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

IODEPO

Sep 01, 2021

CONTENTS

I	Introduction	5
1	Structured Data on the Web	7
1.1	About	7
1.2	Web architecture approach	7
1.3	Terminology	8
1.4	Intellectual Merit	8
1.5	Broader Impacts	8
2	Personas	9
2.1	About	9
2.2	Persona: Publisher	10
2.3	Persona: Aggregator	10
2.4	Persona: User	10
3	Publisher	11
3.1	About	11
3.2	Basics	13
3.3	Full Workflow	15
3.4	Existing support in software	16
4	JSON-LD Foundation	17
4.1	Introduction	17
4.2	The Context	18
4.3	Graph	19
II	Profiles	23
5	Thematic Patterns	25
5.1	Introduction	25
5.2	Thematic Profiles	25
6	Experts and Institutions	27
6.1	About	27
6.2	Example: Person Graph	27
6.3	Example: Institution Graph	30
6.4	References	32
7	Documents	33
7.1	About	33
7.2	Creative works (documents)	33

8	Spatial Maps	37
8.1	About	37
9	Projects	41
9.1	About	41
9.2	Research Project	41
9.3	Full Research Project	42
10	Training	45
10.1	About	45
10.2	Simple Course	45
10.3	Detailed Course	46
10.4	References	47
11	Vessels	49
11.1	About	49
11.2	References	51
12	Spatial Geometry	53
12.1	About	53
12.2	Simple GeoSPARQL WKT	53
12.3	Classic Schema.org	54
12.4	Option review, SOS Issue 105	54
12.5	References	56
13	Services	57
13.1	About	57
13.2	References	58
14	Keywords and Defined Terms	59
14.1	About	59
14.2	Keywords	59
14.3	Defined Terms	61
14.4	References	62
15	Languages	63
15.1	About	63
16	Linking to documents and resources	65
16.1	Organization link options	65
16.2	Sustainable Development Goals	68
16.3	Refs	70
17	Source and Prov Approaches	71
17.1	About	71
17.2	References	77
III	Aggregation	79
18	Aggregator	81
18.1	Intoduction	81
18.2	ODIS Catalog as Index Source	83
19	Indexing with Gleaner	85
19.1	Gleaner (app)	85

19.2	References	93
20	Gleaner CLI Docker	95
20.1	About	95
20.2	Prerequisites	95
20.3	Steps	95
20.4	Working with results	97
20.5	Loading to the triplestore	98
20.6	Conclusion	101
21	Indexing Services	103
22	Data Services	107
22.1	Gleaner Data Services (DS)	107
23	Interfaces	115
23.1	About	115
23.2	Gleaner Web UI (WUI)	115
24	Graph First Approach	117
24.1	About	117
24.2	Graph Only	117
24.3	Item Catalogue Page	118
24.4	Publishing and referencing	118
24.5	Testing	118
25	Prov	121
25.1	About	121
25.2	Gleaner Prov	121
25.3	Nano Prov	123
25.4	Refs	125
26	Alternatives	127
26.1	Options	127
IV	Tooling	129
27	Tooling	131
27.1	About	131
27.2	On-line tooling	131
27.3	Dev	131
27.4	OpenRefine	132
27.5	OIH Notebooks	137
V	Validation	227
28	Validation	229
28.1	About	229
28.2	Implementation	230
VI	Interfaces	231
29	Users	233

29.1 About	233
30 SPARQL	235
30.1 Introdocation	235
30.2 SPARQL	235
30.3 Load a Graph	235
31 Let's define our first query	237
32 Run the query	239
33 Nicer formatting	241
34 KGLab	243
35 PREFIX	245
36 Location Counts	247
37 FILTER	249
38 PERU	251
39 Dates	253
40 A bit of Pandas	255
41 More on Query	257
42 References	259
43 OIH SPARQL	261
43.1 About	261
43.2 Lines 11-13	262
43.3 Lines 17-20	262
43.4 Lines 22-27	262
43.5 Lines 29-31	262
44 APIs	263
44.1 About	263
44.2 SPARQL HTTP Protocol	263
VII Appendix	265
45 Appendix	267
45.1 About	267
45.2 Known Issues	267
45.3 References	268
45.4 Registries	271
45.5 Controlled Vocabularies	272
Bibliography	273

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.

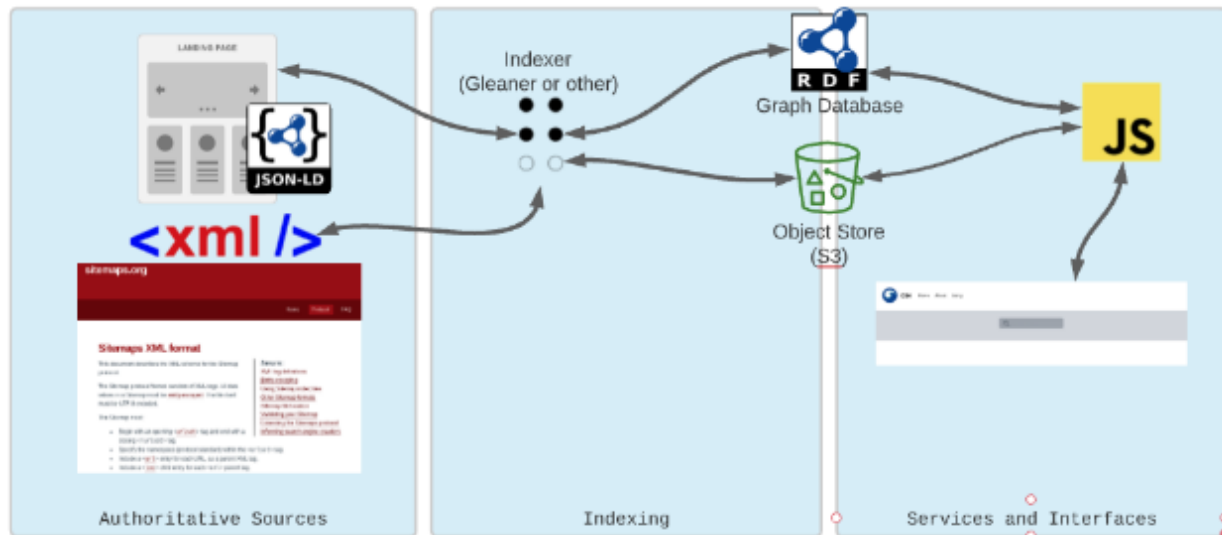
The ODIS-architecture development is being supported by the Ocean InfoHub Project, and it has been tested initially on IOC and partner databases. However, the system and standards are open for any institution or initiative that is interested in accessing the global data ecosystem to adopt and implement.

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 implementing 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

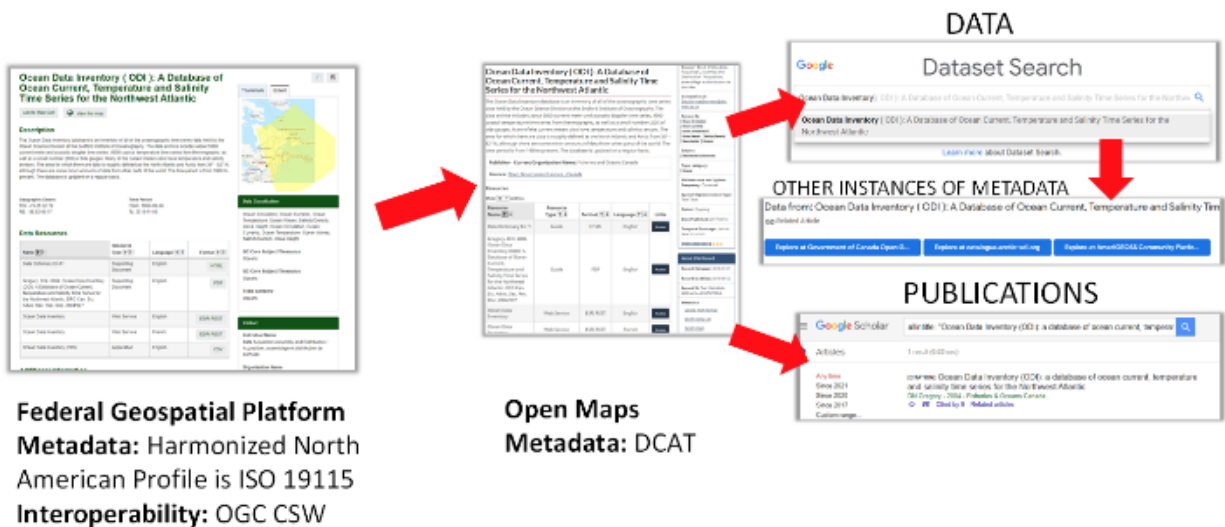
software that can generate the index.

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:

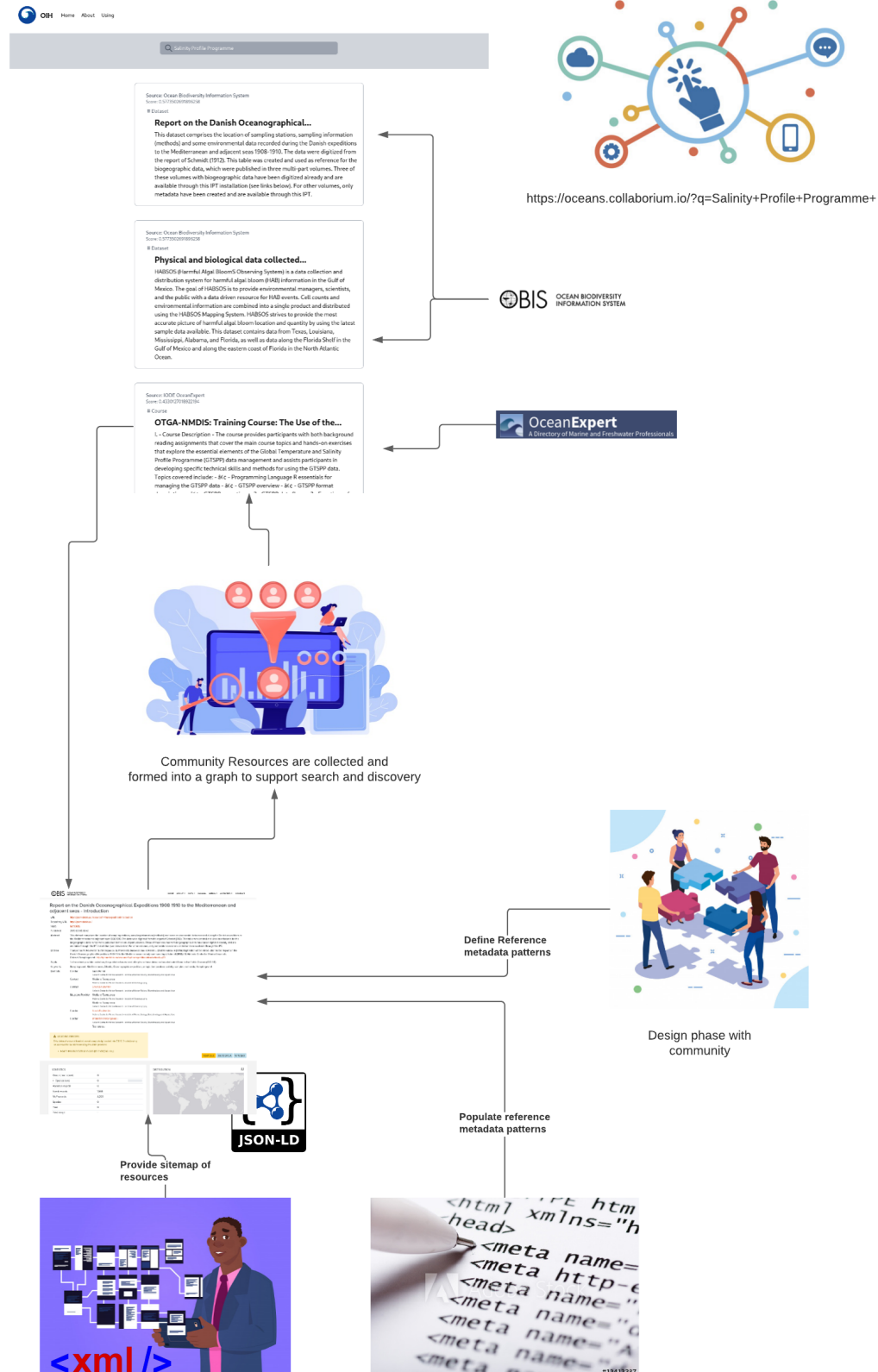
- The Federal Geospatial Platform is a intra-governmental data catalogue implementing the Harmonized North American Profile of ISO 19115 (HNAP), with content exposed externally via OGC CSW (Catalogue Services Web).
- This content is harvested by the public facing Open Maps platform, which includes a 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.

