
4  https://schema.org/Dataset   1.0

                                              name  \
0  Coral Reef Evaluation and Monitoring Project F...
1  Coral Reef Evaluation and Monitoring Project D...
2  Coral Reef Evaluation and Monitoring Project F...
3  Coral Reef Evaluation and Monitoring Project F...
4  Interacciones entre Corales y CÃ©spedes algale...

                                               url    lit  \
0  https://obis.org/dataset/b91d89db-79d6-4bd3-84...  coral
1  https://obis.org/dataset/46005357-02b8-4f17-b0...  coral
2  https://obis.org/dataset/d4ec17b8-fc96-49b9-b7...  coral
3  https://obis.org/dataset/36bca81c-6d77-4fd4-a9...  coral
4  https://obis.org/dataset/e39be6ef-3c91-4e97-ba...  coral

                                       description headline
0  The purpose of the Coral Reef Evaluation and M...     None
1  The purpose of the Coral Reef Evaluation and M...     None
2  The purpose of the Coral Reef Evaluation and M...     None
3  The purpose of the Coral Reef Evaluation and M...     None
4  Para el componente denominado â□□InteracciÃ³n ...     None
```

AquaDocs Alignment Testing

When pulling in documents using the graph only pattern there can be issues with how the prov looks and works for one large document vs many individual ones. This is a test of that

```
df = get_sparql_dataframe(oihad, rq_main)
df.head(5)
```

```
                                                   g  \
0  urn:gleaner:summoned:obps:466a3efb3402cd6c0b53...
1  urn:gleaner:summoned:obps:466a3efb3402cd6c0b53...
2  urn:gleaner:summoned:obps:466a3efb3402cd6c0b53...
3  urn:gleaner:summoned:obps:466a3efb3402cd6c0b53...
4  urn:gleaner:summoned:obps:466a3efb3402cd6c0b53...

                                                   s  \
0  oai:repository.oceanbestpractices.org:11329/1350
1   oai:repository.oceanbestpractices.org:11329/447
2   oai:repository.oceanbestpractices.org:11329/445
3   oai:repository.oceanbestpractices.org:11329/448
4   oai:repository.oceanbestpractices.org:11329/760

                                        wat               orgname  \
0  https://www.re3data.org/repository/obps  Ocean Best Practices
1  https://www.re3data.org/repository/obps  Ocean Best Practices
2  https://www.re3data.org/repository/obps  Ocean Best Practices
3  https://www.re3data.org/repository/obps  Ocean Best Practices
4  https://www.re3data.org/repository/obps  Ocean Best Practices

                               domain                              type  score  \
0  https://oih.oceanbestpractices.org  https://schema.org/CreativeWork  0.625
1  https://oih.oceanbestpractices.org  https://schema.org/CreativeWork  0.625
2  https://oih.oceanbestpractices.org  https://schema.org/CreativeWork  0.625
3  https://oih.oceanbestpractices.org  https://schema.org/CreativeWork  0.625
4  https://oih.oceanbestpractices.org  https://schema.org/CreativeWork  0.625

                                             name  \
0  Next-Generation Optical Sensing Technologies f...
1  Standardised survey procedures for monitoring ...
2  Best practices in RNA & DNA sample preparation...
3  Systematic global assessment of reef fish comm...
4  Recommendations for best practice in deep-sea ...

                                          url            lit  \
0  https://www.oceandocs.org/handle/11329/1350    Coral reefs
1   https://www.oceandocs.org/handle/11329/447    Coral reefs
2   https://www.oceandocs.org/handle/11329/445    Coral reefs
3   https://www.oceandocs.org/handle/11329/448    Coral reefs
4   https://www.oceandocs.org/handle/11329/760  Coral gardens

                                       description headline
0   - We highlight three emerging NASA optical te...     None
1   - This manual describes the standard Reef   L...     None
2   - When the ocean  is  dying,  the  planet  is...     None
3   - Discussion of the Reef Life Survey Methods ...     None
4   - We assert that the reef framework-forming c...     None
```

Query for prov

Count (count(distinct ?tag) as ?count)

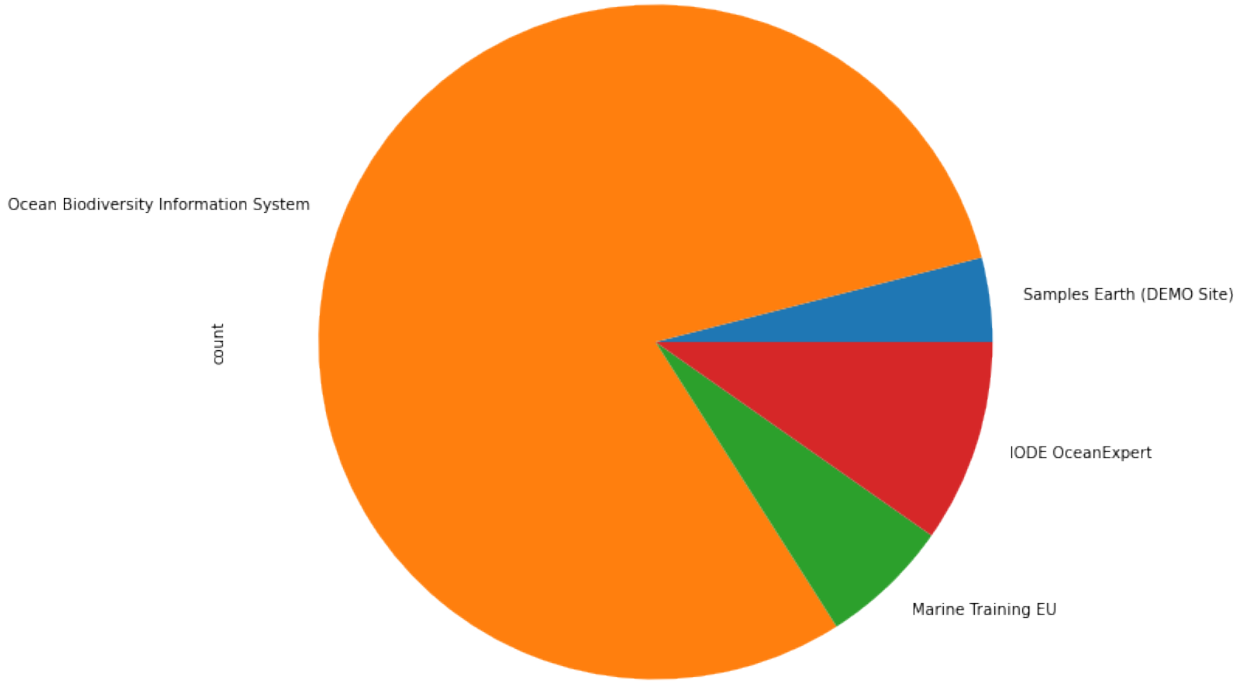
Need to look for the date in the prov record too. I keep it by day granularity, so I should be able to see the difference if I focos on a specific repo or look over the dates

```
rq_prov = """prefix prov: <http://www.w3.org/ns/prov#>
        PREFIX con: <http://www.ontotext.com/connectors/lucene#>
        PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
        PREFIX con-inst: <http://www.ontotext.com/connectors/lucene/instance#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX schema: <https://schema.org/>
        PREFIX schemaold: <http://schema.org/>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>


       SELECT   ( COUNT(?hm) as ?count) ?wat  ?orgname ?domain
        WHERE
        {
           ?hm prov:wasAttributedTo ?wat .
           ?wat rdf:name ?orgname .
           ?wat rdfs:seeAlso ?domain
        }
        GROUP BY ?wat ?orgname ?domain
        """
```

```
dfp = get_sparql_dataframe(oih, rq_prov)
dfp['count'] = dfp["count"].astype(int) # convert count c to int
dfp.set_index('orgname', inplace=True)
dfp.head(10)
```

```
                                    count  \
orgname
Samples Earth (DEMO Site)             202
Ocean Biodiversity Information System  4007
Marine Training EU                     313
IODE OceanExpert                       487


                                                                                 ↵
 ↪wat  \
orgname
Samples Earth (DEMO Site)              https://www.re3data.org/repository/
 ↪samplesearth
Ocean Biodiversity Information System          https://www.re3data.org/repository/
 ↪obis
Marine Training EU                    https://www.re3data.org/repository/
 ↪marinetraining
IODE OceanExpert                        https://www.re3data.org/repository/
 ↪oceanexpert

                                                          domain
orgname
Samples Earth (DEMO Site)                   https://samples.earth
Ocean Biodiversity Information System            https://obis.org
Marine Training EU                    https://marinetraining.eu/
IODE OceanExpert                        https://oceanexpert.org/
```

```
plot = dfp.plot.pie(y='count',legend=False, figsize=(10, 10))
```




```
rq_provdate = """prefix prov: <http://www.w3.org/ns/prov#>
        PREFIX con: <http://www.ontotext.com/connectors/lucene#>
        PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
        PREFIX con-inst: <http://www.ontotext.com/connectors/lucene/instance#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX schema: <https://schema.org/>
        PREFIX schemaold: <http://schema.org/>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>


       SELECT   ( COUNT(?s) as ?count) ?time ?orgname
        WHERE
        {
           ?s a prov:Activity  .
           ?s prov:endedAtTime ?time .
           ?s prov:generated ?gen .
           ?s prov:used ?used .
           ?used prov:hadMember ?mem .
           ?mem prov:wasAttributedTo ?wat .
           ?wat rdf:name ?orgname .
           ?wat rdfs:seeAlso ?domain
        }
        GROUP BY ?time ?orgname
        """
```

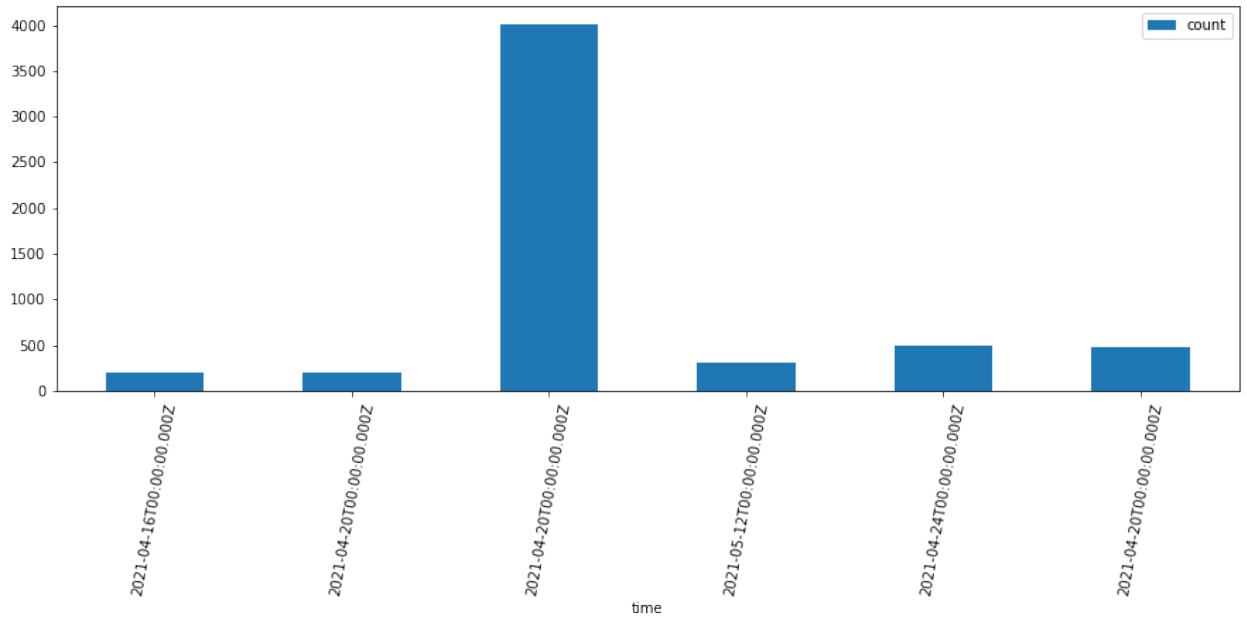
```
dfpd = get_sparql_dataframe(oih, rq_provdate)
dfpd.head(10)
```