CONTENTS



Part I

Introduction

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](http://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 file 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 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](https://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](https://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 on the web patterns in the form of [Schema.org](https://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.

PERSONAS

2.1 About

During the design process of the Ocean InfoHub (OIH), many of the design approaches leverage three personas that help define the various archetypes of people who engage with OIH. It should not be assumed these scope all the potential persona or that a person or organization scope only one. It is quite possible to many. These are simply design approaches representing potential models or characters. They are tools used in the design process of OIH.



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 engaged authoring the JSON-LD documents and publishing them to the web. This persona is focused on describing and presenting structured data on the web to aid in the discovery and use the resources they manage. 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. Their goal is to efficiently and efficiently index the resources exposed by the Publisher persona and generate usable indexes. Further, they would work to exposed these indexes in a manner that is usable by the User persona. 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. They may also use query languages like SPARQL or other APIs or even directly work with the underlying data warehouse of collected data graphs.

User tools may be web sites or scientific notebooks. Some examples of these user experiences are described in the [User](#) section.

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.

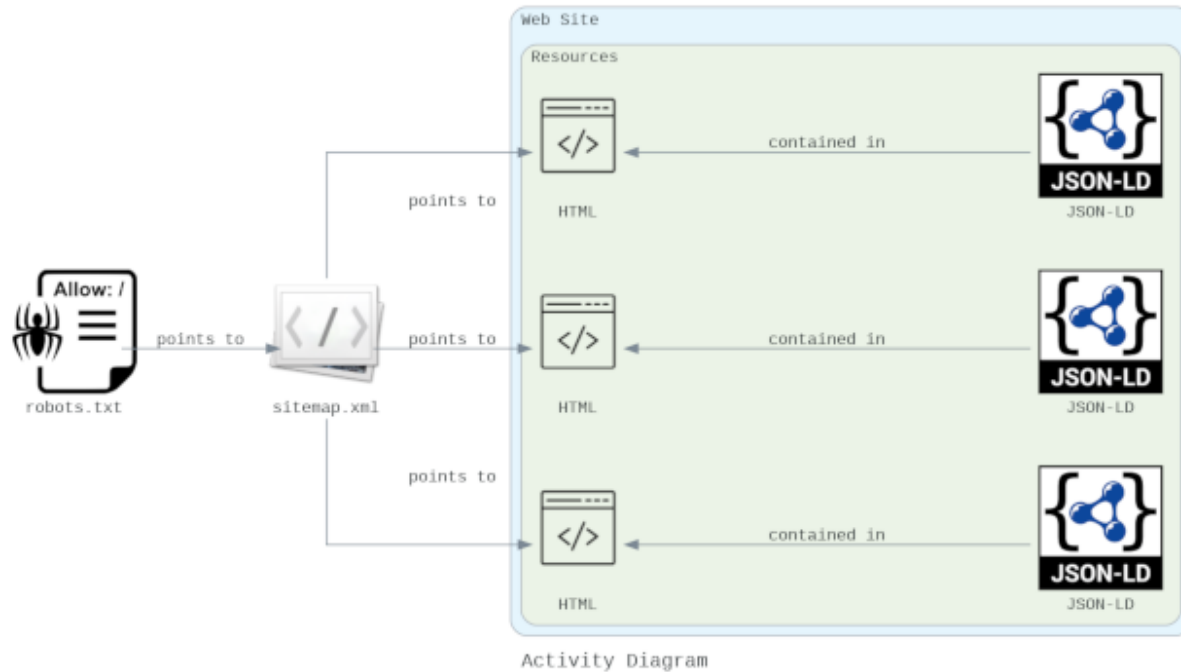
The results of the above can then be loaded into the online JSON-Editor at <https://json-editor.github.io/json-editor/>. (Ref: <https://github.com/json-editor/json-editor>)

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

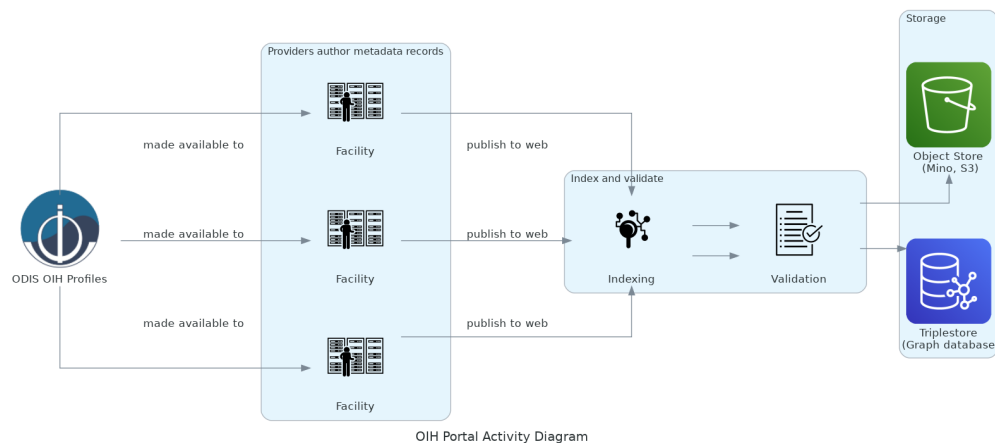
Providers will need to expose a set of resource landing pages using a sitemap.xml file. As noted above, providers can expose a sitemap file to just the target agent to avoid indexing test pages by commercial providers. You may wish to do this during testing or for other reasons. Otherwise, a sitemap.xml 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_docclouds_igsndatagraphs.xml</loc>
    <lastmod>2005-01-01</lastmod>
  </sitemap>
</sitemapindex>
```

3.3 Full Workflow



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 resources, 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 systems other web based data interfaces may already have support for the structured data on the web pattern and schema.org specifically. While it is beyond the scope of this project to detail each one, a few starting points for exploration are provided below for some of the more common ones.

- [Drupal](#)
- [CKAN](#)
- [DSpace](#)
- [DKAN](#)
- [ERDDAP](#) (native support)
- [OPeNDAP](#) (native support)
- [GeoNode](#)
 - [schema.org](#) issue ref

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.

```
1 {  
2   "@context": {  
3     "@vocab": "https://schema.org/"  
4   },  
5   "@type": "CreativeWork",  
6   "@id": "https://example.org/id/XYZ",  
7   "name": "Name or title of the document",  
8   "description": "Description of the creative work to aid in searching",  
9   "url": "https://www.sample-data-repository.org/creativework/report.pdf"  
10 }
```

```
<graphviz.dot.Digraph at 0x7f081c501190>
```

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.

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@type": "CreativeWork",
6   "@id": "https://example.org/id/XYZ",
7   "name": "Name or title of the document",
8   "description": "Description of the creative work to aid in searching",
9   "url": "https://www.sample-data-repository.org/creativework/report.pdf"
10 }
```

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

As just 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.

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/",
4     "geosparql": "http://www.opengis.net/ont/geosparql#"
5   },
6   "@id": "https://example.org/id/XYZ",
7   "@type": "Dataset",
8   "name": "Data set name",
9   "geosparql:hasGeometry": {
10     "@type": "http://www.opengis.net/ont/sf#Point",
11     "geosparql:asWKT": {
12       "@type": "http://www.opengis.net/ont/geosparql#wktLiteral",
13       "@value": "POINT(-76 -18)"
14     },
15     "geosparql:crs": {
16       "@id": "http://www.opengis.net/def/crs/OGC/1.3/CRS84"
17     }
18   }
19 }
20 }
```


4.3 Graph

The next section we will discuss is the graph part of the document seen in lines 5-9 below. This is where the properties and values of our resource are described.

```

1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@type": "CreativeWork",
6   "@id": "https://example.org/id/XYZ",
7   "name": "Name or title of the document",
8   "description": "Description of the creative work to aid in searching",
9   "url": "https://www.sample-data-repository.org/creativework/report.pdf"
10 }

```

First though, let's visit a couple special properties in our document.

4.3.1 Node identifiers (@id)

```

1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@type": "CreativeWork",
6   "@id": "https://example.org/id/XYZ",
7   "name": "Name or title of the document",
8   "description": "Description of the creative work to aid in searching",
9   "url": "https://www.sample-data-repository.org/creativework/report.pdf"
10 }

```

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 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 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 allow 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 you use content negotiation to select between HTML and JSON-LD representations of the record.

4.3.2 Type identifiers (@type)

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@type": "CreativeWork",
6   "@id": "https://example.org/id/XYZ",
7   "name": "Name or title of the document",
8   "description": "Description of the creative work to aid in searching",
9   "url": "https://www.sample-data-repository.org/creativework/report.pdf"
10 }
```

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

Note: In [Schema.org](https://schema.org) and in most vocabularies, types will be named with a capital 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 return to look at the other properties for our type.

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@type": "CreativeWork",
6   "@id": "https://example.org/id/XYZ",
7   "name": "Name or title of the document",
8   "description": "Description of the creative work to aid in searching",
9   "url": "https://www.sample-data-repository.org/creativework/report.pdf"
10 }
```

As noted, we are using [Schema.org](https://schema.org) type for OIH. In this case, as mentioned, this is type `CreativeWork`. So any of the properties seen at the [Schema.org](https://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*.

For the OIH implementation, we will use the following properties as core properties we want all OIH documents to have. These include:

name: The name of the document or item being described

description: A description of the document or item being described

url: A URL for the document or item being described.

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](http://schema.org/Thing) and [Datatype](http://schema.org/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 but also the DataTypes Text and URL, where URL is actually a sub-type of Text.

Part II

Profiles

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](https://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](#).

EXPERTS AND INSTITUTIONS

6.1 About

This thematic type provides a way to describe the experts and institutions. In this case the following definitions are used:

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 use for a person. We will break down some of the key properties used in this graph.

As Ocean InfoHub is leveraging [Schema.org](https://schema.org) we are using schema.org/Person for this type. Any of the properties of Person seen there are valid to use in such a record.

While publishers are free to use as many elements as they wish, our goal with this documentation is provide a simple example that address some of the search and discovery goals of OIH along with those properties most useful in the linking of resources between OIH participants.

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@id": "https://example.org/id/x",
6   "@type": "Person",
7   "name": "Jane Doe",
8   "jobTitle": "Professor",
9   "telephone": "(425) 123-4567",
10  "url": "http://www.janedoe.com",
11  "knowsAbout": [
12    {
13      "@type": "Text",
14      "description": "Invasive species in brackish water"
15    },
16    {
17      "@type": "URL",
18      "url": "https://www.wikidata.org/wiki/Q183368"
19    },
20    {
21      "@id": "https://example.org/id/course/x",
```

(continues on next page)

(continued from previous page)

```

22     "@type": "Course",
23     "description": "In this course ...",
24     "url": "URL to the course"
25   }
26 ],
27   "identifier": {
28     "@id": "https://orcid.org/0000-0002-2257-9127",
29     "@type": "PropertyValue",
30     "propertyID": "https://registry.identifiers.org/registry/orcid",
31     "url": "https://orcid.org/0000-0002-2257-9127",
32     "description": "Optional description of this record..."
33   },
34   "nationality": [
35     {
36       "@type": "Country",
37       "name": "Fiji"
38     },
39     {
40       "@type": "DefinedTerm",
41       "url": "https://unece.org/trade/cefact/unlocode-code-list-country-and-territory
↪",
42       "inDefinedTermSet": "UN/LOCODE Code List by Country and Territory",
43       "name": "Fiji",
44       "termCode": "FJ"
45     }
46   ],
47   "knowsLanguage" :{
48     "@type": "Language",
49     "name": "Spanish",
50     "alternateName": "es"
51   }
52 }