```
  count                       time                                  orgname
0  4007  2021-04-20T00:00:00.000Z  Ocean Biodiversity Information System
1   202  2021-04-16T00:00:00.000Z              Samples Earth (DEMO Site)
2   202  2021-04-20T00:00:00.000Z              Samples Earth (DEMO Site)
3   487  2021-04-24T00:00:00.000Z                        IODE OceanExpert
4   313  2021-05-12T00:00:00.000Z                       Marine Training EU
5   485  2021-04-20T00:00:00.000Z                        IODE OceanExpert
```

```
dfpd = get_sparql_dataframe(oih, rq_provdate)
dfpd['count'] = dfpd["count"].astype(int) # convert count c to int
dfpd.set_index('time', inplace=True)
dfpd.head()
```

```
                          count                                  orgname
time
2021-04-16T00:00:00.000Z    202              Samples Earth (DEMO Site)
2021-04-20T00:00:00.000Z    202              Samples Earth (DEMO Site)
2021-04-20T00:00:00.000Z   4007  Ocean Biodiversity Information System
2021-05-12T00:00:00.000Z    313                       Marine Training EU
2021-04-24T00:00:00.000Z    487                        IODE OceanExpert
```

```
ax = dfpd.plot.bar(rot=80, stacked=True, figsize=(15, 5))
```



Feed query

Goal here is see if the prov will give us the elements for an RSS feed.The RSS specs give us the elements we need to populate. Focus on; title(name), date, author, description

- Element Description Example
- title The title of the item. Venice Film Festival Tries to Quit Sinking
- link The URL of the item. http://www.nytimes.com/2002/09/07/movies/07FEST.html
- description The item synopsis. Some of the most heated chatter at the Venice Film Festival this week was about the way that the arrival of the stars at the Palazzo del Cinema was being staged.
- author Email address of the author of the item. More. oprah@oxygen.net
- category Includes the item in one or more categories. More. Simpsons Characters
- comments URL of a page for comments relating to the item. More. http://www.myblog.org/cgi-local/mt/mt-comments.cgi?entry_id=290
- enclosure Describes a media object that is attached to the item. More.
- guid A string that uniquely identifies the item. More. http://inessential.com/2002/09/01.php#a2
- pubDate Indicates when the item was published. More. Sun, 19 May 2002 15:21:36 GMT
- source The RSS channel that the item came from. More. Quotes of the Day

```
rq_provdatelist = """prefix prov: <http://www.w3.org/ns/prov#>
        PREFIX con: <http://www.ontotext.com/connectors/lucene#>
        PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
        PREFIX con-inst: <http://www.ontotext.com/connectors/lucene/instance#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX schema: <https://schema.org/>
        PREFIX schemaold: <http://schema.org/>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

       SELECT  ?time ?orgname ?memval  ?memname ?memdesc
        WHERE
        {
           ?s a prov:Activity  .
           ?s prov:endedAtTime ?time .
           ?s prov:generated ?gen .
           ?s prov:used ?used .
           ?used prov:hadMember ?mem .
           ?mem prov:value ?memval .
           ?mem schema:name ?memname .
           ?mem schema:description ?memdesc .
           ?mem prov:wasAttributedTo ?wat .
           ?wat rdf:name ?orgname .
           ?wat rdfs:seeAlso ?domain
        }
        ORDER BY DESC(?time)
        LIMIT 1000

        """
```

```
dfpl = get_sparql_dataframe(oih, rq_provdatelist)
dfpl.head(10)
```

```
                        time             orgname  \
0  2021-05-12T00:00:00.000Z  Marine Training EU
1  2021-05-12T00:00:00.000Z  Marine Training EU
2  2021-05-12T00:00:00.000Z  Marine Training EU
3  2021-05-12T00:00:00.000Z  Marine Training EU
4  2021-05-12T00:00:00.000Z  Marine Training EU
5  2021-05-12T00:00:00.000Z  Marine Training EU
6  2021-05-12T00:00:00.000Z  Marine Training EU
7  2021-05-12T00:00:00.000Z  Marine Training EU
8  2021-05-12T00:00:00.000Z  Marine Training EU
9  2021-05-12T00:00:00.000Z  Marine Training EU

                                    memval  \
0  https://www.marinetraining.eu/node/4051
1  https://www.marinetraining.eu/node/3978
2  https://www.marinetraining.eu/node/4394
3  https://www.marinetraining.eu/node/4338
4  https://www.marinetraining.eu/node/4116
5  https://www.marinetraining.eu/node/4287
6  https://www.marinetraining.eu/node/4089
7  https://www.marinetraining.eu/node/4330
8  https://www.marinetraining.eu/node/4005
9  https://www.marinetraining.eu/node/4396

                                           memname  \
0               Stochastic and Nonlinear Ocean Waves
1                                    Water Chemistry
2                                 Behavioural Ecology
3                    Geographical Information System
4                  Fisheries Ecology and Assessment
5  Research Methods for Fish, Marine and Freshwat...
6                                  Invasion Biology
7                 Icelandic Society and Environment
8                                  Marine Insurance
9                      Fish Reproduction (Graduate)

                                           memdesc
0  Kort om emnet\n\nDet gis en introduksjon til s...
1  Water chemistry introduces the principles and ...
2  Course Description:\n\nBoth undergraduates and...
3  Course Description:\n\nThe course provides an ...
4  General course objectives\n\nTo provide the pa...
5  Course Description:\n\nCurrent methods in stud...
6  Basic concepts in invasion biology. Overview o...
7  Course Description:\n\nThe course explores the...
8  The Marine Insurance law deals with the rules ...
9  Course Description:\n\nThe aim of the course i...
```

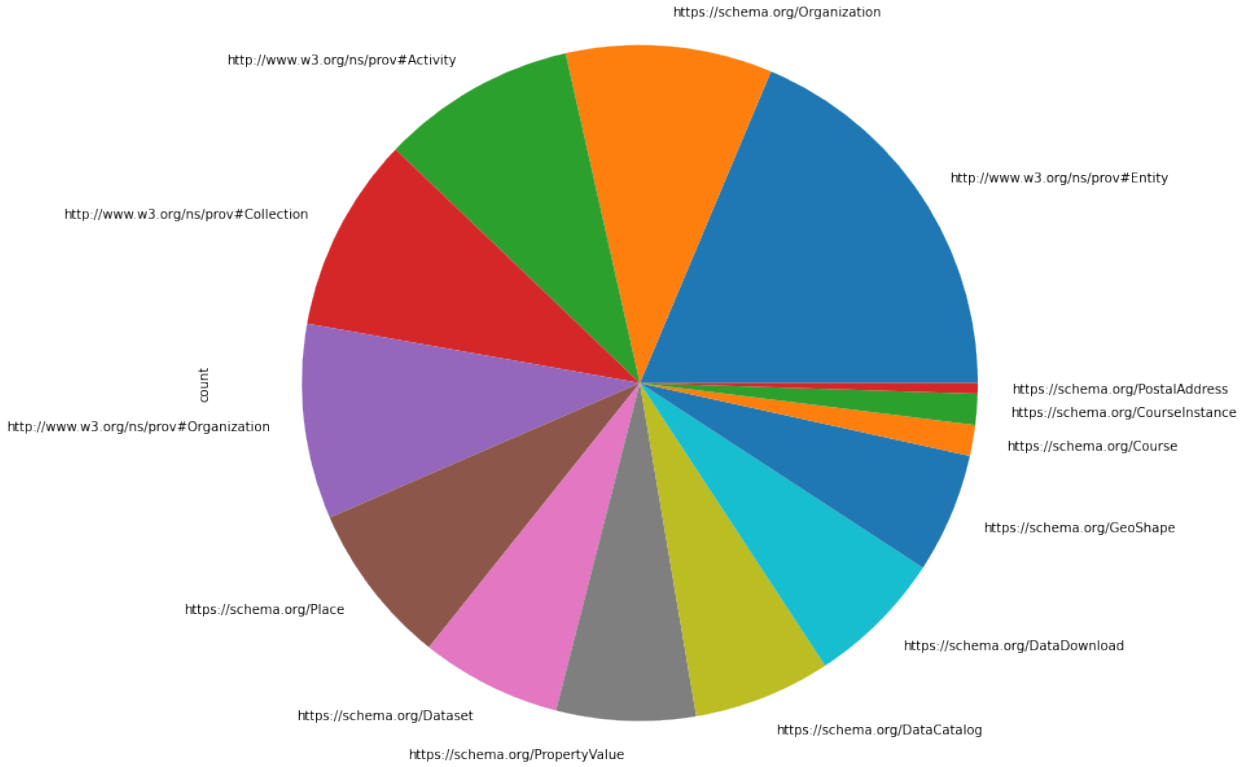
Types Breakdown

```
rq_types = """prefix prov: <http://www.w3.org/ns/prov#>
        PREFIX con: <http://www.ontotext.com/connectors/lucene#>
        PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
        PREFIX con-inst: <http://www.ontotext.com/connectors/lucene/instance#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX schema: <https://schema.org/>
        PREFIX schemaold: <http://schema.org/>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>


       SELECT   ( COUNT(?type) as ?count) ?type
        WHERE
        {
           ?s rdf:type ?type
        }
        GROUP BY ?type
        ORDER BY DESC(?count)
        """
```

```
dft = get_sparql_dataframe(oih, rq_types)
dft['count'] = dft["count"].astype(int) # convert count c to int
dft.set_index('type', inplace=True)
dft.head(10)
```

type	count
http://www.w3.org/ns/prov#Entity	11392
https://schema.org/Organization	6025
http://www.w3.org/ns/prov#Activity	5696
http://www.w3.org/ns/prov#Collection	5696
http://www.w3.org/ns/prov#Organization	5696
https://schema.org/Place	4742
https://schema.org/Dataset	4108
https://schema.org/PropertyValue	4068
https://schema.org/DataCatalog	4007
https://schema.org/DataDownload	4007

```
plot_t = dft.plot.pie(y='count',legend=False, figsize=(12, 12))
```



26.5.4 OIH Editing Playground

For more background on the larger ODIS Ocean Info Hub and related referneces please visit out GitHub repository.

Some package and imports

pip installs

```
#%%quiet
!pip install -q anytree
!pip install -q PyJSONViewer
!pip install -q qwikidata
!pip install -q SPARQLWrapper
!pip install -q Wikidata
!pip install -q pySHACL
!pip install -q 'PyLD>=2.0.3'
!pip install -q rdflib
!pip install -q graphviz
!pip install -q ipywidgets
```

imports

```
#@title
# General imports
import json
import rdflib
import requests
from rdflib import Graph, plugin
from rdflib.serializer import Serializer
from bs4 import BeautifulSoup
import urllib.request
from rdflib.extras.external_graph_libs import rdflib_to_networkx_multidigraph
from rdflib.extras.external_graph_libs import rdflib_to_networkx_graph
import networkx as nx
from networkx import Graph as NXGraph
import matplotlib.pyplot as plt
import statistics
import collections
from pyld import jsonld
from pyshacl import validate
import graphviz
```

functions

```
import graphviz
# from conceptnet5.uri import join_uri, split_uri
API_ROOT = 'http://api.conceptnet.io'


def short_name(value, max_length=40):
    """
    Convert an RDF value (given as a dictionary) to a reasonable label.
    """
    if value['type'] == 'blank node':
        return '_'
    elif value['type'] == 'IRI':
        url = value['value']
        if '#' in url:
            # Show just the fragment of URLs with a fragment
            # (it's probably a property name)
            return url.split('#')[-1]

        # Give URLs relative to the root of our API
        if url.startswith(API_ROOT):
            short_url = url[len(API_ROOT):]
            # If the URL is too long, hide it
            if len(short_url) > max_length:
                pieces = split_uri(short_url)
                return join_uri(pieces[0], '...')
            else:
                return short_url
        else:
            return url.split('://')[-1]
    else:
        # Put literal values in quotes
```

(continues on next page)

(continued from previous page)

```
        text = value['value'].replace(':', '')
        if len(text) > max_length:
            text = text[:max_length] + '...'
        return '"{}"'.format(text)


def show_graph(url, size=10):
    """
    Show the graph structure of a ConceptNet API response.
    """
    rdf = jsonld.normalize(url)['@default']
    graph = graphviz.Digraph(
        strict=False, graph_attr={'size': str(size), 'rankdir': 'LR'}
    )
    for edge in rdf:
        subj = short_name(edge['subject'])
        obj = short_name(edge['object'])
        pred = short_name(edge['predicate'])
        if subj and obj and pred:
            # Apply different styles to the nodes based on whether they're
            # literals, ConceptNet URLs, or other URLs
            if obj.startswith('"'):
                # Literal values
                graph.node(obj, penwidth='0')
            elif obj.startswith('/'):
                # ConceptNet nodes
                graph.node(obj, style='filled', fillcolor="#ddeeff")
            else:
                # Other URLs
                graph.node(obj, color="#558855")
            graph.edge(subj, obj, label=pred)

    return graph
```

```
#@title
def get_sparql_dataframe(service, query):
    """
    Helper function to convert SPARQL results into a Pandas data frame.
    """
    sparql = SPARQLWrapper(service)
    sparql.setQuery(query)
    sparql.setReturnFormat(JSON)
    result = sparql.query()

    processed_results = json.load(result.response)
    cols = processed_results['head']['vars']

    out = []
    for row in processed_results['results']['bindings']:
        item = []
        for c in cols:
            item.append(row.get(c, {}).get('value'))
        out.append(item)

    return pd.DataFrame(out, columns=cols)
```

```
from toggle_cell import toggle_code as hide_solution
```

```
<IPython.core.display.HTML object>
```

```
from datetime import datetime

now = datetime.now()
current_time = now.strftime("%H:%M:%S")
print("Current Time =", current_time)

# hide_solution()
```

```
Current Time = 11:39:51
```

```
# Fetch a single <1MB file using the raw GitHub URL.
!curl --remote-name \
     --location https://raw.githubusercontent.com/ESIPFed/science-on-schema.org/
 ↪master/examples/dataset/full.jsonld
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  9241  100  9241    0     0  35542      0 --:--:-- --:--:-- --:--:-- 35406
```

Editor playground

```
import ipywidgets as widgets

titleWidget = widgets.Text(
    description='Enter a title:',
    disabled=False
)

display(titleWidget)
```

```
Text(value='', description='Enter a title:')
```

```
print(titleWidget.value)
```

```

```

```
from __future__ import print_function
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets

def f(x):
    return x

interact(f, x='Hi there!');
```

```
interactive(children=(Text(value='Hi there!', description='x'), Output()), _dom_
 ↪classes=('widget-interact',))
```

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@type": "CreativeWork",
    "@id": "https://example.org/id/XYZ",
    "name": "Name or title of the document",
    "description": "Description of the creative work to aid in searching",
    "url":  "https://www.sample-data-repository.org/creativework/report.pdf"
}
```

```
{'@context': {'@vocab': 'https://schema.org/'},
 '@type': 'CreativeWork',
 '@id': 'https://example.org/id/XYZ',
 'name': 'Name or title of the document',
 'description': 'Description of the creative work to aid in searching',
 'url': 'https://www.sample-data-repository.org/creativework/report.pdf'}
```

Introduction to JSON-LD files

```
#@title Google Data Set Required+

name = 'Data Set Name one' #@param {type:"string"}
sdotype = 'http://schema.org/Dataset' #@param ["http://schema.org/Dataset", "http://
↪schema.org/DataCatalog"]
description = 'Descriptive text of the dataset.' #@param {type:"string"}
url = 'http://foo.org/data/distribution' #@param {type:"string"}
version = 'version' #@param {type:"string"}
license = 'CC-BY-4.0' #@param ["CC-BY-4.0", "CC-0"]
keywords = 'geochemistry, Earth System Modeling, climate change' #@param {type:"string
↪"}
```

```
Data Set Name one
Descriptive text of the dataset.
```

```
#@title
from pyld import jsonld
import json

doc = {}
doc["https://schema.org/name"] = name
doc["@type"] = sdotype
doc["@id"] = "http://cooldata.io/id/doc/1"
doc["https://schema.org/description"] = description
doc["https://schema.org/url"] =  url
doc["https://schema.org/version"] =  version
doc["https://schema.org/license"] =  license

# parse comma seperated keywords, clean white spaces
k = keywords.split(",")
kp = []
for i in k:
  j = i.strip()
  kp.append(j)
```

(continues on next page)

(continued from previous page)

```
doc["http://schema.org/keywords"] =  kp


context = {
    "@vocab": "https://schema.org/",
}

# compact a document according to a particular context
# see: http://json-ld.org/spec/latest/json-ld/#compacted-document-form
compacted = jsonld.compact(doc, context)

jd = json.dumps(compacted, indent=4)
print(jd)
```

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "http://cooldata.io/id/doc/1",
    "@type": "http://schema.org/Dataset",
    "http://schema.org/keywords": [
        "geochemistry",
        "Earth System Modeling",
        "climate change"
    ],
    "description": "Descriptive text of the dataset.",
    "license": "CC-BY-4.0",
    "name": "Data Set Name one",
    "url": "http://foo.org/data/distribution",
    "version": "version"
}
```

```
show_graph(doc, size=30)
```

```
<graphviz.dot.Digraph at 0x7f345a95a940>
```

Framing

Understanding Framing is not a first order concern. However, understanding it and what it does can help you to think about how your data graph will be used.

Let's make a frame that allows us to view only the elements of the JSON-LD data graph that we are interested in. In this case let's target the keywords.

```
urlf = "https://raw.githubusercontent.com/ESIPFed/science-on-schema.org/master/
 ↪examples/dataset/minimal.jsonld"

frame = {
  "@context": {"@vocab": "http://schema.org/"},
  "@explicit": "true",
   "@type":     "Dataset",
  "keywords": "",
}
```

(continues on next page)

(continued from previous page)

```
framed = jsonld.frame(doc, frame)
print(framed)

show_graph(framed)
```

```
{'@context': {'@vocab': 'http://schema.org/'}, '@id': 'http://cooldata.io/id/doc/1',
 ↪'@type': 'Dataset', 'keywords': ['geochemistry', 'Earth System Modeling', 'climate␣
 ↪change']}
```

```
<graphviz.dot.Digraph at 0x7f345aa37e20>
```

Parse out the keywords

At this point we can now take out resulting JSON-LD graph and extract the items we are interested int.

```
#g = framed['@graph']  # get the graph
#kw = g[0]['keywords']  # get the keywords (you could do this in 1 line, 2 here for␣
 ↪exposition)


kw = framed['keywords']  # get the keywords (you could do this in 1 line, 2 here for␣
 ↪exposition)


print("We have the individual keywords, what shall we do with them?")
for w in kw:
  print(w)
```

```
We have the individual keywords, what shall we do with them?
geochemistry
Earth System Modeling
climate change
```

Wikidata

So, we have used framing to arrive at a way to link to other graphs. A nice use of framing but perhaps more important is the linking. This exercise has shown how linking across graph gets us from strings to things.

Let's see if we can query wikidata and pull back some concepts. We will see if antyhing we use as a keyword aligns with a JSTOR (https://www.jstor.org/) topic.

Reference also: https://www.wikidata.org/wiki/Wikidata:WikidataCon_2019/Program/Sessions/Lightning_talks_2

```
from SPARQLWrapper import SPARQLWrapper, JSON
import pandas as pd
import json

dbsparql = "http://dbpedia.org/sparql"
ufokn = "http://graph.openknowledge.network/blazegraph/namespace/demo/sparql"
wikidata = "https://query.wikidata.org/sparql"
```

In the example below we can BIND in keywords like geochemistry but also other terms we may extract from the name, keywords and description.

```
rq = '''SELECT ?term ?topic WHERE {{ BIND ( "{var}" as ?term ) ?topic wdt:P3827 ?term
 ↪. }}'''.format(var="geochemistry")
test = get_sparql_dataframe(wikidata, rq)
test.head()
```

```
           term                                        topic
0  geochemistry  http://www.wikidata.org/entity/Q161764
```

```
#df = pd.DataFrame(columns=['term', 'topic'])

for w in range(len(kw)):
  print(kw[w])
  rq = '''SELECT ?term ?topic WHERE {{ BIND ( "{var}" as ?term ) ?topic wdt:P3827 ?
 ↪term . }}'''.format(var=kw[w])
  sdf = get_sparql_dataframe(wikidata, rq)
  #df.append(sdf)
  x = sdf.head()
  print(x)
  print("-------------------------------------")

#df.head()
```

```
geochemistry
           term                                        topic
0  geochemistry  http://www.wikidata.org/entity/Q161764
-------------------------------------
Earth System Modeling
Empty DataFrame
Columns: [term, topic]
Index: []
-------------------------------------
climate change
Empty DataFrame
Columns: [term, topic]
Index: []
-------------------------------------
```

```
from wikidata.client import Client
from urllib.parse import urlparse

client = Client()

for row in test.iterrows():
    o = urlparse(row[1]['topic'])
    e = o.path.rsplit('/', 1)[-1]
    entity = client.get('Q161764', load=True)
    print('---\nTerm:{} \nURL:{} \ndescription:{}'.format(row[1]['term'], row[1][
 ↪'topic'],  entity.description))
```

```
---
Term:geochemistry
URL:http://www.wikidata.org/entity/Q161764
```

(continues on next page)

(continued from previous page)

```
description:science that applies chemistry to geological systems
```

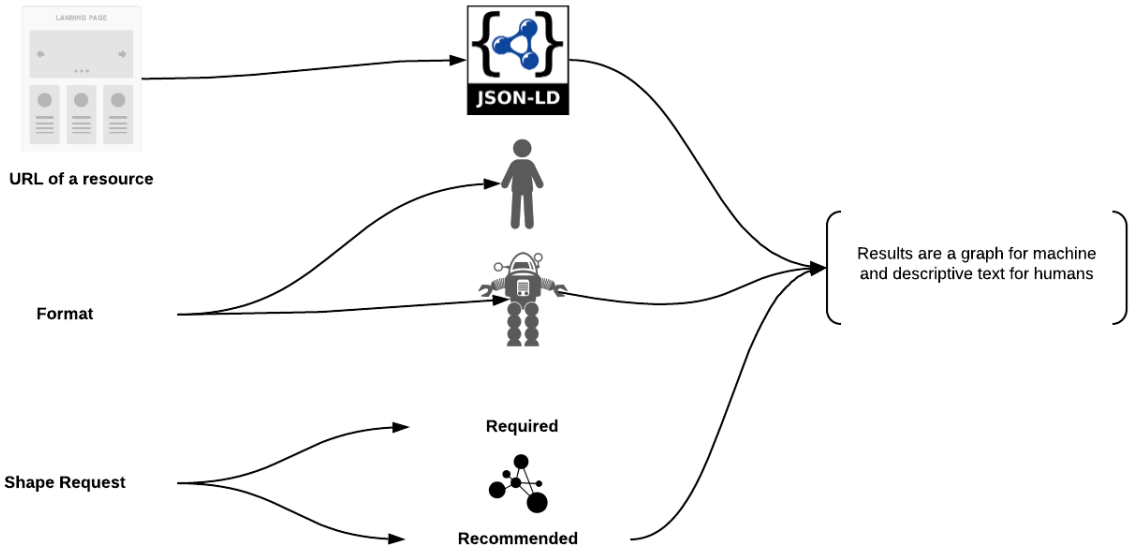
26.5.5 Validation

About

The code below invokes pySHACL on some data and shape graphs out of GitHub. Note, we could edit these local to this notebook too. The human output is a bit hard to read since some of the encoding is off.

It might actually work to use the graph output and route it through the graph package and into Pandas too. It might let us parse and present the results a bit better.

The imaage below is just a test of putting images into this document. We can also upload and associate a document with the notebook and use it locally too.