```

```
<graphviz.dot.Digraph at 0x7f06f81cb6d0>
```

6.2.1 Details: Identifier

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 0x7f06e9d06a00>
```

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 0x7f06e9d0a5b0>
```

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"
  }
}
```

(continues on next page)

(continued from previous page)

```
}  
}
```

```
<graphviz.dot.Digraph at 0x7f06ea177850>
```

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 0x7f06e9d0af70>
```

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.

```
1 {  
2   "@context": {  
3     "@vocab": "https://schema.org/"  
4   },  
5   "@id": "https://index.example.org/id/org/x",  
6   "@type": "Organization",  
7   "address": {  
8     "@type": "PostalAddress",  
9     "addressLocality": "Paris, France",  
10    "postalCode": "F-75002",
```

(continues on next page)

(continued from previous page)

```

11     "streetAddress": "38 avenue de l'Opera"
12 },
13 "email": "secretariat(at)example.org",
14 "name": "Organization X",
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
34     ↪Development",
35     "propertyID": "https://registry.identifiers.org/registry/grid",
36     "url": "https://grid.ac/institutes/grid.475727.4"
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 of 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 0x7f06f81fb790>
```

6.3.2 Details: Identifier

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",
  "@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 0x7f06e9d06070>
```

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](#)

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](#)

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@type": "CreativeWork",
6   "@id": "https://example.org/id/XYZ",
7   "name": "Name or title of the document",
8   "description": "Description of the creative work to aid in searching",
9   "url": "https://www.sample-data-repository.org/creativework/report.pdf",
10  "contributor": {
11    "@type": "Organization",
12    "@id": "http://www.foo.org/orgID",
13    "legalName": "Some Institute"
14  },
15  "author": {
16    "@id": "https://www.sample-data-repository.org/person/51317",
17    "@type": "Person",
18    "name": "Dr Uta Passow",
19    "givenName": "Uta",
20    "familyName": "Passow",
21    "url": "https://www.sample-data-repository.org/person/51317"
22  },
23  "identifier": {
```

(continues on next page)

(continued from previous page)

```

24     "@id": "https://doi.org/10.5066/F7VX0DMQ",
25     "@type": "PropertyValue",
26     "propertyID": "https://registry.identifiers.org/registry/doi",
27     "value": "doi:10.5066/F7VX0DMQ",
28     "url": "https://doi.org/10.5066/F7VX0DMQ"
29 },
30 "keywords": {
31     "@type": "DefinedTerm",
32     "inDefinedTermSet": {
33         "@type": "DefinedTermSet",
34         "name": "Name of the set",
35         "description": "Description of the set",
36         "url": "url for the set"
37     },
38     "termCode": "A code that identifies this DefinedTerm within a DefinedTermSet"
39 },
40 "provider": {
41     "@id": "https://www.repositoryB.org",
42     "@type": "Organization",
43     "legalName": "Sample Data Repository Office",
44     "name": "SDRO",
45     "sameAs": "http://www.re3data.org/repository/r3dxxxxxxx",
46     "url": "https://www.sample-data-repository.org"
47 },
48 "license": "http://spdx.org/licenses/CC0-1.0",
49 "publisher": {
50     "@id": "https://www.publishingrus.org",
51     "@type": "Organization",
52     "legalName": "Some Institute"
53 }
54 }

```

```
<graphviz.dot.Digraph at 0x7f3ad45056d0>
```

7.2.1 Details: Identifier

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 0x7f3acd83bac0>
```

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/r3dxxxxxxxxx",
    "url": "https://www.sample-data-repository.org"
  },
  "publisher": {
    "@id": "https://www.publishingrus.org",
    "@type": "Organization",
    "legalName": "Some Institute"
  }
}
```

```
<graphviz.dot.Digraph at 0x7f3acd83fb20>
```

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 0x7f3acdca400>
```

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:CreativeWork](#) 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

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](https://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](https://schema.org/Map) type Map only offers one special property beyond the parent CreativeWork. That is a [mapType](https://schema.org/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](#)

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@type": "Map",
6   "@id": "https://example.org/id/XYZ",
7   "name": "Name or title of the document",
8   "description": "Description of the map to aid in searching",
9   "url": "https://www.sample-data-repository.org/creativework/map.pdf",
10  "identifier": {
11    "@id": "https://doi.org/10.5066/F7VX0DMQ",
12    "@type": "PropertyValue",
13    "propertyID": "https://registry.identifiers.org/registry/doi",
14    "value": "doi:10.5066/F7VX0DMQ",
15    "url": "https://doi.org/10.5066/F7VX0DMQ"
16  },
17  "keywords": [
18    {
19      "@id": "http://purl.org/dc/dcmitype/Image",
20      "@type": "DefinedTerm",
21      "inDefinedTermSet": "http://purl.org/dc/terms/DCMIType",
22      "termCode": "Image",
23      "name": "Image"
24    }
25  ]
26 }
```

(continues on next page)

(continued from previous page)

```

24     },
25     "Region X",
26     {
27         "@id": "https://www.wikidata.org/wiki/Q350134",
28         "@type": "URL",
29         "url": "https://www.wikidata.org/wiki/Q350134",
30         "name": "North Atlantic Ocean"
31     }
32 ]
33 }

```

<graphviz.dot.Digraph at 0x7fc5c45e26d0>

8.1.1 Details: Identifier

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 0x7fc5af9a8cd0>

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 0x7fc5af9a7cd0>
```

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:CreativeWork](#) 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

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 0x7f40b9b655e0>
```

9.2.1 Details: Identifier

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 0x7f40c074c220>
```

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.

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@type": "ResearchProject",
6   "@id": "https://example.org/id/resproj/X",
7   "legalName": "Example Data Repository",
8   "name": "ExDaRepo",
9   "url": "https://www.example-data-repository.org",
10  "description": "Repo description ... ",
11  "logo": {
12    "@type": "ImageObject",
13    "url": "https://www.example-data-repository.org/logo.jpg"
14  },
15  "identifier": {
16    "@id": "https://grid.ac/institutes/grid.475727.4",
17    "@type": "PropertyValue",
18    "description": "UN Department of Economic and Social Affairs Sustainable
↪Development",
19    "propertyID": "https://registry.identifiers.org/registry/grid",
20    "url": "https://grid.ac/institutes/grid.475727.4"
21  },
22  "contactPoint": {
23    "@id": "https://www.example-data-repository.org/about-us",
24    "@type": "ContactPoint",
25    "name": "Support",
```

(continues on next page)

(continued from previous page)

```

26     "email": "info@example-data-repository.org",
27     "url": "https://www.example-data-repository.org/about-us",
28     "contactType": "customer support"
29 },
30 "funder": {
31     "@type": "FundingAgency",
32     "@id": "https://dx.doi.org/10.13039/100000001",
33     "legalName": "National Science Foundation",
34     "alternateName": "NSF",
35     "url": "https://www.nsf.gov/"
36 },
37 "ethicsPolicy": {
38     "@type": "CreativeWork",
39     "@id": "https://example.org/id/XYZ",
40     "name": "Name or title of the document",
41     "description": "Description of the creative work ",
42     "url": "https://www.foo.org/creativework/ethicsPolicy.pdf"
43 },
44 "diversityPolicy": {
45     "@type": "CreativeWork",
46     "@id": "https://example.org/id/ABC",
47     "name": "Name or title of the document",
48     "description": "Description of the creative work",
49     "url": "https://www.foo.org/creativework/diversityPolicy.pdf"
50 },
51 "areaServed": [
52     {
53         "@type": "Place",
54         "geo": {
55             "@type": "GeoCoordinates",
56             "latitude": 39.3280,
57             "longitude": 120.1633
58         },
59         "description": "Description of the area served"
60     },
61     {
62         "@type": "Text",
63         "description": "Textual description of area served"
64     },
65     {
66         "@type": "AdministrativeArea",
67         "geo": {
68             "@type": "GeoCoordinates",
69             "latitude": 39.3280,
70             "longitude": 120.1633
71         },
72         "description": "Needs to be subset of Place, Review Place"
73     }
74 ],
75 "parentOrganization": {
76     "@type": "Organization",
77     "@id": "http://www.someinstitute.edu",
78     "legalName": "Some Institute",
79     "name": "SI",
80     "url": "http://www.someinstitute.edu",
81     "address": {
82         "@type": "PostalAddress",

```

(continues on next page)

(continued from previous page)

```
83     "streetAddress": "234 Main St.",
84     "addressLocality": "Anytown",
85     "addressRegion": "ST",
86     "postalCode": "12345",
87     "addressCountry": "USA"
88   }
89 }
90 }
```

```
<graphviz.dot.Digraph at 0x7f40c074ce80>
```

9.3.1 References

- <https://schema.org/Project>

TRAINING

10.1 About

A thematic type to describe potential training activities. In [Schema.org](https://schema.org) a Course is a subtype of [CreativeWork](https://schema.org/CreativeWork) and [LearningResource](https://schema.org/LearningResource).

As defined 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.

We can start by looking at a basic Course description.

10.2 Simple Course

A basic course might simply present the name and description of the course along with a few other key properties.

```
1 {  
2   "@context": {  
3     "@vocab": "https://schema.org/"  
4   },  
5   "@id": "https://example.org/id/course/2",  
6   "@type": "Course",  
7   "courseCode": "SD100",  
8   "name": "Structured Data",  
9   "description": "An introduction to authoring JSON-LD documents for OIH",  
10  "provider": {  
11    "@type": "Organization",  
12    "name": "Example University",  
13    "@id": "https://grid.ac/institutes/grid.475727.4",  
14    "description": "UN Department of Economic and Social Affairs Sustainable  
15    ↪Development"  
16  }  
}
```

<graphviz.dot.Digraph at 0x7f73f08986d0>

Here we can see the emphasized line 7 and lines 10-15 highlighting some unique types.

courseCode: The `courseCode` is used to provide the ID used by the provider for this course.

provider: The *provider* is the organization offering the course. This property is from the CreativeWork supertype. In this case the provider may be of type Organization or Person. For Ocean InfoHub these would be described in the *Experts and Institutions* section.

Note: In this case you can see we use a simple @id in the provider property. You can see this same @id used in the *Experts and Institutions* section. By doing this, we connect this provider to the Organization described by that document.

As such these will be connected in the graph. So there is no need to duplicate the information here. This is a common graph pattern that allows us to simply connect resources. If there was no existing Organization or Person resource you could simply create one here. However, you may also find it useful to create a given resource and link to it in the graph.

10.3 Detailed Course

There are a wide range of properties that can be used to describe a course. Many of these can be seen at the *Course* type as the properties from Course and properties from *LearningResource*.

We wont go into the details of each property here, but we will show a couple.

The example below present two.