```
from pyshacl import validate
import json
import rdflib
from rdflib.extras.external_graph_libs import rdflib_to_networkx_multidigraph
import requests
from rdflib import Graph, plugin
from rdflib.serializer import Serializer

from bs4 import BeautifulSoup
import urllib.request

dg = 'https://raw.githubusercontent.com/ESIPFed/science-on-schema.org/master/examples/
 ↪dataset/minimal.jsonld'
sg = 'https://raw.githubusercontent.com/geoschemas-org/geoshapes/master/shapegraphs/
 ↪googleRecommendedCoverageCheck.ttl'

s = rdflib.Graph()
```

(continues on next page)

(continued from previous page)

```
sr = s.parse(sg, format="ttl")
d = rdflib.Graph()
dr = d.parse(dg, format="json-ld")

conforms, v_graph, v_text = validate(dr, shacl_graph=sr,
  data_graph_format="json-ld",
  shacl_graph_format="ttl",
  inference='none', debug=False,
  serialize_report_graph=False)

print('{} {}'.format(conforms, v_text))
```

```
True Validation Report
Conforms: True
```

26.5.6 Thematic topic: Documents

About

A testing area for the work on type Document

```
# Load examples from ODIS-Arch
!curl --remote-name \
     --location https://raw.githubusercontent.com/iodepo/odis-arch/master/schema/
 ↪thematics/docs/graphs/doc.json
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   888  100   888    0     0   5016      0 --:--:-- --:--:-- --:--:--  5016
```

```
# read into var

with open('/content/doc.json', 'r') as file:
    docstring = file.read()

docjson = json.loads(docstring)

# could I %load the file via magic commands?  see all via %lsmagic
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-31-ef7bc9a603c6> in <module>
      1 # read into var
      2
----> 3 with open('/content/doc.json', 'r') as file:
      4     docstring = file.read()
      5

FileNotFoundError: [Errno 2] No such file or directory: '/content/doc.json'
```

Dev note

During development it would good to edit in the notebook and process the results through testing and viz. So we have to edit the JSON in the notebook, which is hideous.

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/XYZ",
    "@type": "Dataset",
    "description": "Description of the dataset to aid in searching",
    "distribution": {
        "@type": "DataDownload",
        "contentUrl": "https://www.sample-data-repository.org/dataset/472032.tsv",
        "encodingFormat": "text/tab-separated-values"
    },
    "maintainer": {
        "@id": "https://link.to/PID_like_re3_or_others",
        "@type": "Organization",
        "description": "Organization or Person who maintains the creative work"
    },
    "name": "Name or title of the document",
    "subjectOf": {
        "@type": "DataDownload",
        "dateModified": "2019-06-12T14:44:15Z",
        "description": "EML metadata describing the dataset",
        "encodingFormat": [
            "application/xml",
            "https://eml.ecoinformatics.org/eml-2.2.0"
        ],
        "name": "eml-metadatafile.xml"
    }
}
```

```
{'@context': {'@vocab': 'https://schema.org/'},
 '@id': 'https://example.org/id/XYZ',
 '@type': 'Dataset',
 'description': 'Description of the dataset to aid in searching',
 'distribution': {'@type': 'DataDownload',
  'contentUrl': 'https://www.sample-data-repository.org/dataset/472032.tsv',
  'encodingFormat': 'text/tab-separated-values'},
 'maintainer': {'@id': 'https://link.to/PID_like_re3_or_others',
  '@type': 'Organization',
  'description': 'Organization or Person who maintains the creative work'},
 'name': 'Name or title of the document',
 'subjectOf': {'@type': 'DataDownload',
  'dateModified': '2019-06-12T14:44:15Z',
  'description': 'EML metadata describing the dataset',
  'encodingFormat': ['application/xml',
   'https://eml.ecoinformatics.org/eml-2.2.0'],
  'name': 'eml-metadatafile.xml'}}
```

```
context = {
    "@vocab": "https://schema.org/",
}
```

(continues on next page)

(continued from previous page)

```
# compact a document according to a particular context
# see: http://json-ld.org/spec/latest/json-ld/#compacted-document-form
compacted = jsonld.compact(_, context)  # CAUTION..  note _ which is reading the␣
↪previous cell output...

jd = json.dumps(compacted, indent=4)
print(jd)
```

```
{
    "@context": {
        "@vocab": "https://schema.org/"
    },
    "@id": "https://example.org/id/XYZ",
    "@type": "Dataset",
    "description": "Description of the dataset to aid in searching",
    "distribution": {
        "@type": "DataDownload",
        "contentUrl": "https://www.sample-data-repository.org/dataset/472032.tsv",
        "encodingFormat": "text/tab-separated-values"
    },
    "maintainer": {
        "@id": "https://link.to/PID_like_re3_or_others",
        "@type": "Organization",
        "description": "Organization or Person who maintains the creative work"
    },
    "name": "Name or title of the document",
    "subjectOf": {
        "@type": "DataDownload",
        "dateModified": "2019-06-12T14:44:15Z",
        "description": "EML metadata describing the dataset",
        "encodingFormat": [
            "application/xml",
            "https://eml.ecoinformatics.org/eml-2.2.0"
        ],
        "name": "eml-metadatafile.xml"
    }
}
```

```
show_graph(docjson)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-34-d4281ae8ca4c> in <module>
----> 1 show_graph(docjson)

NameError: name 'docjson' is not defined
```

26.5.7 SHACL Validation For Warehouses

Ocean Info Hub SHACL validation on S3(minio) objects

It should be noted here that SHACL validation is not a service OIH offers. Rather, the validation is a capacity that the OIH architectural approach facilities. Further this validation follows W3C recommendations as describted in https://www.w3.org/TR/shacl/.

Flow

- get an object (use the dask notebook)
- process the object against OIH SHACL shapes

```
%%capture
!pip install pyshacl
!pip install 'PyLD>=2.0.3'
!pip install flatten_json
!pip install 'fsspec>=0.3.3'
!pip install s3fs
!pip install boto3
!pip install seaborn
!pip install dask
```

```
def label_status (row):
  result = row['http://www.w3.org/ns/shacl#resultSeverity']
  if result == "nan":
    return "NA"
  elif "Warning" in result:
    return "Warning"
  elif "Violation" in result:
    return "Violation"
  else:
    return result

def source_shape (row):
  result = row['http://www.w3.org/ns/shacl#sourceShape']
  if type(result) is list:
    return result[0]['@id']
  else:
    return "NA"
```

Gleaner Data

First lets load up some of the data Gleaner has collected. This is just simple data graph objects and not any graphs or other processed products from Gleaner.

```
# Set up our S3FileSystem object
import s3fs

oss = s3fs.S3FileSystem(
      anon=True,
      key="",
      secret="",
      client_kwargs = {"endpoint_url":"https://oss.collaborium.io"}
   )
```

```
# Create the Dask tasks..   created..  not run..
import json
import dask, boto3
import dask.dataframe as dd

@dask.delayed()
def read_a_file(fn):
    # or preferably open in text mode and json.load from the file
    with oss.open(fn, 'rb') as f:
        #return json.loads(f.read().replace('\n',' '))
        return json.loads(f.read().decode("ascii", "ignore").replace('\n',' '))

# List of buckets to work with..   if you don't know them, you could print out above
buckets = ['gleaner/summoned/oceanexperts']
filenames = []

for d in range(len(buckets)):
  print("indexing {}".format(buckets[d]))
  f = oss.ls(buckets[d])
  filenames += f

#filenames = oss.cat('gleaner/summoned/opentopo', recursive=True)
output = [read_a_file(f) for f in filenames]
print(len(filenames))
# print(filenames)
```

```
indexing gleaner/summoned/oceanexperts
481
```

```
%%time
from pyshacl import validate
from os import path
from pandas import json_normalize
import pandas as pd
import json
import rdflib
import seaborn as sns
import matplotlib.pyplot as plt

gldf = pd.DataFrame(columns=["id", "status", "shape"])

for ndx in range(len(output)):
# for ndx in range(10):

  if "/.jsonld" not in filenames[ndx] :
    try:
      jld = output[ndx].compute()  ## Now pull from dask..   In REAL version, move␣
↪this logic into Dask!  to get the parallel approach
    except:
      print(filenames[ndx])
      print("Doc has bad encoding")

    jd = json.dumps(jld, sort_keys=True, indent=4)

    try:
      conforms, v_graph, v_text = validate(jd,
                shacl_graph='./oih_learning.ttl',
```

(continues on next page)

(continued from previous page)

```
                data_graph_format="json-ld",
                shape_graph_format="ttl",
                inference='none',
                serialize_report_graph="json-ld")

      gd = v_graph.decode("ascii")
      df = pd.DataFrame(json.loads(gd))
      conforms = df["http://www.w3.org/ns/shacl#conforms"]
      tf = conforms[0][0]['@value']

      if "False" in str(tf):
        df['http://www.w3.org/ns/shacl#resultSeverity'] = df['http://www.w3.org/ns/
↪shacl#resultSeverity'].astype(str)
        df['ID'] = filenames[ndx] #  'Object:{}'.format(ndx)
        df['Status'] = df.apply (lambda row: label_status(row), axis=1)
        df['Shape'] = df.apply (lambda row: source_shape(row), axis=1)

        data = [df["ID"], df["Status"], df['Shape']]
        headers = ["id", "status", "shape"]
        df3 = pd.concat(data, axis=1, keys=headers)
        gldf = gldf.append(df3, ignore_index=True)
      elif "True" in str(tf):
        df['ID'] = filenames[ndx] #  'Object:{}'.format(ndx)
        df['Status'] = "Valid"
        df['Shape'] = "AllPassed"

        data = [df["ID"], df["Status"], df['Shape']]
        headers = ["id", "status", "shape"]
        df3 = pd.concat(data, axis=1, keys=headers)
        gldf = gldf.append(df3, ignore_index=True)

#       print("------------------")
#       print(conforms)
#       print(v_graph)
#       print(v_text)

    except:
      print("ERROR")
      df = pd.DataFrame()
      df['ID'] = filenames[ndx] #  'Object:{}'.format(ndx)
      df['Status'] = "ErrorProcessing"
      df['Shape'] = "ErrorProcessing"

      data = [df["ID"], df["Status"], df['Shape']]
      headers = ["id", "status", "shape"]
      df3 = pd.concat(data, axis=1, keys=headers)
      gldf = gldf.append(df3, ignore_index=True)
      print("PySHACL decode error: {}",format(filenames[ndx]))
```

```
CPU times: user 3.62 s, sys: 132 ms, total: 3.75 s
Wall time: 23.2 s
```

```
gldf.info()
gldf.head(5)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 481 entries, 0 to 480
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      481 non-null    object
 1   status  481 non-null    object
 2   shape   481 non-null    object
dtypes: object(3)
memory usage: 11.4+ KB
```

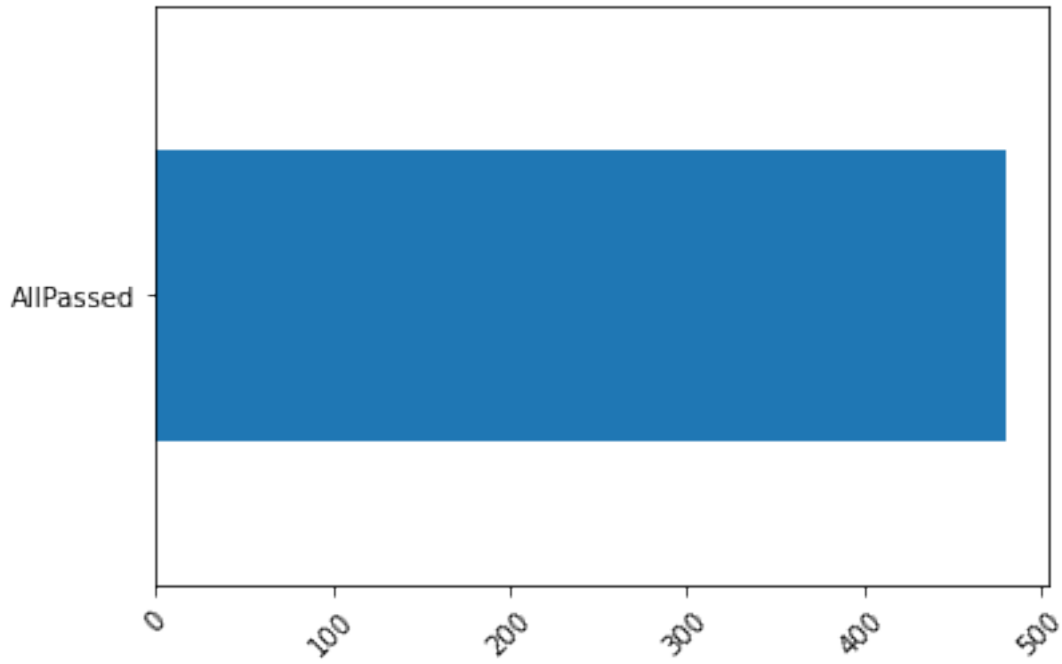
```
                                                  id status      shape
0  gleaner/summoned/oceanexperts/00eae339a41708c6...  Valid  AllPassed
1  gleaner/summoned/oceanexperts/014dbf631db7b122...  Valid  AllPassed
2  gleaner/summoned/oceanexperts/019224fb3174aace...  Valid  AllPassed
3  gleaner/summoned/oceanexperts/0223a997319c102b...  Valid  AllPassed
4  gleaner/summoned/oceanexperts/022ac35a670a36a3...  Valid  AllPassed
```

```
pd.value_counts(gldf['shape'])
```

```
AllPassed    481
Name: shape, dtype: int64
```

```
pd.value_counts(gldf['shape']).plot.barh()
plt.xticks(rotation=45)
```

```
(array([  0., 100., 200., 300., 400., 500., 600.]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



26.5.8 ISO 19139 To Schema.org

This work is an implementation of Steve Richard's work at: https://github.com/usgin/metadataTransforms/tree/master/iso-19139-to-HTMLwSDO

It demonstrates a simple transform from an ISO record to JSON-LD and schema.org. There are alternatie paths to HTML + embedded JSON-LD that can be found in the examples directory of the repository.

Refs:

- https://lxml.de/index.html
- https://www.seadatanet.org/content/download/4534/file/CDI_ISO19139_full_example_12.2.0.xml
- https://raw.githubusercontent.com/usgin/metadataTransforms/master/iso-19139-to-HTMLwSDO/ISO19139ToSchemaOrgDataset1.0.xslt

```
!pip install -q lxml
```

```
import lxml.etree as ET
import urllib.request
```

```
!wget https://raw.githubusercontent.com/usgin/metadataTransforms/master/iso-19139-to-
↪HTMLwSDO/ISO19139ToSDODatasetStandalone1.0.xslt
```

```
--2021-06-29 20:58:05--  https://raw.githubusercontent.com/usgin/metadataTransforms/
↪master/iso-19139-to-HTMLwSDO/ISO19139ToSDODatasetStandalone1.0.xslt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133,␣
↪185.199.111.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.
↪133|:443... connected.
HTTP request sent, awaiting response... 200 OK
```

(continues on next page)

(continued from previous page)

```
Length: 79137 (77K) [text/plain]
Saving to: ‘ISO19139ToSDODatasetStandalone1.0.xslt.1’

ISO19139ToSDODatase 100%[===================>]  77.28K  --.-KB/s    in 0.01s

2021-06-29 20:58:05 (5.85 MB/s) - ‘ISO19139ToSDODatasetStandalone1.0.xslt.1’ saved␣
↪[79137/79137]
```

```
dom = ET.parse("./CDI_ISO19139_full_example_12.2.0.xml")
xslt = ET.parse("./ISO19139ToSDODatasetStandalone1.0.xslt") ## convert to JSON-LD␣
↪schema.org voc
transform = ET.XSLT(xslt)
newdom = transform(dom)
```

```
print (newdom)
```

```
{
  "@context": {
 "@vocab": "http://schema.org/",
"datacite": "http://purl.org/spar/datacite/",

                "earthcollab": "https://library.ucar.edu/earthcollab/schema#",

                "geolink": "http://schema.geolink.org/1.0/base/main#",

                "vivo": "http://vivoweb.org/ontology/core#",

                "dcat":"http://www.w3.org/ns/dcat#"

                  },
 "@id": "urn:urnSDNCDILOCALMARIS-TEST",
  "@type": "Dataset",
  "additionalType": [
    "geolink:Dataset",
    "vivo:Dataset"
  ],
  "name": "Test record with full coverage",
  "alternateName": "MARIS-TEST",
  "citation": "not provided, not provided, not provided, not provided, not provided␣
↪(2012-04-16), Test record with full coverage, urn:urnSDNCDILOCALMARIS-TEST.",
  "creator":
[{
    "@type":"Role",
      "roleName": "originator",
"creator": {

  "@type": "Role",
      "roleName": "originator"
}
},
{
    "@type":"Role",
      "roleName": "originator",
"creator": {
```

(continues on next page)

(continued from previous page)

```
  "@type": "Role",
      "roleName": "originator"
}
},
{
    "@type":"Role",
      "roleName": "originator",
"creator": {

  "@type": "Role",
      "roleName": "originator"
}
},
{
    "@type":"Role",
      "roleName": "originator",
"creator": {

  "@type": "Role",
      "roleName": "originator"
}
},
{
    "@type":"Role",
      "roleName": "originator",
"creator": {

  "@type": "Role",
      "roleName": "originator"
}
}],
  "datePublished": "2012-04-16",
  "description": "This record is meant for test purposes. It contains a value for␣
 ↪every field and multiple values whereever possible.",
  "distribution": [
{
      "@id": "http://www.sdn-taskmanager.org/",
    "@type": "DataDownload",
    "additionalType": "dcat:distribution",
      "dcat:accessURL": "http://www.sdn-taskmanager.org/",
      "url": "http://www.sdn-taskmanager.org/",
      "description": "DBTEST. Service Protocol: DBTEST. Link Function:␣
 ↪downloadRegistration-- manual interaction with an on-line system by registered␣
 ↪users following successful authentication and authorisation. ",
      "provider": {
    "@type":"Role",
      "roleName": "distributor",
"provider": {

  "@type": "Role",
      "roleName": "distributor"
}
},
      "fileFormat": [
"Ocean Data View ASCII input v.0.3", "MEDATLAS ASCII v.1"],
      "contentSize": "123 "}
,
```

(continues on next page)

(continued from previous page)


```
{
      "@id": "http://geoservice.maris2.nl/wms/seadatanet/seadatanet/?",
    "@type": "DataDownload",
    "additionalType": "dcat:distribution",
      "dcat:accessURL": "http://geoservice.maris2.nl/wms/seadatanet/seadatanet/?",
      "url": "http://geoservice.maris2.nl/wms/seadatanet/seadatanet/?",
      "description": "WMS example url. Service Protocol: WMS example url. Link␣
↪Function: URL-- online resource locator for accessing data using a specific web␣
↪protocol. ",
      "provider": {
    "@type":"Role",
      "roleName": "distributor",
"provider": {

  "@type": "Role",
      "roleName": "distributor"
}
},
      "fileFormat": [
"Ocean Data View ASCII input v.0.3", "MEDATLAS ASCII v.1"]}
  ],
  "identifier":
 [
 {
            "@type": "PropertyValue",
            "propertyID": "dataset identifier",
            "value": "urn:urnSDNCDILOCALMARIS-TEST"
}],
  "includedInDataCatalog": {

    "@type":"DataCatalog",
  "name":"Name of catalog source for record being transformed",
  "url": "not defined"
},
  "keywords": ["Oceanographic geographical features", "Atmospheric visibility and␣
↪transparency", "Ammonium concentration parameters in the water column",
↪"Atmospheric humidity", "aerosol samplers", "Differential Global Positioning System␣
↪receivers", "cetacean", "Integrated Ocean Drilling Program (IODP) - Artic␣
↪expedition (ACEX) {acronym=IODP organisation=Natural Environment Research Council␣
↪(NERC) country=United Kingdom}", "National Coastal Data Co-ordinator {acronym=␣
↪organisation=Department for Environment, Food and Rural Affairs (DEFRA)␣
↪country=United Kingdom}", "GEOWARN - Geo-spatial warning system Nisyros volcano␣
↪(Greece). An emergency case study. {acronym=GEOWARN organisation=Hellenic Centre␣
↪for Marine Research, Institute of Oceanography (HCMR/IO) country=Greece}"],
  "license": [{
     "@type": "DigitalDocument","name": "MD_Constraints",
        "description": "useLimitation: Not applicable.   "},
       {
     "@type": "DigitalDocument","name": "MD_LegalConstraints",
        "description": "accessConstraints: otherRestrictions.    otherConstraints: "},
       {
     "@type": "DigitalDocument","name": "MD_LegalConstraints",
        "description": "accessConstraints: otherRestrictions.    otherConstraints: "},
       {
     "@type": "DigitalDocument","name": "MD_LegalConstraints",
        "description": "accessConstraints: otherRestrictions.    otherConstraints: "}
↪],
```


(continues on next page)

(continued from previous page)

```
  "publisher": "publisher not specified",
  "spatialCoverage": {

    "@type":"Place",
      "geo":
        [
        {
       "@type": "GeoShape",
          "box": "-68.548849, 59.400296 -49.007153, 73.889864"
            },
                    {
       "@type": "GeoShape",
          "box": "106.574831, 5.916728 114.842479, 17.636232"
            },
                    {
       "@type": "GeoShape",
          "box": "-80.198605, -57.079131 -70.052045, -36.777203"
            }]
}}
```

26.5.9 DCAT Mapping

Testing approaches to mapping DCAT to schema.org

Current thinking

- JSON-LD Frame with default values
- SPARQL construct on these resulting frame to generate the new triples

Mapping references

- https://www.w3.org/2015/spatial/wiki/ISO_19115_-*DCAT*-_Schema.org_mapping
- https://ec-jrc.github.io/dcat-ap-to-schema-org/
- https://data.gov.au/data/dataset/67ca5de1-8774-4678-9d1b-8b1cb70ab33c.jsonld

Note: We should consider using the subjectOf property to link the generated schema.org to the source DCAT record where we can.

Methodology

We will load the DCAT JOSN-LD example and explore approaches to converting this to a form that can be used for schema.org.

Possible approaches include

- Inferencing
  - ref: https://derwen.ai/docs/kgl/infer/
- SPARQL CONSTRUCT
  - https://rdflib.readthedocs.io/en/stable/apidocs/rdflib.html
  - https://derwen.ai/docs/kgl/ex4_0/
- JSON-LD APIs

  - https://w3c.github.io/json-ld-framing/#omit-default-flag

- Context modification

```
!pip install -q kglab
```

```
ERROR: pip's dependency resolver does not currently take into account all the␣
 ↪packages that are installed. This behaviour is the source of the following␣
 ↪dependency conflicts.
boto3 1.17.102 requires botocore<1.21.0,>=1.20.102, but you have botocore 1.20.49␣
 ↪which is incompatible.
```

```
import kglab
import json
import rdflib
```

```
# load our JSON into a var to use later
f = open('dcatEx.json',)
j = json.load(f)
f.close()
```

JSON-LD

Use a frame to pull the elements we want to map, then alter the context for that frame or otehrwise cast to new namespace.

Frame with defaults and then work to convert to new names space with SPARQL construct

SPARQL CONSTRUCT example

Refs:

- https://derwen.ai/docs/kgl/ex4_0/