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/",
4     "endDate": {
5       "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
6     },
7     "startDate": {
8       "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
9     }
10  },
11  "@id": "https://example.org/id/course/2",
12  "@type": "Course",
13  "courseCode": "SD100",
14  "name": "Structured Data",
15  "description": "An introduction to authoring JSON-LD documents for OIH",
16  "teaches": "JSON-LD",
17  "provider": {
18    "@type": "Organization",
19    "name": "Example University",
20    "@id": "https://grid.ac/institutes/grid.475727.4",
21    "description": "UN Department of Economic and Social Affairs Sustainable
↪Development"
22  },
23  "hasCourseInstance": [
24    {
25      "@type": "CourseInstance",
26      "courseMode": [
27        "MOOC1",
28        "online"
29      ],
30      "endDate": "2019-03-21",
31      "startDate": "2019-02-15",
32      "attendee": {
33        "@type": "Person",
```

(continues on next page)

(continued from previous page)

```

34         "name": "Jane Doe",
35         "jobTitle": "Professor",
36         "telephone": "(425) 123-4567",
37         "url": "http://www.janedoe.com",
38         "identifier": {
39             "@id": "ID_value_string",
40             "@type": "PropertyValue",
41             "propertyID": "This can be text or URL for an ID like ORCID",
42             "url": "https://foo.org/linkToPropertyIDPage",
43             "description": "Optional description of the ID"
44         }
45     },
46     {
47         "@type": "CourseInstance",
48         "courseMode": [
49             "MOOC2",
50             "online"
51         ],
52         "endDate": "2019-05-21",
53         "startDate": "2019-04-15"
54     }
55 ]
56 }
57 }
```

```
<graphviz.dot.Digraph at 0x7f73f087b9d0>
```

Line 16 shows the *teaches* property. It should be noted while this property can point to simple text, it is also possible to leverage *DefinedTerm*. This means a controlled vocabulary can be used to describe what the course teaches. Ocean InfoHub provides some more information and links to further information on defined term in the *Keywords and Defined Terms* section.

Lines 23-56 show using a *hasCourseInstance* property to show instances where this course is being taught. Also of note in this example are the lines 4-9 in the context where we can type the *endDate* and *startDate* as type *dateTime*. By doing this we must provide the dates in a format that is in line with the [XML Datatype] and in particular the *ISO 8601 Data and Time Formats*.

By doing this we can then later conduct searches on the graph that use date ranges to allow us to find courses, or any resources, that are being taught in a given time period.

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/>

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](https://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 as the WikiData entry for [Research Vessel, Q391022](https://www.wikidata.org/wiki/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.

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@id": "https://example.org/id/X",
6   "@type": "Vehicle",
7   "name": "JOIDES Resolution",
8   "identifier": {
9     "@id": "https://example.org/id/vessel/X",
10    "@type": "PropertyValue",
11    "propertyID": "https://en.wikipedia.org/wiki/IMO_number",
12    "url": "https://example.org/id/vessel/X",
13    "description": "Vessel ID "
14  },
15  "additionalProperty": {
16    "@id": "ID_value_string",
17    "@type": "PropertyValue",
18    "propertyID": "https://en.wikipedia.org/wiki/IMO_number",
19    "url": "https://foo.org/linkToPropertyIDPage",
20    "description": "Any additional properties for the vessel"
21  },
22  "subjectOf": {
23    "@type": "DataDownload",
24    "name": "external-metadata.xml",
25    "description": "Metadata describing the vessel",
26    "encodingFormat": [
27      "application/xml",
```

(continues on next page)

(continued from previous page)

```
28         "https://foo.org/ship01"
29     ],
30     "dateModified": "2019-06-12T14:44:15Z"
31 }
32 }
```

<graphviz.dot.Digraph at 0x7f29187c3640>

11.1.1 Details: Identifier

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 0x7f29180c1070>

11.1.2 Details: subjectOf

Like SOS, we are recommending the use of `subjectOf` to link a simple [Schema.org](https://schema.org/) type to a more detailed metadata description record. This allows us to use the easy discovery layer in [Schema.org](https://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 0x7f2918026cd0>

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

SPATIAL GEOMETRY

12.1 About

For spatial geometry Ocean InfoHub guidance will be to use the OGC [GeoSPARQL](#) vocabulary to express geometry using Well Known Text (WKT). The [schema.org](#) spatial types and properties are not well defined and difficult at times to reliably translate to geometries for use in more Open Geospatial Consortium (OGC) environments.

12.2 Simple GeoSPARQL WKT

The following is a simple example of how to embed a WKT string via GeoSPARQL into a JSON-LD record. Well Known Text (WKT) is a OGC standard referenced at: <https://www.ogc.org/standards/wkt-crs>. A more accessible description and set of examples can be found at Wikipedia: [Well-known text representation of geometry](#).

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/",
4     "geosparql": "http://www.opengis.net/ont/geosparql#"
5   },
6   "@id": "https://example.org/id/XYZ",
7   "@type": "Dataset",
8   "name": "Data set name",
9   "geosparql:hasGeometry": {
10     "@type": "http://www.opengis.net/ont/sf#Point",
11     "geosparql:asWKT": {
12       "@type": "http://www.opengis.net/ont/geosparql#wktLiteral",
13       "@value": "POINT(-76 -18)"
14     },
15     "geosparql:crs": {
16       "@id": "http://www.opengis.net/def/crs/OGC/1.3/CRS84"
17     }
18   }
19 }
```

Line 4 declare the GeoSPARQL prefix for the vocabulary that we will leverage in this document.

Lines 9-17 are the GeoSPARQL node and property definitions. In this case our type is a simple point geometry. We then go on to declare the asWKT with a type and value. The value is our actual WKT string for our geometry. We can further declare the coordinate reference system (CRS) of the geometry using the crs property.

```
<graphviz.dot.Digraph at 0x7f2f727db640>
```

12.3 Classic Schema.org

Ocean InfoHub only recommends the use of [Schema.org](#) spatial geometries in the case where a provider wishes to be properly indexed by Google and to have the spatial information used by Google for maps. Note, the lack of spatial information will not prevent Google from indexing your resources.

[Schema.org](#) spatial geometries are not well defined in comparison to OGC standards and recommendations. Also, converting from [Schema.org](#) spatial to geometries in WKT or GeoJSON can be problematic. There are inconsistencies with [Schema.org](#) guidance for textual geometry representation and that of Well Known Text (WKT).

That said, if you desire to leverage [Schema.org](#) geometries an example follows. This is a simple example of the existing [Schema.org](#) pattern for a lat long value. There is the pending [GeospatialGeometry](#) which is a type Intangible (and not Place referenced by spatialCoverage). This will be a subtype of [GeoShape](#).

```

1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@id": "https://example.org/id/XYZ",
6   "@type": "Dataset",
7   "name": "Data set name",
8   "spatialCoverage": {
9     "@type": "Place",
10    "geo": {
11      "@type": "GeoCoordinates",
12      "latitude": 39.3280,
13      "longitude": 120.1633
14    }
15  }
16 }
```

```
<graphviz.dot.Digraph at 0x7f2f804eda00>
```

12.4 Option review, SOS Issue 105

There are several approaches to expressing spatial geometries in JSON-LD. While Ocean InfoHub will recommend the use of GeoSPARQL, it is worth noting that there are alternative and solid cases for using them

One such case could be the case where your WKT geometry string are highly detailed and as a result quite long. These might result in both very large JSON-LD documents that are hard to read and maintain. It may also be that this imparts a performance penalty in your GeoSPARQL queries.

It may be that you simplify your WKT geometry strings to a more basic form. Then link out to the detailed geometry in a separate document. The simplified WKT (or [Schema.org](#) spatial) make the documents smaller and easier to read and could help query performance. The resource can then point to a dereferencable URL for the detailed geometry.

ref Selfie: When linking out to complex geometries we recommend following: <https://docs.ogc.org/per/20-067.html>

From the referenced SOS issue 105:

```

1 {
2   "@context": {
3     "@version": 1.1,
4     "geoblob": {
5       "@id": "http://example.com/vocab/json",
```

(continues on next page)

(continued from previous page)

```

6      "@type": "@json"
7    },
8    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
9    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
10   "xsd": "http://www.w3.org/2001/XMLSchema#",
11   "description": "http://igsn.org/core/v1/description",
12   "geosparql": "http://www.opengis.net/ont/geosparql#",
13   "schema": "https://schema.org/"
14 },
15 "@id": "https://samples.earth/id/do/bqs2dn2u6s73o70jdup0",
16 "@type": "http://igsn.org/core/v1/Sample",
17 "description": "A fake ID for testing",
18 "schema:subjectOf": [
19   {
20     "schema:url": "https://samples.earth/id/do/bqs2dn2u6s73o70jdup0.geojson",
21     "@type": "schema:DigitalDocument",
22     "schema:format": [
23       "application/vnd.geo+json"
24     ],
25     "schema:conformsTo": "https://igsn.org/schema/spatial.schema.json"
26   }
27 ],
28 "geosparql:hasGeometry": {
29   "@id": "_:N98e75cacc29f40deb555eb583cb162dc",
30   "@type": "http://www.opengis.net/ont/sf#Point",
31   "geosparql:asWKT": {
32     "@type": "http://www.opengis.net/ont/geosparql#wktLiteral",
33     "@value": "POINT(-76 -18)"
34   },
35   "geosparql:crs": {
36     "@id": "http://www.opengis.net/def/crs/OGC/1.3/CRS84"
37   }
38 },
39 "geoblob": {
40   "type": "GeometryCollection",
41   "geometries": [{
42     "type": "Point",
43     "coordinates": [-76, -18]
44   }]
45 },
46 "schema:spatialCoverage": {
47   "@type": "schema:Place",
48   "schema:geo": {
49     "@type": "schema:GeoCoordinates",
50     "schema:latitude": -18,
51     "schema:longitude": -76
52   }
53 }
54 }

```

```
<graphviz.dot.Digraph at 0x7f2f724a7be0>
```

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 pattern follows:
 - `spatialCoverage` -> `Place` -> `geo` -> `GeoCoordinates` OR `GeoShape`
- Some groups are using `GeoNode`
 - [schema.org](https://schema.org/issues) issues
- ICAN & Schema.org
- OGC SELFIE
- Think broad
- Science on Schema `spatial` for dataset guidance

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](https://www.hydra.cc/) 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.

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@type": "Action",
6   "@id": "https://us-central1-top-operand-112611.cloudfunctions.net/function-1",
7   "result": {
8     "@type": "DataDownload",
9     "encodingFormat": "text/plain",
10    "description": "a simple text result for the RGB counts"
11  },
12  "target": {
13    "@type": "EntryPoint",
14    "urlTemplate": "https://us-central1-top-operand-112611.cloudfunctions.net/
15 ➔function-1",
16    "httpMethod": "POST",
17    "contentType": ["image/jpeg", "image/png"]
18  },
19  "object": {
20    "@type": "ImageObject",
21    "description": "A JPEG or PNG to analyze the RGB counts"
22  }
23 }
```

```
<graphviz.dot.Digraph at 0x7fc5ac0e46d0>
```

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/>

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](#). This impedance is caused by publishing approaches for the terms that don't align well with the above publishing practices. This does not mean we can not use these terms, rather that we may find multiple ways to connect them used by the community. This can result in some ambiguity in linking in a community.

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.

Note: This topic of keyword linking with DefinedTerms is under review at the [Science on Schema](#) work at ESIP. Reference [Describing a Dataset](#) for the latest on their recommendations.

14.2 Keywords

The [Schema.org keywords](#) property of [CreativeWork](#) can point to three different values. These are: [DefinedTerm](#), [Text](#) and [URL](#).

We can see the three different approaches here to defining keywords. Here, *Region X* is a classic text keyword. The other two are defined as a [DefinedTerm](#).

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@type": "Map",
6   "@id": "https://example.org/id/XYZ",
7   "name": "Name or title of the document",
8   "description": "Description of the map to aid in searching",
9   "url": "https://www.sample-data-repository.org/creativework/map.pdf",
10  "identifier": {
11    "@id": "https://doi.org/10.5066/F7VX0DMQ",
12    "@type": "PropertyValue",
13    "propertyID": "https://registry.identifiers.org/registry/doi",
14    "value": "doi:10.5066/F7VX0DMQ",
15    "url": "https://doi.org/10.5066/F7VX0DMQ"
```

(continues on next page)

(continued from previous page)

```

16  },
17  "keywords": [
18    "Region X",
19    {
20      "@id": "http://purl.org/dc/dcmitype/Image",
21      "@type": "DefinedTerm",
22      "inDefinedTermSet": "http://purl.org/dc/terms/DCMIType",
23      "termCode": "Image",
24      "name": "Image"
25    },
26    {
27      "@id": "https://www.wikidata.org/wiki/Q350134",
28      "@type": "URL",
29      "url": "https://www.wikidata.org/wiki/Q350134"
30    }
31  ]
32 }

```

```
<graphviz.dot.Digraph at 0x7fd29fa92670>
```

14.2.1 Text

Keywords can be defined as a **Text** value. This is the most common approach though it doesn't provide some of the benefits of the other two approaches. For example, it doesn't allow for terms to be dereferenced on the net or for connects in the graph to be made for common terms by their subject IRIs.

Note: Be sure to use the [] notation to define the keyword. This defined an array of items vs a single items. If you use an approach like {"term1, term2, term4"} you have only created a single text string with comma separated values. However that is viewed as a single string in the graph. The [] notation creates an array of strings all connected to the subject IRI by the property *keywords*.

14.2.2 URL

Keywords can also point to a URL. This provides a way to link to a vocabulary entry that defines the term. This approach has some benefits of linking to more details but does easily provide an easy descriptive text for humans. There is nothing preventing putting in a text keyword followed up by another entry with a related URL.

14.2.3 DefinedTerm

This is the most complex approach. Keywords can point to a **DefinedTerm** as defined in a **DefinedTermSet** pointed to by the property **inDefinedTermSet**. It does offer the ability to present both a human focused textual name and description of the term. This is a great way to link to a vocabulary entry that defines the term. It also allows for a URL to be used to link to the vocabulary entry. While this approach is the most comprehensive, it does incur a complexity during the query process to extract and present the information.

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” and “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](https://schema.org). So we can use them in places such as:

- CreativeWork -> keyword
- CreativeWork -> learningResourceType
- LearningResource -> teaches (and many others)
- LearningResource -> competencyRequired (and many others)
- PropertyValue -> valueReference

The following example is from the [Schema.org](https://schema.org) DefinedTermSet reference.

```
[
  {
    "@context": {
      "@vocab": "https://schema.org/"
    },
  },
  {
    "@type": "DefinedTermSet",
    "@id": "http://openjurist.org/dictionary/Ballentine",
    "name": "Ballentine's Law Dictionary",
    "description": "A description of Ballentine's Law Dictionary Term Set"
  },
  {
    "@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": {
      "@type": "DefinedTermSet",
      "@id": "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": {
      "@type": "DefinedTermSet",
      "@id": "http://openjurist.org/dictionary/Ballentine"
    }
  }
]
```

```
<graphviz.dot.Digraph at 0x7fd2b4bb2f70>
```

14.4 References

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

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 notation.

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

```
1 {
2   "@context": {
3     "vocab": "http://example.com/vocab/",
4     "label": {
5       "@id": "vocab:label",
6       "@container": "@language"
7     }
8   },
9   "@id": "http://example.com/queen",
10  "label": {
11    "en": "The Queen",
12    "de": [ "Die Königin", "Ihre Majestät" ]
13  }
14 }
```

```
<graphviz.dot.Digraph at 0x7f600953c6d0>
```

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.

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 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 will review here. Namely the `subjectOf` and `publishingPrinciples` predicates.

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@id": "https://example.org/id/org/x",
6   "@type": "Organization",
7   "address": {
8     "@type": "PostalAddress",
9     "addressLocality": "Paris, France",
10    "postalCode": "F-75002",
11    "streetAddress": "38 avenue de l'Opera"
12  },
13   "email": "secretariat(at)example.org",
14   "name": "Organization X",
15   "description": "Description of org ...",
16   "telephone": "( 33 1) 42 68 53 00",
17   "member": [
18     {
19       "@type": "Organization",
20       "name": "Organization A",
```

(continues on next page)

(continued from previous page)

```

21         "description": "Org A is a potential parent organization of Org X"
22     }
23 },
24     "identifier": {
25         "@id": "https://grid.ac/institutes/grid.475727.4",
26         "@type": "PropertyValue",
27         "description": "UN Department of Economic and Social Affairs Sustainable
↳Development",
28         "propertyID": "https://registry.identifiers.org/registry/grid",
29         "url": "https://grid.ac/institutes/grid.475727.4"
30     },
31     "subjectOf": {
32         "@type": "CreativeWork",
33         "@id": "http://purl.unep.org/sdg/SDGIO_00020173",
34         "url": "http://www.ontobee.org/ontology/SDGIO?iri=http://purl.unep.org/sdg/
↳SDGIO_00020173",
35         "name": "UNSD SDG indicator code:C140c01",
36         "description": "Number of countries making progress ... the oceans and their
↳resources"
37     },
38     "publishingPrinciples": [
39         {
40             "@type": "CreativeWork",
41             "@id": "https://sdgs.un.org/goals/goal14",
42             "url": "https://sdgs.un.org/goals/goal14",
43             "name": "Sustainable Development Goal 14",
44             "description": "Conserve and sustainably use the oceans, seas and marine
↳resources for sustainable development"
45         },
46         {
47             "@type": "CreativeWork",
48             "@id": "https://dx.doi.org/10.1038/sdata.2016.18",
49             "url": "https://www.nature.com/articles/sdata201618",
50             "name": "FAIR data principles",
51             "description": "FAIR Principles definition as referenced from: Wilkinson,
↳M. D. et al. The FAIR Guiding Principles for scientific data management and
↳stewardship."
52         }
53     ]
54 }

```

<graphviz.dot.Digraph at 0x7f6ed5fb3820>

16.1.1 subjectOf

Values expected to be one of these types

- Event
- CreativeWork

Range

Used on these types

- Thing

Domain

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 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`. Reference 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 0x7f6ed52fcac0>
```

16.1.2 publishingPrinciples

Values expected to be one of these types

- [CreativeWork](#)
- [Organization](#)
- [Person](#)

Range

Used on these types

- [CreativeWork](#)
- [URL](#)

Domain

Lets take a look at `publishingPrinciples`. This can be used to connect `CreativeWork`, `Organization`, or `Person` to either of `CreativeWork` or `URL`. So this allows us to link a `CreativeWork` to a policy or principle statement. This has some very useful use cases where resources can be grouped based on their connection to those principles and policies.

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

(continues on next page)

(continued from previous page)

```

},
"@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 0x7f6ed52fdc40>

16.2 Sustainable Development Goals

The following example provides an approach to connecting Sustainable Development Goals (SDGs) could be linked to via `subjectOf`.

As this is a `CreateWork`, we can now use one more linking property, the [Schema.org](https://schema.org/citation) citation property. By comparison, the `publishingPrinciples` or `subjectOf` connections carry a bit more semantic meaning.

```

1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@type": "CreativeWork",
6   "@id": "https://example.org/id/XYZ",
7   "name": "Name or title of the document",
8   "description": "Description of the resource to aid in searching",
9   "distribution": {
10     "@type": "DataDownload",
11     "contentUrl": "https://www.sample-data-repository.org/dataset/472032.tsv",
12     "encodingFormat": "text/tab-separated-values"
13   },
14   "citation": {
15     "@type": "CreativeWork",
16     "@id": "http://purl.unep.org/sdg/SDGIO_00020173",
17     "url": "http://www.ontobee.org/ontology/SDGIO?iri=http://purl.unep.org/sdg/SDGIO_
resources
00020173",
18     "name": "UNSD SDG indicator code:C140c01",
19     "description": "Number of countries making progress ... the oceans and their
resources"

```

(continues on next page)

(continued from previous page)

```

20 },
21 "maintainer" : {
22   "@type" : "Organization",
23   "@id": "https://ror.org/050bms902",
24   "description": "UN Department of Economic and Social Affairs Sustainable
↪Development"
25 }
26 }

```

```
<graphviz.dot.Digraph at 0x7f6ed5742af0>
```

16.2.1 citation

Values expected to be one of these types

- [Text](#)
- [CreativeWork](#)

Range

Used on these types

- [CreativeWork](#)

Domain

[Schema.org citation](#) provides a way to link to another creative work. This property can be pointed to either [Text](#) or [CreativeWork](#). It should also be noted that citation can only be used on type [CreativeWork](#).

Due to the limit to use on [CreativeWork](#) only, this example is not seen in the above example which is of type [Organization](#).

The actual semantics of citation is rather vague stating it is a method to cite or reference another creative work.

```

{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@id": "https://example.org/id/XYZ",
  "@type": "CreativeWork",
  "citation": {
    "@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 0x7f6ed5308790>
```

16.3 Refs

- [SDGs](#)
- [SDG targets](#)
- [SDG indicators](#)

SOURCE AND PROV APPROACHES

17.1 About

This section will present an identifier discussion and then focus on approaches publishers can use to provide more source and provenance information. This section is heavily influenced by [1] [2] and the reader is highly encouraged to read these references.

It is not uncommon for a single resource to be described at multiple locations. The following items represent properties that can help address the disambiguation of resources as the Ocean InfoHub graph grows.

We will look at the following seven properties: identifier, provider, publisher, sameAs, isBasedOn, subjectOf and its inverse, about.

It should be noted that only subjectOf and sameAs can be used on type Thing. That is, these properties can be used on any [Schema.org](https://schema.org/) type.

The others have their domain and range values listed. All these properties work on CreativeWork. So they are all valid for that type and subtypes like Dataset and Map.

17.1.1 Example Graph

The following example graph shows some of the properties we can use to provide source and provenance information about a resource.

```
1 {
2   "@context": {
3     "@vocab": "https://schema.org/"
4   },
5   "@id": "https://example.org/id/XYZ",
6   "@type": "CreativeWork",
7   "sameAs": "https://doi.org/10.5066/F7VX0DMQ",
8   "isBasedOn": "https://example.org/id/xyz",
9   "identifier": {
10     "@id": "https://doi.org/10.5066/F7VX0DMQ",
11     "@type": "PropertyValue",
12     "propertyID": "https://registry.identifiers.org/registry/doi",
13     "url": "https://doi.org/10.5066/F7VX0DMQ",
14     "value": "doi:10.5066/F7VX0DMQ"
15   },
16   "subjectOf": {
17     "@type": "CreativeWork",
18     "@id": "http://purl.unep.org/sdg/SDGIO_00020173",
19     "url": "http://www.ontobee.org/ontology/SDGIO?iri=http://purl.unep.org/sdg/SDGIO_00020173",
```

(continues on next page)

(continued from previous page)

```

20     "name": "UNSD SDG indicator code:C140c01",
21     "description": "Number of countries making progress ... the oceans and their
    resources"
22   },
23   "provider": {
24     "@id": "https://www.sample-data-repository.org",
25     "@type": "Organization",
26     "legalName": "Sample Data Repository Office",
27     "name": "SDRO",
28     "sameAs": "http://www.re3data.org/repository/r3dxxxxxxxx",
29     "url": "https://www.sample-data-repository.org"
30   },
31   "publisher": {
32     "@id": "https://www.sample-data-repository.org"
33   }
34 }

```

<graphviz.dot.Digraph at 0x7fec886766d0>

17.1.2 identifier

This is the main subject of the start of this section. Please refer there for details on this property.