```
from icecream import ic
from pathlib import Path

txt = Path('dcatEx.json').read_text()

g = rdflib.Graph()
g.parse(data=txt, format="json-ld")
```

```
<Graph identifier=Nad1628ac8eb84e5881b07f2b9ac96afd (<class 'rdflib.graph.Graph'>)>
```

```
sparql = """
    SELECT ?s ?p ?o
    WHERE {
        ?s ?p ?o .
    }
    LIMIT 1
"""
```

```
for row in g.query(sparql):
    ic(row.asdict())
```

```
ic| row.asdict(): {'o': rdflib.term.Literal('LAND-Cover'),
                   'p': rdflib.term.URIRef('http://www.w3.org/ns/dcat#keyword'),
                   's': rdflib.term.URIRef('https://data.gov.au/dataset/67ca5de1-8774-
 ↪4678-9d1b-8b1cb70ab33c')}
```

```
sparqlc = """
 PREFIX dbpedia: <http://dbpedia.org/resource/>
 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
 PREFIX dc: <http://purl.org/dc/elements/1.1/>
 PREFIX dct: <http://purl.org/dc/terms/>
 PREFIX mo: <http://purl.org/ontology/mo/>
 PREFIX schema: <https://schema.org/>

CONSTRUCT {
       ?s schema:identifier ?o .
 }
 WHERE {
       ?s dct:identifier ?o .
 }
"""

qres = g.query(sparqlc)
context = {"@vocab": "https://schema.org/", "@language": "en"}
print(qres.serialize(format='json-ld', context=context, indent=4))

# g.parse(qres, format="nt")

# for row in qres:
#     print("-----")
#     print(row)
```

```
b'{\n    "@context": {\n        "@language": "en",\n        "@vocab": "https://schema.
 ↪org/"\n    },\n    "@id": "https://data.gov.au/dataset/67ca5de1-8774-4678-9d1b-
 ↪8b1cb70ab33c",\n    "identifier": {\n        "@value": "67ca5de1-8774-4678-9d1b-
 ↪8b1cb70ab33c"\n    }\n}'
```

```
import kglab

namespaces =  {
    "adms": "http://www.w3.org/ns/adms#",
    "dcat": "http://www.w3.org/ns/dcat#",
    "dct": "http://purl.org/dc/terms/",
    "foaf": "http://xmlns.com/foaf/0.1/",
    "gsp": "http://www.opengis.net/ont/geosparql#",
    "locn": "http://www.w3.org/ns/locn#",
    "owl": "http://www.w3.org/2002/07/owl#",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "schema": "http://schema.org/",
    "skos": "http://www.w3.org/2004/02/skos/core#",
    "time": "http://www.w3.org/2006/time",
    "vcard": "http://www.w3.org/2006/vcard/ns#",
    "xsd": "http://www.w3.org/2001/XMLSchema#"
  }

kg = kglab.KnowledgeGraph(
```

(continues on next page)

(continued from previous page)

```
    name = "DCAT example",
    base_uri = "https://www.example.org/",
    namespaces = namespaces,
    )

kg.load_jsonld("dcatEx.json")
```

```
<kglab.kglab.KnowledgeGraph at 0x7f702ccf86a0>
```

```
sparql2 = """
    SELECT ?s  ?o
    WHERE {
        ?s dct:description ?o .
    }
"""
```

```
import pandas as pd
pd.set_option("max_rows", None)

df = kg.query_as_df(sparql2)
df.head(20)
```

```
                                                   s  \
0  <https://data.gov.au/dataset/67ca5de1-8774-467...
1  <https://data.gov.au/dataset/67ca5de1-8774-467...

                                                   o
0                                          Data File
1  ## **Abstract** \n\nThis dataset and its metad...
```

```
pyvis_graph = kg.visualize_query(sparql2, notebook=True)

pyvis_graph.force_atlas_2based()
pyvis_graph.show("tmp.fig06.html")
```

```
<IPython.lib.display.IFrame at 0x7f702ccc4d00>
```

SHACL Rules

```
import pyshacl
```

```
from pyshacl import validate

conforms, v_graph, v_text = validate(data_graph="./learning.jsonld",
                shacl_graph='./oih_learning.ttl',
                data_graph_format="json-ld",
                shape_graph_format="ttl",
                inference='none',
                serialize_report_graph="json-ld")
```

```
print(conforms)
print(v_graph)
print(v_text)
```

```
True
b'[\n  {\n    "@id": "_:N8be15736f0b945d19b55b266f8475c54",\n    "@type": [\n
 ↪"http://www.w3.org/ns/shacl#ValidationReport"\n    ],\n    "http://www.w3.org/ns/
 ↪shacl#conforms": [\n      {\n        "@value": true\n      }\n    ]\n  }\n]'
Validation Report
Conforms: True
```

```
from pyshacl import Validator

# v = Validator(data_graph=dg_basin, shacl_graph=rule, options={"inference": "rdfs"},
 ↪ont_graph=ont)
# conforms, report_graph, report_text = v.run()
# expanded_graph = v.target_graph

df = Path('data.ttl').read_text()
dg = rdflib.Graph()
dg.parse(data=df, format="ttl")

sf = Path('shape.ttl').read_text()
sg = rdflib.Graph()
sg.parse(data=sf, format="ttl")

v = Validator(data_graph=dg, shacl_graph=sg,  options={"inference": "none", "advanced
 ↪": True})  # turn off rdfs inferencing
conforms, report_graph, report_text = v.run()
expanded_graph = v.target_graph
```

```
# print(conforms)
# print(v_graph)
# print("------------")
# print(v_text)
# print(expanded_graph)
```

```
print(expanded_graph.serialize(format="ttl").decode("utf-8"))
```

```
@prefix ex: <http://example.com/ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:InvalidRectangle a ex:Rectangle .

ex:NonSquareRectangle a ex:Rectangle ;
    ex:height 2 ;
    ex:width 3 .

ex:SquareRectangle a ex:Rectangle,
        ex:Square ;
    ex:height 4 ;
    ex:width 4 .
```

Notes on SHACL AF Rules

We need to add in PROV triples in this process to note the generation of these triples and the souce IRI tht results in the product IRI and the actor (?reference)

Maybe review: https://www.w3.org/TR/2013/REC-prov-o-20130430/#qualifiedPrimarySource

```
df = Path('dcat.ttl').read_text()
dg = rdflib.Graph()
dg.parse(data=df, format="ttl")

sf = Path('dcatsdo.ttl').read_text()
sg = rdflib.Graph()
sg.parse(data=sf, format="ttl")

v = Validator(data_graph=dg, shacl_graph=sg,  options={"inference": "none", "advanced
 ↪": True})  # turn off rdfs inferencing
conforms, report_graph, report_text = v.run()
expanded_graph = v.target_graph

print(expanded_graph.serialize(format="ttl").decode("utf-8"))
```

```
@prefix dct: <http://purl.org/dc/terms/> .
@prefix ex: <http://example.com/ns#> .
@prefix ns1: <http://www.w3.org/ns/dcat#> .
@prefix ns2: <http://xmlns.com/foaf/0.1/> .
@prefix ns3: <http://www.w3.org/ns/locn#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<https://data.gov.au/dataset/67ca5de1-8774-4678-9d1b-8b1cb70ab33c> a ns1:Dataset ;
    ex:area2 "a descriptiono of this dataset" ;
    dct:description "a descriptiono of this dataset" ;
    dct:identifier "67ca5de1-8774-4678-9d1b-8b1cb70ab33c" ;
    dct:issued "2016-03-23T05:08:17.991412"^^xsd:dateTime ;
    dct:language "eng" ;
    dct:modified "2019-11-19T23:18:49.871451"^^xsd:dateTime ;
    dct:publisher <https://data.gov.au/organization/69f37b4c-bdf0-4c85-bd56-
 ↪82fa6d6b087a> ;
    dct:spatial [ a dct:Location ;
             ns3:geometry "POLYGON ((110.0012 -10.0012, 115.0080 -10.0012, 155.0080 -
 ↪45.0036, 110.0012 -45.0036, 110.0012 -10.0012))"^^<http://www.opengis.net/ont/
 ↪geosparql#wktLiteral>,
                 "{\"type\": \"Polygon\", \"coordinates\": [[[110.0012, -10.00117],␣
 ↪[115.008, -10.00117], [155.008, -45.00362], [110.0012, -45.00362], [110.0012, -10.
 ↪00117]]]}"^^<https://www.iana.org/assignments/media-types/application/vnd.geo+json>␣
 ↪] ;
    dct:title "Dynamic Land Cover Dataset" ;
    ns1:distribution <https://data.gov.au/dataset/67ca5de1-8774-4678-9d1b-
 ↪8b1cb70ab33c/resource/1f8174f8-573e-43f2-b110-3d1a13c380e8> ;
    ns1:keyword "Australia",
        "Cooper subregion",
        "LAND-Cover",
        "Maranoa-Balonne-Condamine subregion",
        "biota",
        "environment",
        "planningCadastre" .

<https://data.gov.au/dataset/67ca5de1-8774-4678-9d1b-8b1cb70ab33c/resource/1f8174f8-
 ↪573e-43f2-b110-3d1a13c380e8> a ns1:Distribution ;
```

(continues on next page)

(continued from previous page)

```
    dct:description "Data File" ;
    dct:format "ZIP" ;
    dct:title "Dynamic Land Cover Dataset" ;
    ns1:accessURL <https://datagovau.s3.amazonaws.com/bioregionalassessments/BA_ALL/
 ↪ALL/DATA/Geography/LandCoverNDLC/1556b944-731c-4b7f-a03e-14577c7e68db.zip> ;
    ns1:byteSize 186838338.0 .

<https://data.gov.au/organization/69f37b4c-bdf0-4c85-bd56-82fa6d6b087a> a␣
 ↪ns2:Organization ;
    ns2:name "Bioregional Assessment Program" .
```

26.5.10 Georeference KG completion example

About

This is a simple test notebook to explore approachs to associating geometries in the OIH graph with named ocean regions. From Marine Regions (https://www.marineregions.org/) we downloaded the IHO Sea Areas dataset. This is a shapefile that is converted to WKT and loaded into a geopandas dataframe.

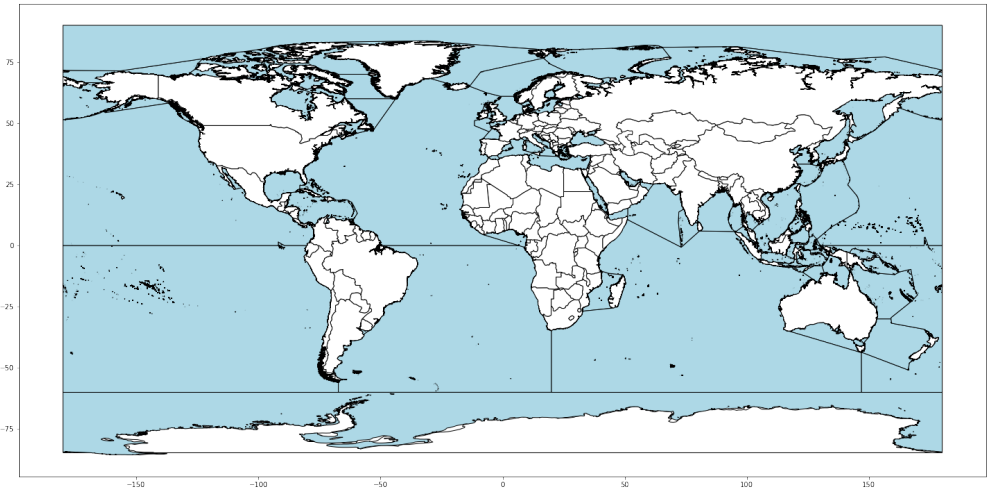
We then create a few points to compare against this. In this test we are only comparing points to the polygons. So this is what is called a "point in polygon" test. Later revisions to this could do polygon intersection or other tests. This is only a proof of concept notebook.

We then generate a new dataframe based on the matches. These matches could then be fed back into the graph as a sort of "Knowledge Graph Completion" workflow. Alternatively we can do gepsparql calls based on the sea WKT strings and do the search in the geosparql aware triple store.

References

- https://geopandas.org/docs/user_guide/set_operations.html
- https://geopandas.org/docs/user_guide/geocoding.html
- https://medium.com/analytics-vidhya/point-in-polygon-analysis-using-python-geopandas-27ea67888bff

Process



Conclusion

The generated dataframe holds the matches of the test Lat Long pairs to the named seas from the reference shape file. These results couild be fed back into the graph as keywords or items from a known list of terms for more explicate relation mapping.

Specifically, someting like https://schema.org/DefinedTerm where the property https://schema.org/DefinedTermSet would point back to the Marine Regions source documents and URL. Similarly these resources could be connected up to WikiData in a similar manner.