Values expected to be one of these types

- [PropertyValue](#)
- [Text](#)
- [URL](#)

Range

Used on these types

- [Thing](#)

Domain

```

{
  "@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 0x7fec7aa4dcd0>

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 (pointing to type URL)

URL of the item.

17.1.3 provider

schema.org/provider

The service provider, service operator, or service performer; the goods producer. Another party (a seller) may offer those services or goods on behalf of the provider. A provider may also serve as the seller.

For OIH this is the agent that is responsible for distributing the resource and the descriptive metadata. That is, the provider actually runs and supports the services that presents the resource on the net or otherwise makes the data available.

Values expected to be one of these types

- [Organization](#)
- [Person](#)

Range

Used on these types

- [CreativeWork](#)
- [EducationalOccupationalProgram](#)
- [Invoice](#)
- [ParcelDelivery](#)
- [Reservation](#)
- [Service](#)
- [Trip](#)

Domain

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

(continues on next page)

(continued from previous page)

```
"@type": "CreativeWork",
"provider": {
  "@id": "https://www.sample-data-repository.org",
  "@type": "Organization",
  "legalName": "Sample Data Repository Office",
  "name": "SDRO",
  "sameAs": "http://www.re3data.org/repository/r3dxxxxxxx",
  "url": "https://www.sample-data-repository.org"
}
```

<graphviz.dot.Digraph at 0x7fec7a9bdca0>

17.1.4 publisher

See: schema.org/publisher

The publisher is defined as “The publisher of the creative work”. This is viewed as the agent that is primarily responsible for making the content described by the structured metadata.

Values expected to be one of these types

- [Organization](#)
- [Person](#)

Range

Used on these types

- [CreativeWork](#)

Domain

```
{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@id": "https://example.org/id/XYZ",
  "@type": "CreativeWork",
  "publisher": {
    "@id": "https://www.sample-data-repository.org",
    "@type": "Organization",
    "legalName": "Sample Data Repository Office",
    "name": "SDRO",
    "sameAs": "http://www.re3data.org/repository/r3dxxxxxxx",
    "url": "https://www.sample-data-repository.org"
  }
}
```

<graphviz.dot.Digraph at 0x7fec7ae266d0>

17.1.5 sameAs

See: schema.org/sameAs

The sameAs property links a Thing to a URL. It is expected that the URL is a resource that is the the most canonical URL for the original.

In cases where your resource is not the canonical URL, you can use the sameAs property to link to the canonical URL. This is useful when you wish to publish a resource and give credit to the original resource. Note, in this case the resource must not be altered, but actually be the same as the canonical URL resource.

The sameAs property should always point to only one resource. It is not logically consistent to point to multiple sameAs resources.

Values expected to be one of these types

- [URL](#)

Range

Used on these types

- [Thing](#)

Domain

```
{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@id": "https://example.org/id/XYZ",
  "@type": "CreativeWork",
  "sameAs": "https://doi.org/10.5066/F7VX0DMQ"
}
```

```
<graphviz.dot.Digraph at 0x7fec7a9bd0d0>
```

17.1.6 isBasedOn

See: schema.org/isBasedOn

Where sameAs is used to link to the canonical URL of the resource, isBasedOn provides a means to link derivative works to the original resource this new resources is based on.

The isBasedOn property can be used to link to multiple resources if more than one was used in the generation of this new resource.

Values expected to be one of these types

- [CreativeWork](#)
- [Product](#)
- [URL](#)

Range

Used on these types

- [CreativeWork](#)

Domain

17.1.7 subjectOf and inverse about

See: schema.org/subjectOf

The property `subjectOf` of can be used to indicate a resource that is related to the described resource, although not necessarily a part of it. The `subjectOf` property can be used in an educational framework to indicate the field(s) of science or literature the dataset relates to.

Values expected to be one of these types

- [CreativeWork](#)
- [Event](#)

Range

Used on these types

- [Thing](#)

Domain

The `subjectOf` property has an inverse-property [about](#). The property `about` can be used to:

indicate the subject matter this thing is about

Values expected to be one of these types

- [Thing](#)

Range

Used on these types

- [CreativeWork](#)
- [Event](#)
- [CommunicateAction](#)

Domain

```
{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@id": "https://example.org/id/XYZ",
  "@type": "CreativeWork",
  "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 0x7fec8865aa30>
```

17.2 References

Part III

Aggregation

AGGREGATOR

18.1 Introduction

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 [Overview 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.

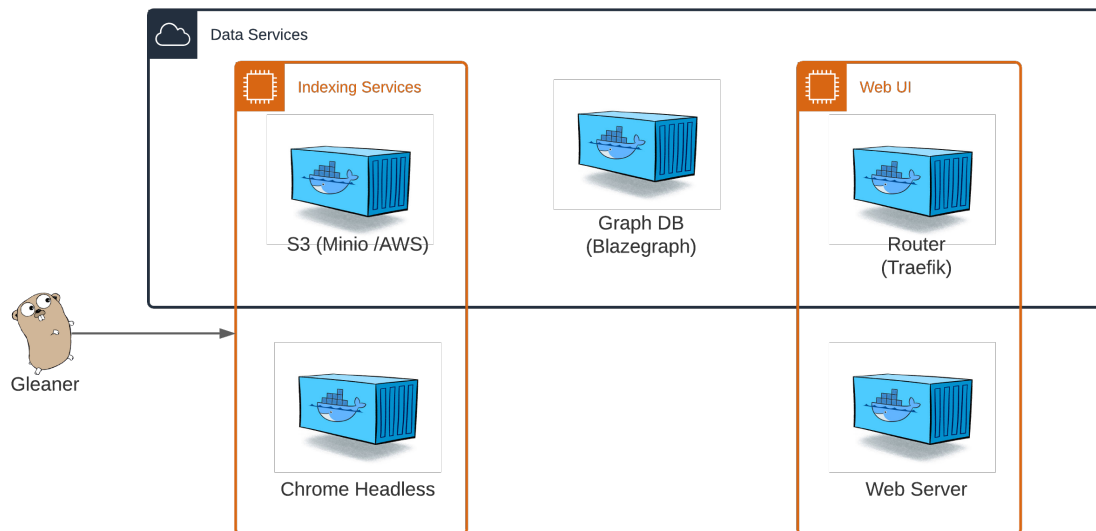


Fig. 1: The various compose options for Gleaner

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 [Caddy](#) 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 elsewhere 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 Services* box above. More details on this set can be found in [Indexing Services](#).

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.

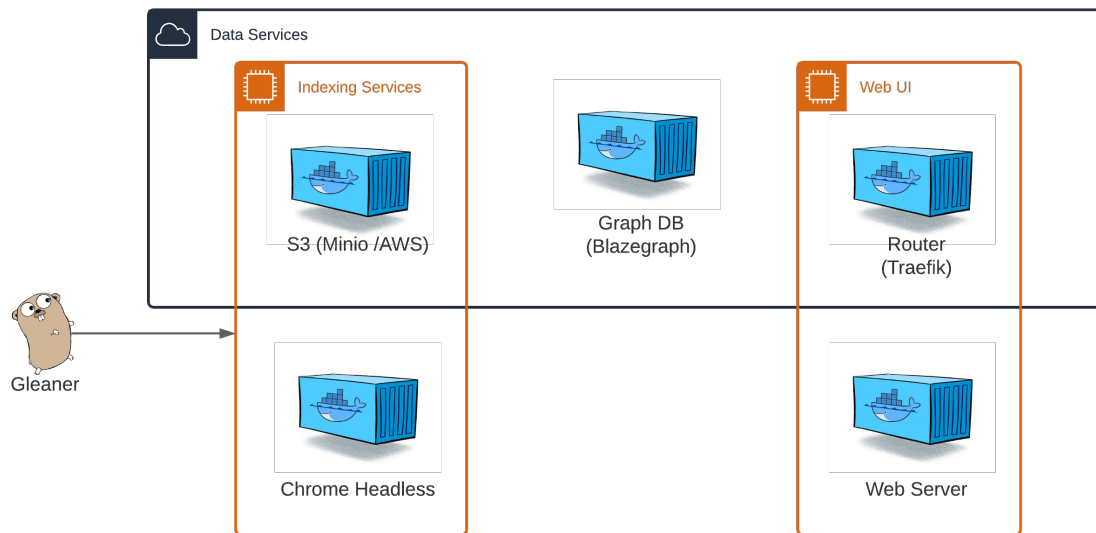
18.2 ODIS Catalog as Index Source

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 is 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.

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 environment. 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](https://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 to 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 won't 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 the [Gleaner-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)

```
6
7 # local data volumes
8 export GLEANER_BASE=/tmp/gleaner/
9 mkdir -p ${GLEANER_BASE}
10 export GLEANER_OBJECTS=${GLEANER_BASE}/datavol/s3
11 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.

```

1  ---
2  minio:
3    address: 0.0.0.0
4    port: 9000
5    accessKey: worldsbestaccesskey
6    secretKey: worldsbestsecretkey
7    ssl: false
8    bucket: gleaner
9  gleaner:
10   runid: oih # this will be the bucket the output is placed in...
11   summon: true # do we want to visit the web sites and pull down the files
12   mill: true
13  context:
14   cache: true
15  contextmaps:
16   - prefix: "https://schema.org/"
17     file: "./jsonldcontext.json" # wget http://schema.org/docs/jsonldcontext.jsonld
18   - prefix: "http://schema.org/"
19     file: "./jsonldcontext.json" # wget http://schema.org/docs/jsonldcontext.jsonld
20  summoner:
21   after: "" # "21 May 20 10:00 UTC"
22   mode: full # full || diff: If diff compare what we have currently in gleaner to
23   ↪ sitemap, get only new, delete missing
24   threads: 1
25   delay: 0 # milliseconds (1000 = 1 second) to delay between calls (will FORCE
26   ↪ threads to 1)
27   headless: http://0.0.0.0:9222 # URL for headless see docs/headless
28  millers:
29   graph: true
30   #geojson: false
31  sitegraphs:
32   - name: aquadocs
33     url: https://oih.aquadocs.org/aquadocs.json
34     headless: false
35     pid: https://www.re3data.org/repository/aquadocs
36     properName: AquaDocs
37     domain: https://aquadocs.org
38  sources:
39   - name: samplesearth
40     url: https://samples.earth/sitemap.xml
41     headless: false
42     pid: https://www.re3data.org/repository/samplesearth
43     properName: Samples Earth (DEMO Site)
44     domain: https://samples.earth
45   - name: marinetraining
46     url: https://www.marinetraining.eu/sitemap.xml
47     headless: false
48     pid: https://www.re3data.org/repository/marinetraining
49     properName: Marine Training EU
50     domain: https://marinetraining.eu/

```

(continues on next page)

(continued from previous page)

```
49 - name: marineie
50   url: http://data.marine.ie/geonetwork/srv/eng/portal.sitemap
51   headless: true
52   pid: https://www.re3data.org/repository/marineie
53   properName: Marine Institute Data Catalogue
54   domain: http://data.marine.ie
55 - name: oceanexperts
56   url: https://oceanexpert.org/assets/sitemaps/sitemapTraining.xml
57   headless: false
58   pid: https://www.re3data.org/repository/oceanexpert
59   properName: OceanExpert UNESCO/IOC Project Office for IODE
60   domain: https://oceanexpert.org/
61 # - name: obis
62 #   url: https://obis.org/sitemap/sitemap_datasets.xml
63 #   headless: false
64 #   pid: https://www.re3data.org/repository/obis
65 #   properName: Ocean Biodiversity Information System
66 #   domain: https://obis.org
```

Object store

```
1 minio:
2   address: 0.0.0.0
3   port: 9000
4   accessKey: worldsbestaccesskey
5   secretKey: worldsbestsecretkey
6   ssl: false
7   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

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

This passes a few high level concepts.

- runid:
- summon
- mill

Context sections

```
1 context:
2   cache: true
3 contextmaps:
4   - prefix: "https://schema.org/"
5     file: "./jsonldcontext.json" # wget http://schema.org/docs/jsonldcontext.jsonld
6   - prefix: "http://schema.org/"
7     file: "./jsonldcontext.json" # wget http://schema.org/docs/jsonldcontext.jsonld
```

Comments for the context sections

Summoner section

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

Comments for the summoner sections

Millers section

```
1 millers:
2   graph: true
3   #geojson: false
```

Comments for the miller sections

Site graphs section

```
1 sitegraphs:
2   - name: aquadocs
3     url: https://oih.aquadocs.org/aquadocs.json
4     headless: false
5     pid: https://www.re3data.org/repository/aquadocs
6     properName: AquaDocs
7     domain: https://aquadocs.org
```

Comments for the sitegraph sections

Sources section

```
1 sources:
2 - name: samplesearth
3   url: https://samples.earth/sitemap.xml
4   headless: false
5   pid: https://www.re3data.org/repository/samplesearth
6   properName: Samples Earth (DEMO Site)
7   domain: https://samples.earth
8 - name: marinetraining
9   url: https://www.marinetraining.eu/sitemap.xml
10  headless: false
11  pid: https://www.re3data.org/repository/marinetraining
12  properName: Marine Training EU
13  domain: https://marinetraining.eu/
14 - name: marineie
15   url: http://data.marine.ie/geonetwork/srv/eng/portal.sitemap
16   headless: true
17   pid: https://www.re3data.org/repository/marineie
18   properName: Marine Institute Data Catalogue
19   domain: http://data.marine.ie
20 - name: oceanexperts
21   url: https://oceanexpert.org/assets/sitemaps/sitemapTraining.xml
22   headless: false
23   pid: https://www.re3data.org/repository/oceanexpert
24   properName: OceanExpert UNESCO/IOC Project Office for IODE
25   domain: https://oceanexpert.org/
26 # - name: obis
27 #   url: https://obis.org/sitemap/sitemap_datasets.xml
28 #   headless: false
29 #   pid: https://www.re3data.org/repository/obis
30 #   properName: Ocean Biodiversity Information System
31 #   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](#)

GLEANER CLI DOCKER

20.1 About

This is a new approach for quick starts with Gleaner. It is a script that exposes a containerized version of Gleaner as a CLI interface.