```
        {
                "@type": "DefinedTermSet",
                "@id": "http://geonetwork.vliz.be/geonetwork/srv/eng/catalog.search#/
↪metadata/f4cfa278-730f-4646-b6cc-a3dceaa3a1e5",
                "name": "IHO Sea Areas"
        },
        {
                "@type": "DefinedTerm",
                "name": "Bay of Bengal",
                "description": "IHO Sea Area Bay of Bengal",
                "inDefinedTermSet": "http://geonetwork.vliz.be/geonetwork/srv/eng/
↪catalog.search#/metadata/f4cfa278-730f-4646-b6cc-a3dceaa3a1e5"
        },
```

For reference the WikiData resource is: https://www.wikidata.org/wiki/Q38684 which is an instance of “body of water”. So leveraging this type and the IHO names should allow relatively reliable link detection. Leveraging the top level Thing class in schema.org we would be looking at a simple

```
    "sameAs": "https://www.wikidata.org/wiki/Q38684",
```

in the DefinedTerm type.

In the final listing below the "id" is the just the random string I associated with the test lat longs. I used a simple online map to just pick some random locations and gave them names. The "region" cames from the offical Marine Regions file.

Geospatial KG Completion

About

This is a simple test notebook to explore approachs to associating geometries in the OIH graph with named ocean regions. From Marine Regions (https://www.marineregions.org/) we downloaded the IHO Sea Areas dataset. This is a shapefile that is converted to WKT and loaded into a geopandas dataframe.

We then create a few points to compare against this. In this test we are only comparing points to the polygons. So this is what is called a "point in polygon" test. Later revisions to this could do polygon intersection or other tests. This is only a proof of concept notebook.

We then generate a new dataframe based on the matches. These matches could then be fed back into the graph as a sort of "Knowledge Graph Completion" workflow. Alternatively we can do gepsparql calls based on the sea WKT strings and do the search in the geosparql aware triple store.

References

- https://geopandas.org/docs/user_guide/set_operations.html
- https://geopandas.org/docs/user_guide/geocoding.html
- https://medium.com/analytics-vidhya/point-in-polygon-analysis-using-python-geopandas-27ea67888bff

```
#@title
# !apt-get install libproj-dev proj-data proj-bin
# !apt-get install libgeos-dev
!pip install -q cython
!pip install -q cartopy
!pip install -q SPARQLWrapper
!pip install -q rdflib
!pip install -q geopandas
!pip install -q contextily==1.0rc2
!pip install -q rtree
!pip install -q pygeos
```

```
from SPARQLWrapper import SPARQLWrapper, JSON
import pandas as pd
import numpy as np
import json
import geopandas
import matplotlib.pyplot as plt
import shapely

#dbsparql = "http://dbpedia.org/sparql"
# ufokn = "http://graph.collaborium.io/blazegraph/namespace/oihdev/sparql"
```

```
# Point in Polygon function from the cited Vidhya reference.  A few small changes
# to align with our dataframes here

def get_pip (gdf, regions):
    r_list = list(regions.NAME)
    #create empty dataframe
    df = pd.DataFrame().reindex_like(gdf).dropna()
    for r in r_list:
        #get geometry for specific region
        pol = (regions.loc[regions.NAME==r])
        pol.reset_index(drop = True, inplace = True)
        #identify those records from gdf that are intersecting with the region polygon
        pip_mask = gdf.within(pol.loc[0, 'WKT'])
        #filter gdf to keep only the intersecting records
        pip_data = gdf.loc[pip_mask].copy()
        #create a new column and assign the region name as the value
        pip_data['region']= r
        #append region data to empty dataframe
        df = df.append(pip_data)
    #checking there are no more than one region assigned to an event
    print('Original dataframe count=',len(gdf),'\nNew dataframe count=', len(df))
    if df.loc[df.id.duplicated() == True].shape[0] > 0:
        print("There are id's with more than one region")
    #checking all events have a region
    elif gdf.loc[~gdf.id.isin(df.id)].shape[0] > 0:
        print("There are id's without an assigned region")
    else:
        print("No discrepancies in results!")
    df.reset_index(inplace=True, drop=True)
    df = df.drop(columns='geometry')
    return df
```

```
# load CSV into pandas that holds the WKT strings for the world seas dataset
df = pd.read_csv('./World_Seas_IHO_v3/out.wkt/World_Seas_IHO_v3.csv')
# df.head(2)
```

```
# convert the WKT strings to WKT geometry
from shapely import wkt
df['WKT'] = df['WKT'].apply(wkt.loads)
```

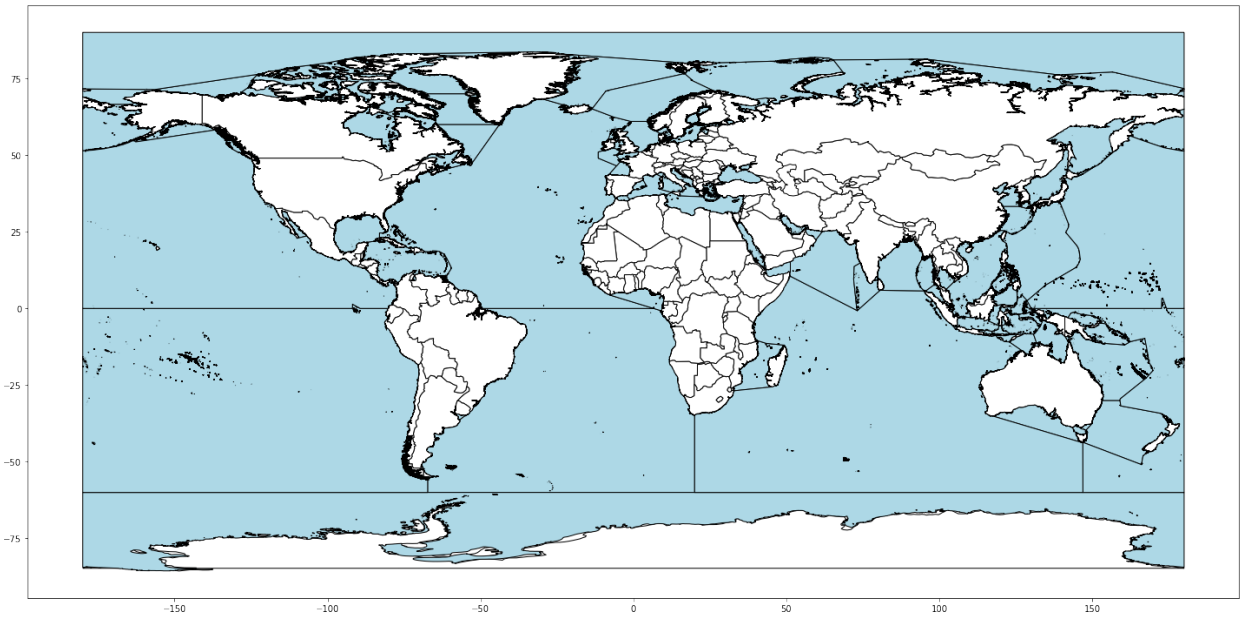
```
# Load and plot the ocean regions
import contextily as ctx

# Make geopandas from df
gdf = geopandas.GeoDataFrame(df, geometry='WKT', crs={"init": "epsg:3857"}) #␣
 ↪contextily requires 3875?
# gdf.head(2)
```

```
# Make geopandas from GeoJSON of the world for plot
import matplotlib.pyplot as plt

url = "https://raw.githubusercontent.com/johan/world.geo.json/master/countries.geo.
 ↪json"
dftx = geopandas.read_file(url)

# plot
```

(continues on next page)

(continued from previous page)

```
ax = dftx.plot(color='white', edgecolor='black', figsize=(25,15))
gdf.plot(ax=ax, edgecolor='black', color='lightblue')
plt.savefig('map.png')
```




```
# Make test datarame with some lat long pairs to test
# Later these will come from SPARQL calls to the OIH graph

df = pd.DataFrame({ 'Latitude': [38, -1,  -37,  -22,14],
                    'Longitude': [-146, 0, 129,  37,85],
                    'id':["Off US West Coast","Atlantic","South of Australia", "Africa
 ↪","Near India"]})

testdf = geopandas.GeoDataFrame(df, geometry=geopandas.points_from_xy(df.Longitude,␣
 ↪df.Latitude))
```

```
# Call the function and do pip calculations
eq_df = get_pip(testdf, gdf)
```

```
Original dataframe count= 5
New dataframe count= 5
No discrepancies in results!
```

Conclusion

The generated dataframe holds the matches of the test Lat Long pairs to the named seas from the reference shape file. These results couild be fed back into the graph as keywords or items from a known list of terms for more explicate relation mapping.

Specifically, someting like https://schema.org/DefinedTerm where the property https://schema.org/DefinedTermSet would point back to the Marine Regions source documents and URL. Similarly these resources could be connected up to WikiData in a similar manner.

```
        {
                "@type": "DefinedTermSet",
                "@id": "http://geonetwork.vliz.be/geonetwork/srv/eng/catalog.search#/
↪metadata/f4cfa278-730f-4646-b6cc-a3dceaa3a1e5",
                "name": "IHO Sea Areas"
        },
        {
                "@type": "DefinedTerm",
                "name": "Bay of Bengal",
                "description": "IHO Sea Area Bay of Bengal",
                "inDefinedTermSet": "http://geonetwork.vliz.be/geonetwork/srv/eng/
↪catalog.search#/metadata/f4cfa278-730f-4646-b6cc-a3dceaa3a1e5"
        },
```

For reference the WikiData resource is: https://www.wikidata.org/wiki/Q38684 which is an instance of "body of water". So leveraging this type and the IHO names should allow relatively reliable link detection. Leveraging the top level Thing class in schema.org we would be looking at a simple

```
    "sameAs": "https://www.wikidata.org/wiki/Q38684",
```

in the DefinedTerm type.

In the final listing below the "id" is the just the random string I associated with the test lat longs. I used a simple online map to just pick some random locations and gave them names. The "region" cames from the offical Marine Regions file.