You can use the `-init` flag to pull down all the support files you need including the Docker Compose file for setting up the object store, a triplestore and the support for headless indexing.

20.2 Prerequisites

You need Docker installed. Later, to work with the results and load them into a triplestore, you will also need an S3 compatible client. We will use the Minio client, `mc`, for this.

20.3 Steps

Download the script `gleanerDocker.sh` from <https://github.com/earthcubearchitecture-project418/gleaner/tree/master/docs/cliDocker> You may need to make it run-able with

```
curl -O https://raw.githubusercontent.com/earthcubearchitecture-project418/gleaner/  
master/docs/cliDocker/gleanerDocker.sh  
  
chmod 755 gleanerDocker.sh
```

Next you can run the script with the `-init` flag to pull down all the support files you need.

```
./gleanerDocker.sh -init
```

This will also download the needed docker image and the support files. Your directory should look like this now:

```
files@ubuntu:~/clidocker# ls -lt  
total 1356  
-rw-r--r-- 1 files files 1281 Aug 15 14:07 gleaner-IS.yml  
-rw-r--r-- 1 files files 290 Aug 15 14:07 setenvIS.sh  
-rw-r--r-- 1 files files 1266 Aug 15 14:07 demo.yaml  
-rw-r--r-- 1 files files 1371350 Aug 15 14:07 schemaorg-current-https.jsonld  
-rwxr-xr-x 1 files files 1852 Aug 15 14:06 gleanerDocker.sh
```

Let's see if we can setup our support infrastructure for Gleaner. The file `gleaner-IS.yml` is a docker compose file that will set up the object store, and a triplestore.

To do this we need to set up a few environment variables. To do this we can leverage the `setenvIS.sh` script. This script will set up the environment we need. Note you can also use a `.env` file or other approaches. You can reference the [Environment variables in Compose documentation](#).

```
root@ubuntu:~/clidocker# source setenvIS.sh
root@ubuntu:~/clidocker# docker-compose -f gleaner-IS.yml up -d
Creating network "clidocker_traefik_default" with the default driver
Creating clidocker_triplestore_1 ... done
Creating clidocker_s3system_1    ... done
Creating clidocker_headless_1   ... done
```

Note: In a fresh run all the images will be pulled down. This may take a while.

In the end, you should be able to see these images running:

```
root@ubuntu:~/clidocker# docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
STATUS        PORTS
Names
a26f7c945479   nawer/blazegraph                   "docker-entrypoint.s..." About a
minute ago    Up About a minute    0.0.0.0:9999->9999/tcp
clidocker_triplestore_1
f3a4197c42be   minio/minio:latest                "/usr/bin/docker-ent..." About a
minute ago    Up About a minute    0.0.0.0:9000->9000/tcp, 0.0.0.0:54321->54321/tcp
clidocker_s3system_1
062f029462b1   chromedp/headless-shell:latest     "/headless-shell/hea..." About a
minute ago    Up About a minute    0.0.0.0:9222->9222/tcp
```

At this point we should be able to do a run. During the init process a working config file was downloaded.

Note: This config file will change... it's pointing to an OIH partner and I will not do that for the release. I have a demo site I will use.

Next we need to setup our object for Gleaner. Gleaner itself can do this task so we will use

```
root@ubuntu:~/clidocker# ./gleanerDocker.sh -setup -cfg demo
main.go:35: EarthCube Gleaner
main.go:110: Setting up buckets
check.go:58: Gleaner Bucket gleaner not found, generating
main.go:117: Buckets generated. Object store should be ready for runs
```

Note: Here is where we go off the rails. The config file uses 0.0.0.0 as the location and this is not working. You need to edit the config file with the "real" IP of the host machine. In my case is this 192.168.122.77. This is obviously still a local network IP but it does work. I am still investigating this issue.

We can now do a run with the example template file.

Note: Best to delete the "sitegraph" node, I will do that soon. It should work, but is currently slow and gives little feedback

If everything goes well, you should see something like the following:

```
root@ubuntu:~/clidocker# ./gleanerDocker.sh -cfg demo
main.go:35: EarthCube Gleaner
main.go:122: Validating access to object store
check.go:39: Validated access to object store: gleaner.
org.go:156: Building organization graph (nq)
```

(continues on next page)

(continued from previous page)

```
org.go:163: {samplesearch https://samples.earth/sitemap.xml false https://www.
↳re3data.org/repository/samplesearch Samples Earth (DEMO Site) https://samples.earth}
main.go:154: Sitegraph(s) processed
summoner.go:16: Summoner start time: 2021-08-15 14:34:08.907152656 +0000 UTC m=+0.
↳067250623
resources.go:74: samplesearch : 202
100%↳
↳
↳(202/202, 20 it/s)
summoner.go:34: Summoner end time: 2021-08-15 14:34:20.36804137 +0000 UTC m=+11.
↳528139340
summoner.go:35: Summoner run time: 0.191015
webfeed.go:37: 1758
millers.go:26: Miller start time: 2021-08-15 14:34:20.368063453 +0000 UTC m=+11.
↳528161421
millers.go:40: Adding bucket to milling list: summoned/samplesearch
millers.go:51: Adding bucket to prov building list: prov/samplesearch
100%↳
↳
↳(202/202, 236 it/s)
graphng.go:77: Assembling result graph for prefix: summoned/samplesearch to: milled/
↳samplesearch
graphng.go:78: Result graph will be at: results/runX/samplesearch_graph.nq
pipecopy.go:16: Start pipe reader / writer sequence
graphng.go:84: Pipe copy for graph done
millers.go:80: Miller end time: 2021-08-15 14:34:21.84702814 +0000 UTC m=+13.007126109
millers.go:81: Miller run time: 0.024649
```

20.4 Working with results

If all has gone well, at this point you have downloaded the JSON-LD documents into Minio or some other object store. Next we will install a client that we can use to work with these objects.

Note, there is a web interface exposed on the port mapped in the Docker compose file. In the case of these demo that is 9000. You can access it at <http://localhost:9000/> with the credentials set in the environment variable file.

However, to work with these objects it would be better to use a command line tool, like mc. The Minio Client, can be installed following their [Minio Client Quickstart Guide](#). Be sure to place it somewhere where it can be seen from the command line, ie, make sure it is in your PATH variable.

If you are on Linux this might look something like:

```
wget https://dl.min.io/client/mc/release/linux-amd64/mc
chmod +x mc
./mc --help
```

There is also a [Minio Client Docker image](#) that you can use as well but it will be more difficult to use with the following scripts due to container isolation.

To man an entry in the mc config use:

```
mc alias set oih http://localhost:9000 worldsbestaccesskey worldsbestsecretkey
```

We should now be able to list our object store. We have set it up using the alias *oih*.

```
user@ubuntu:~/clidocker# mc ls oih
[2021-08-15 14:31:20 UTC]      0B gleaner/
user@ubuntu:~/clidocker# mc ls oih/gleaner
[2021-08-19 13:36:04 UTC]      0B milled/
[2021-08-19 13:36:04 UTC]      0B orgs/
[2021-08-19 13:36:04 UTC]      0B prov/
[2021-08-19 13:36:04 UTC]      0B results/
[2021-08-19 13:36:04 UTC]      0B summoned/
```

You can explore mc and see how to copy and work with the object store.

20.5 Loading to the triplestore

As part of our Docker compose file we also spun up a triplestore. Let's use that now.

Now Download the `minio2blaze.sh` script.

```
curl -O https://raw.githubusercontent.com/earthcubearchitecture-project418/gleaner/
-master/scripts/minio2blaze.sh
chmod 755 minio2blaze.sh
```

The content we need to load into the triplestore needs to be in RDF for Blazegraph. We also need to tell the triplestore how we have encoded that RDF. If look in the object store at

```
mc ls oih/gleaner/milled
[2021-08-19 13:26:52 UTC]      0B samplesearch/
```

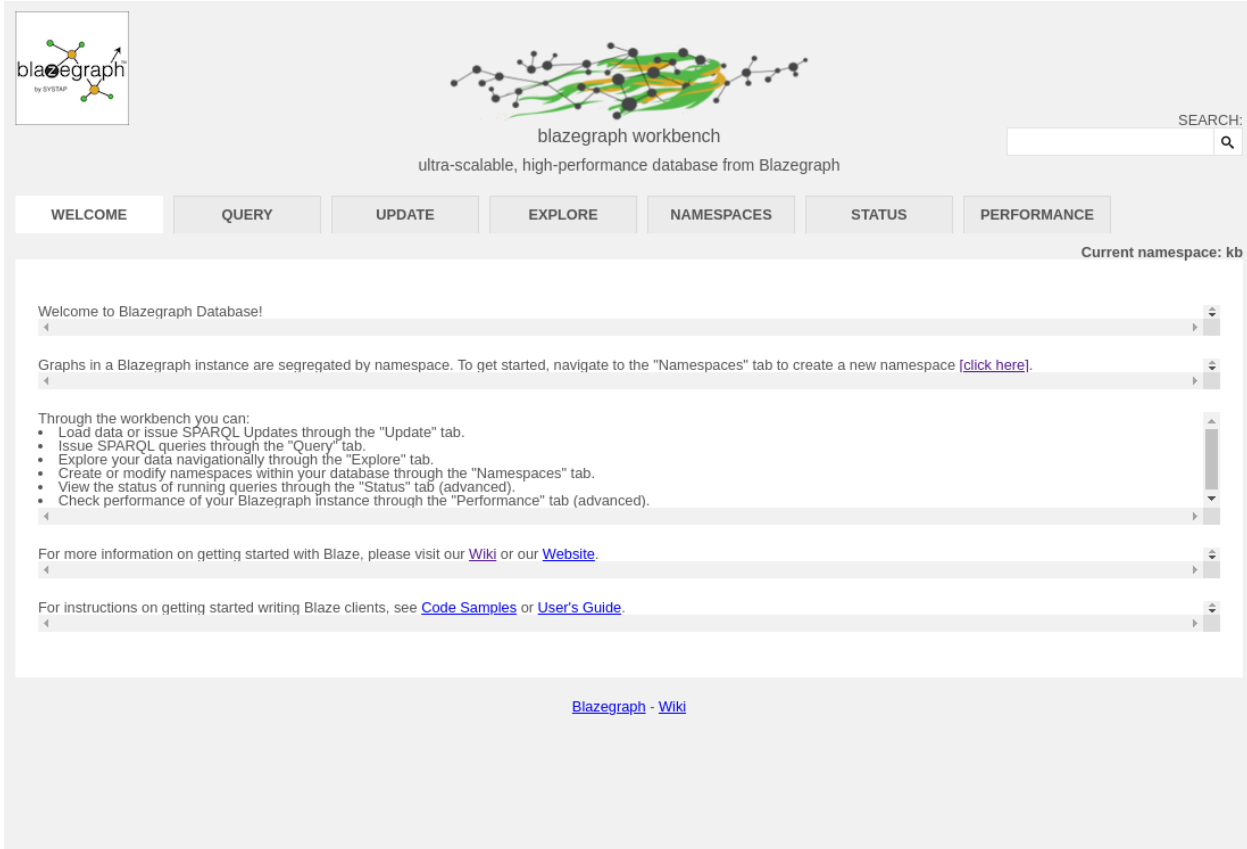
We should see a bucket that is holding the RDF data converted from the JSON-LD. Let's use this in our test. We can pass this path to the `minio2blaze.sh` script. This script will go looking for the mc command we installed above, so be sure it is in a PATH location that script can see.

```
./minio2blaze.sh oih/gleaner/milled/samplesearch
... lots of results removed
```

If all has gone well, we should have RDF in the triplestore. We started our triplestore as part of the docker-compose.yml file. You can visit the triplestore at <http://localhost:9999/blazegraph/#splash>

Note, you may have to try other addresses other than `localhost` if your networking is a bit different with Docker. For me, I had to use a real local IP address for my network, you might also try `0.0.0.0`.