```
eq_df.head()
```

```
   Latitude  Longitude                  id                   region
0     -37.0      129.0  South of Australia  Great Australian Bight
1     -22.0       37.0              Africa      Mozambique Channel
2      -1.0        0.0            Atlantic    South Atlantic Ocean
3      14.0       85.0          Near India           Bay of Bengal
4      38.0     -146.0   Off US West Coast     North Pacific Ocean
```

Part V

Validation

CHAPTER

TWENTYSEVEN

VALIDATION

27.1 About

This section contains some initial work on developing some validation approaches for OIH. The focus initially is not on validating approaches with the full publishing guidance. Rather the focuses is on the the "info hub" as a search application and develops validation to support that.

Initial approach:

- Develop a base SHACL document that assess a data graph based on elements needed to support search
- SHACL
- Leverage SHACL Playground

To support this will need an initial data graph to work with. The type is not important. All types will need to satisfy the search needs.

Examples of these needs include:

- Have an @id
- Have a name
- Have a description
- Have a Distribution and contentURL
- Reference authority

27.2 Implementation

Work this implementation can be found in the notebooks section *pySHACL testing*.

Part VI

Interfaces

CHAPTER

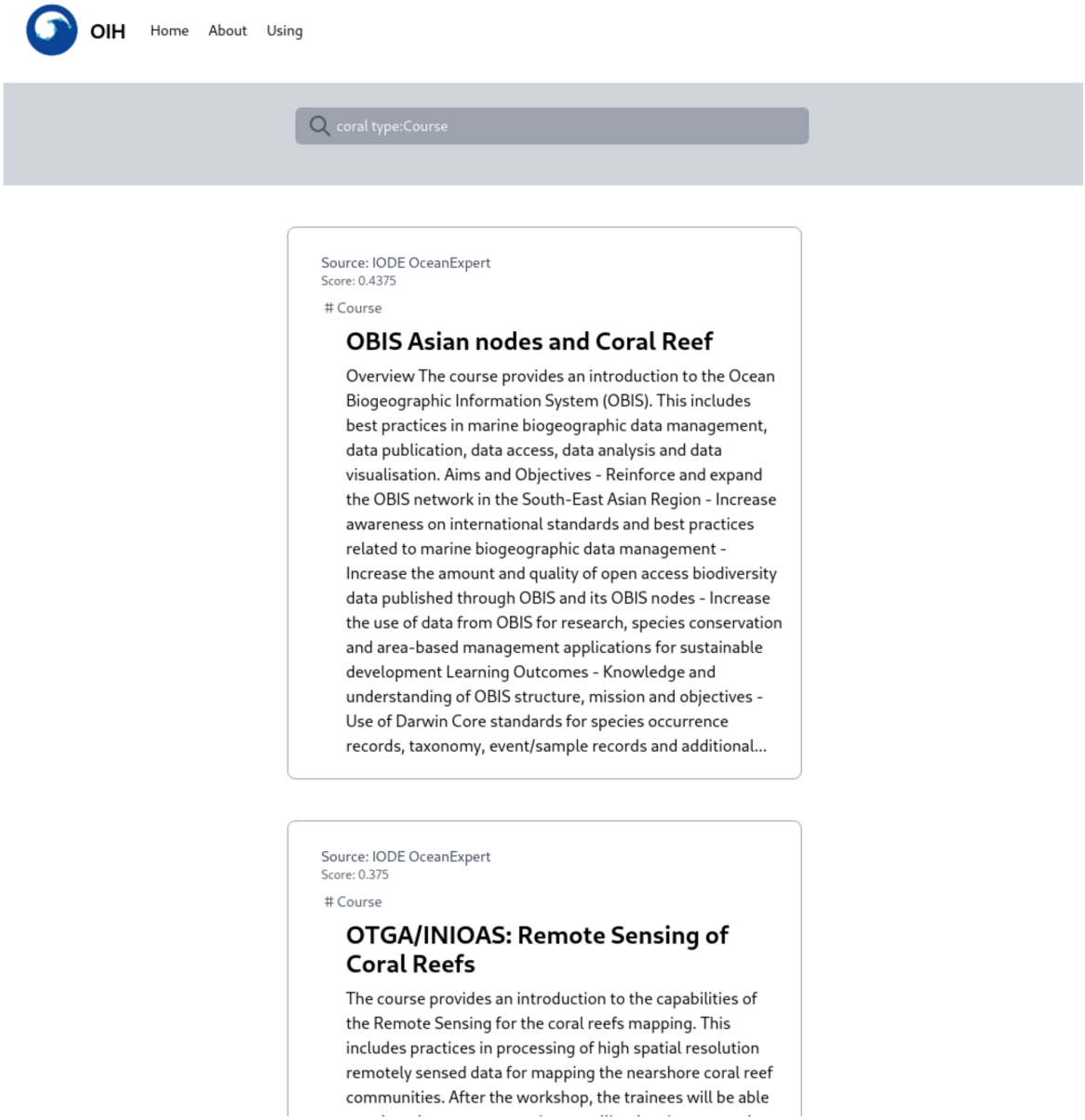
TWENTYEIGHT

USERS

28.1 About

Google dataset search

The reference client used for development is currently hosted at oceans.collaborium.io. This is a fully client side Javascript based client to the OIH index. All the code is hosted at https://github.com/iodepo/odis-arch/tree/master/schema/client/referenceclient/website


OIH
Home
About
Using
coral type:Course
Source: IODE OceanExpert
Score: 0.4375
Course
OBIS Asian nodes and Coral Reef
Overview The course provides an introduction to the Ocean Biogeographic Information System (OBIS). This includes best practices in marine biogeographic data management, data publication, data access, data analysis and data visualisation. Aims and Objectives - Reinforce and expand the OBIS network in the South-East Asian Region - Increase awareness on international standards and best practices related to marine biogeographic data management - Increase the amount and quality of open access biodiversity data published through OBIS and its OBIS nodes - Increase the use of data from OBIS for research, species conservation and area-based management applications for sustainable development Learning Outcomes - Knowledge and understanding of OBIS structure, mission and objectives - Use of Darwin Core standards for species occurrence records, taxonomy, event/sample records and additional...
Source: IODE OceanExpert
Score: 0.375
Course
OTGA/INIOAS: Remote Sensing of Coral Reefs
The course provides an introduction to the capabilities of the Remote Sensing for the coral reefs mapping. This includes practices in processing of high spatial resolution remotely sensed data for mapping the nearshore coral reef communities. After the workshop, the trainees will be able

CHAPTER

TWENTYNINE

QUERYING SPARQL

29.1 About

This page will hold some information about the SPARQL queries we use and how they connect with some of the profile guidance in this document. We will show how this relates to and depends on the Gleaner prov as well as the Authoritative Reference elements of the patterns. It is expected that the Gleaner prov will be present, though this can be made optional in case other indexing systems are used that do not provide this prov shape. The SPARQL will be looking for both Gleaner prov and the Authroitative Reference elements.

This will be different for different patterns. For example, it might relate to the publisher provider elements for Creativeworks, but to the identity element for People and Organizations.

Also here will be a SHACL shape to help validate a record as fit for use with the query. It will provide guidance on what is optional and and is required.

```
prefix prov: <http://www.w3.org/ns/prov#>
PREFIX con: <http://www.ontotext.com/connectors/lucene#>
PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
PREFIX con-inst: <http://www.ontotext.com/connectors/lucene/instance#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <https://schema.org/>
PREFIX schemaold: <http://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#
SELECT DISTINCT ?g  ?s  ?wat ?orgname ?domain ?type ?score ?name ?url ?lit ?
 ↪description ?headline
WHERE

   ?lit bds:search "coral" .
   ?lit bds:matchAllTerms "false" .
   ?lit bds:relevance ?score .
   graph ?g {
    ?s ?p ?lit .
    ?s rdf:type ?type .
    OPTIONAL { ?s schema:name ?name .   }
    OPTIONAL { ?s schema:headline ?headline .   }
    OPTIONAL { ?s schema:url ?url .   }
    OPTIONAL { ?s schema:description ?description .    }
  }
   ?sp prov:generated ?g  .
   ?sp prov:used ?used .
   ?used prov:hadMember ?hm .
   ?hm prov:wasAttributedTo ?wat .
   ?wat rdf:name ?orgname .
   ?wat rdfs:seeAlso ?domai
```

(continues on next page)

(continued from previous page)

```
}
ORDER BY DESC(?score)
LIMIT 30
OFFSET 0
```

Part VII

Appendix

CHAPTER

THIRTY

APPENDIX

30.1 About

A collection of items related to the OIH development. Mostly related to examples around representing concepts in the graph such as date and time, language etc.

As these develop they may be moved into other sections of the book.

30.2 Known Issues

30.2.1 About

This document will collect some of the various issues we have encounter in publishing the JSON-LD documents.

control characters in URL string for sitemap or in the JSON-LD documents

Make sure there are no control characters such as new line, carriage returns, tabs or others in the document. These can be problematic both for processing and display.

context is a map (changed from 1.0)

Be sure to use a context style like:

```
    "@context": {
        "@vocab": "https://schema.org/"
    },
```

The context section must be a map starting in JSON-LD 1.1

data graphs need @id

Be sure to include an @id in your graph that points to the identifier or the web address of the resource providing the metadata. This is not the material the metadata is about, but rather the metadata record itself.

string literals must be valid

The string literals must be sure to not have quotation marks or other invalid characters without escaping or encoding them.

30.3 References

A broad collection of references.

30.3.1 General

Science on Schema

BioSchemas

Ocean Best Practices on Schema

PID policy for European Open Science Cloud

DCAT Schema.org mappings

DCAT US Data.gov reference

FAIR Semantics

30.3.2 Developer References

Schema.org releases

Schema.org RDF graph (turtle format)

JSON-LD Playground

Google Developers Search Gruid

Google Developers Fact Check

Structured Data Testing Tool

Rich Results Testing Tool

JSON-LD

Ruby JSON-LD

schema.org Java

Perl classes for schema.org markup

30.3.3 Content Management and Web Server support

Drupal Support

Wordpress Claim Review

30.3.4 Organizations

Google Open Source

DataCommons & DataCommons REST

30.3.5 Indexers

Gleaner BMUSE

30.3.6 Graph tools

Wikipedia SPARQL implementation list

RDFlib

Any23

rdfjs

shemaram

shemarama demo

validatingrdf

Structured Data Linter

30.3.7 Blogs and Press Releases

Yandex: What is Scheme.org

Bing: Fact Check Label

Bing: Contextual Awareness

Facebook: Marketing API

Facebook: fact checking

Amazon: Alex skills

Google Developers mail invoice

30.3.8 Not Categorized Yet

Lighthouse Plugins

Lighthouse

Science on Schema

BioSchemas

CodeMeta

Linter Structured Data

JSON-LD Playground

JSON-LD.org

SHACL playground

Google Structured Data testing tool (

Google Dataset for developers

Press article

Rich results

SchemaApp.com

Yandex

Schema dev

Chromeextension

Google Rich Results

Datashapes

ACL Web Alexa Meaning Representation Language

hash.aio Volcano schema

Yoast Structured Data Guide

Schema App

Springer: Scigraph

iPhylo Biodiversity KG

ozymandias

RDA group meeting notes

RDA Plenary meeting

30.4 Registries

30.4.1 Documents and Datasets

DOI

A not-for-profit membership organization that is the governance and management body for the federation of Registration Agencies providing Digital Object Identifier (DOI) services and registration, and is the registration authority for the ISO standard (ISO 26324) for the DOI system. The DOI system provides a technical and social infrastructure for the registration and use of persistent interoperable identifiers, called DOIs, for use on digital networks.

Datacite

Locate, identify, and cite research data with the leading global provider of DOIs for research data.

Archival Resource Key or ARK: and N2T ARKs and Names to Thinkgs

30.4.2 People

Orcid

ORCID's mission is to enable transparent and trustworthy connections between researchers, their contributions, and their affiliations by providing a unique, persistent identifier for individuals to use as they engage in research, scholarship, and innovation activities.

30.4.3 Organizations

re3data

A registry of research data repositories

ROR

ROR is a community-led project to develop an open, sustainable, usable, and unique identifier for every research organization in the world.

Grid

GRID is a free and openly available global database of research-related organisations, cataloging research-related organisations and providing each with a unique and persistent identifier. With GRID you have over 99,609 carefully curated records at hand, enabling you to identify and distinguish research-related institutions worldwide.

30.4.4 Physical Samples

IGSN

The objective of the IGSN e.V. is to implement and promote standard methods for identifying, citing, and locating physical samples with confidence by operating an international IGSN registration service.

30.5 Controlled Vocabularies

30.5.1 About

See also: ODIS-ARCH Vocabularies

A list of possible controlled vocabularies to use in the schema.org documents. Many such resources can be found by searching at BARTOC.org or the UNESCO Thesaurus.

Note, at present this is an exploration and there is not yet a recommendation for use in OIH.

30.5.2 List

Essential Climate Variables

Vocabulary based on DFG'S Classification of Subject Area, Review Board, Research Area and Scientific Discipline (2016 - 2019)

Nature Subjects Ontology