Hopefully you will see something like the following.




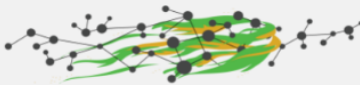
The image shows the Blazegraph Workbench interface. At the top left is the Blazegraph logo. In the center is a network graph visualization. Below the graph, the text reads "blazegraph workbench" and "ultra-scalable, high-performance database from Blazegraph". On the right, there is a search bar with the label "SEARCH:" and a magnifying glass icon. Below the header, there is a row of tabs: WELCOME, QUERY, UPDATE, EXPLORE, NAMESPACES, STATUS, and PERFORMANCE. The "WELCOME" tab is currently selected. To the right of the tabs, it says "Current namespace: kb". The main content area contains several sections of text and links. The first section says "Welcome to Blazegraph Database!". The second section says "Graphs in a Blazegraph instance are segregated by namespace. To get started, navigate to the 'Namespaces' tab to create a new namespace [\[click here\]](#)". The third section lists several actions you can perform through the workbench: load data or issue SPARQL Updates through the "Update" tab, issue SPARQL queries through the "Query" tab, explore your data navigationally through the "Explore" tab, create or modify namespaces within your database through the "Namespaces" tab, view the status of running queries through the "Status" tab (advanced), and check performance of your Blazegraph instance through the "Performance" tab (advanced). The fourth section says "For more information on getting started with Blaze, please visit our [Wiki](#) or our [Website](#)". The fifth section says "For instructions on getting started writing Blaze clients, see [Code Samples](#) or [User's Guide](#)". At the bottom of the interface, there is a link to "Blazegraph - Wiki".

We loaded into the default *kb* namespace, so we should be good there. We can see that is listed as the active namespace at the *Current Namespace: kb* report.

Let's try a simple SPARQL query. Click on the *Query* tab to get to the query user interfaced. We can use something like:

```
select *
where
{
  ?s ?p ?o
}
LIMIT 10
```

A very simple SPARQL to give us the first 10 results from the triplestore. If all has gone well, we should see something like:

SEARCH:

blazegraph workbench

ultra-scalable, high-performance database from Blazegraph

WELCOME
QUERY
UPDATE
EXPLORE
NAMESPACES
STATUS
PERFORMANCE

Current namespace: kb

[Wiki - SPARQL Query](#)

Namespace shortcuts:
Bigdata
W3C
Dublin Core
Social/Other
Custom
Edit

```

1 select *
2 where
3 {
4   ?s ?p ?o
5 }
6 LIMIT 10

```

[Advanced features](#)

Execute Clear

s	p	
https://samples.earth/id/learning/clonhobh2h4brdov30j0	https://schema.org/description	steps Topics management ocean methods methodologies or decade manage
https://samples.earth/id/learning/clonhobh2h4brdov30j0	https://schema.org/hasCourseInstance	t13
https://samples.earth/id/learning/clonhobh2h4brdov30j0	https://schema.org/hasCourseInstance	t855
https://samples.earth/id/learning/clonhobh2h4brdov30j0	https://schema.org/name	Fake: students and expected
https://samples.earth/id/learning/clonhobh2h4brdov30j0	rdf:type	https://schema.org/Course
https://samples.earth/id/learning/clonhobh2h4brdov3060	https://schema.org/description	sites fields Managers. and typical information of management. and li
https://samples.earth/id/learning/clonhobh2h4brdov3060	https://schema.org/hasCourseInstance	t16
https://samples.earth/id/learning/clonhobh2h4brdov3060	https://schema.org/hasCourseInstance	t856
https://samples.earth/id/learning/clonhobh2h4brdov3060	https://schema.org/name	Fake: for
https://samples.earth/id/learning/clonhobh2h4brdov3060	rdf:type	https://schema.org/Course

Total results: 10, displaying 1-10
50
per page

Page 1 of 1

☐ Show datatypes
☐ Show languages

Time	Query	Results	Execution Time	Delete
2021-08-19T13:45:27.025Z	select * where { ?s ?p ?o } LIMIT 10	10	200ms	X

Clear history

Export Clear

[Blazegraph - Wiki](#)

You can explore more about SPARQL and the wide range of queries you can do with it at the [W3C SPARQL 1.1 Query Language](#) reference.

100

Chapter 20. Gleaner CLI Docker

20.6 Conclusion

We have attempted here to give a quick introduction to the use of Gleaner in a Docker environment. This is a very simple example, but it should give you an idea of the approach used. This approach can then be combined with other approaches documented to establish a more production oriented implementation. Most of this documentation will be located at the [Gleaner.io GitHub repository](#) and [Gleaner repository](#).

Note: The plan is to merge the [Gleaner.io](#) GitHub repository into the first.

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
```

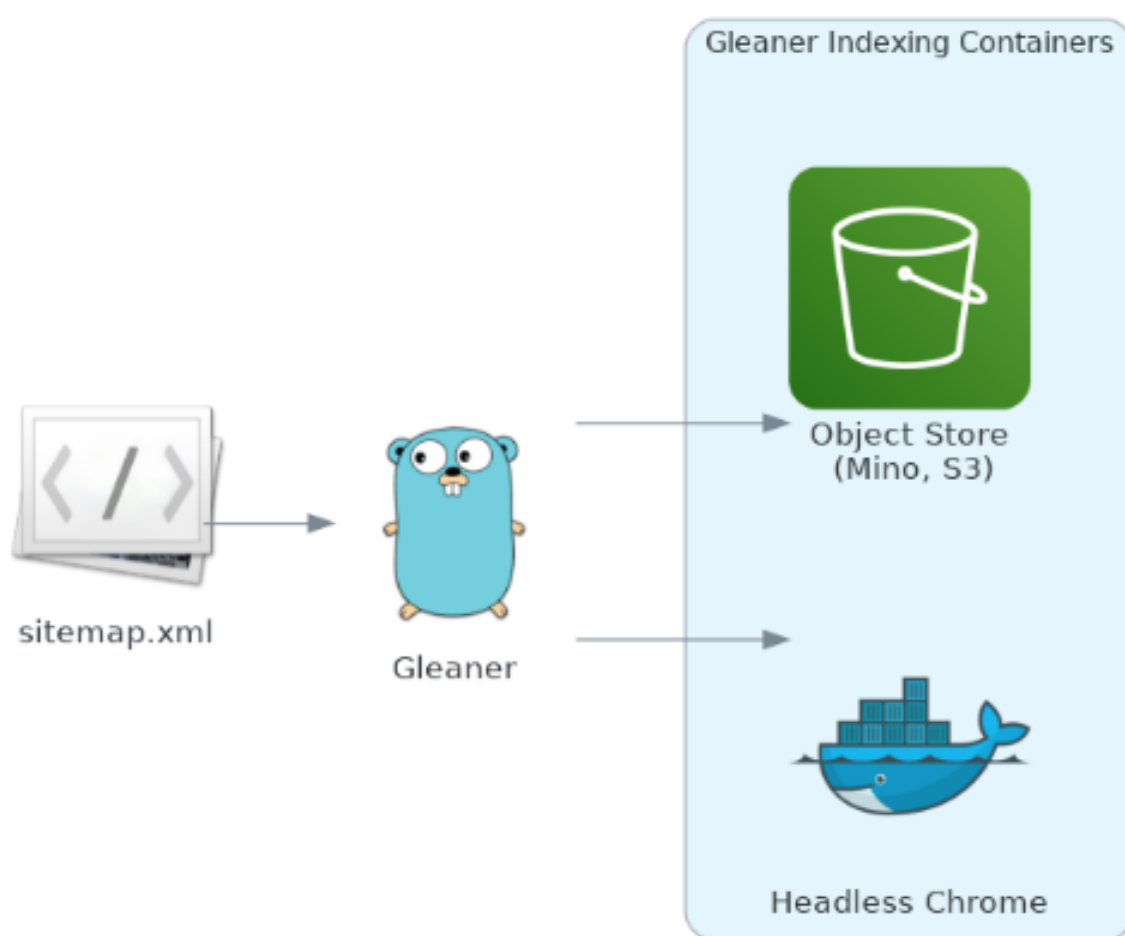


Fig. 1: Basic Gleaner Indexing Service Activity Workflow

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

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)

22.1 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.

22.1.1 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.

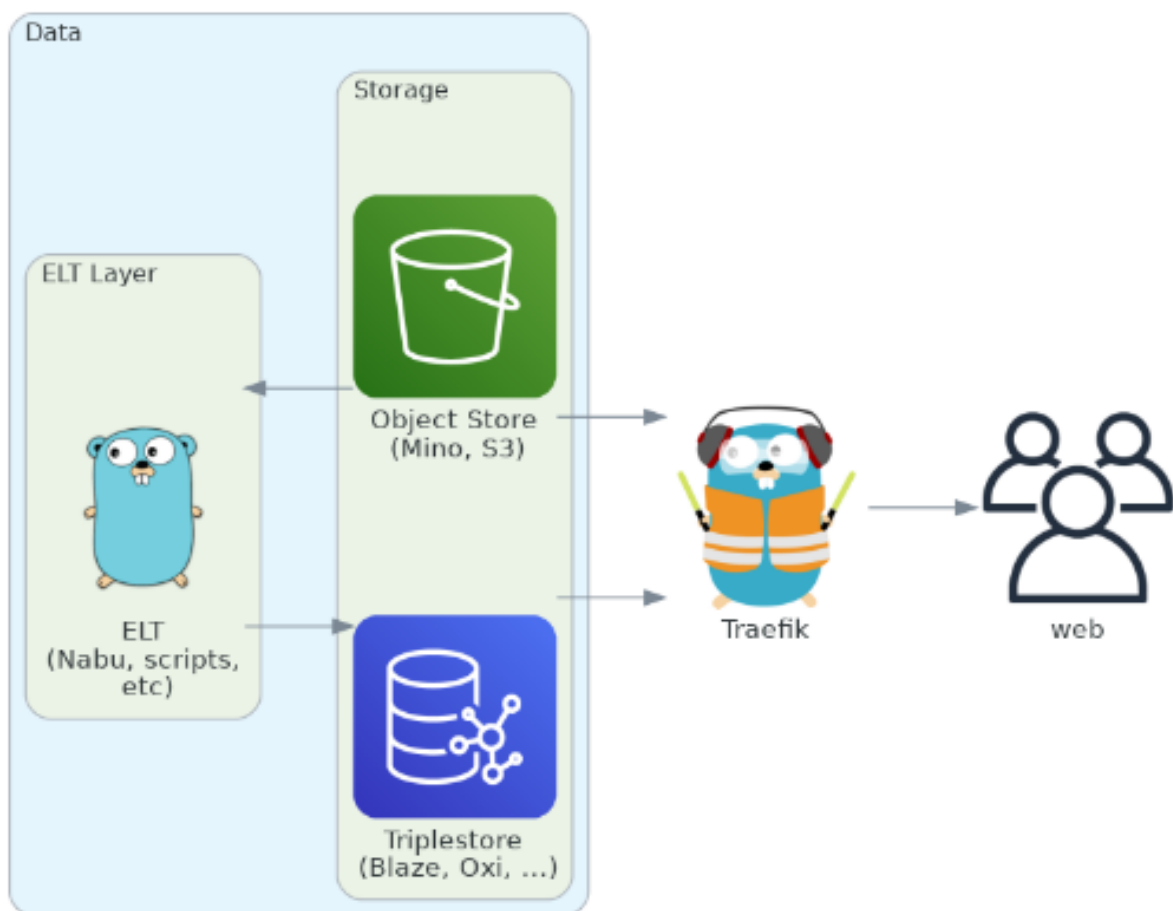


Fig. 1: Gleaner Data Service Activity Workflow

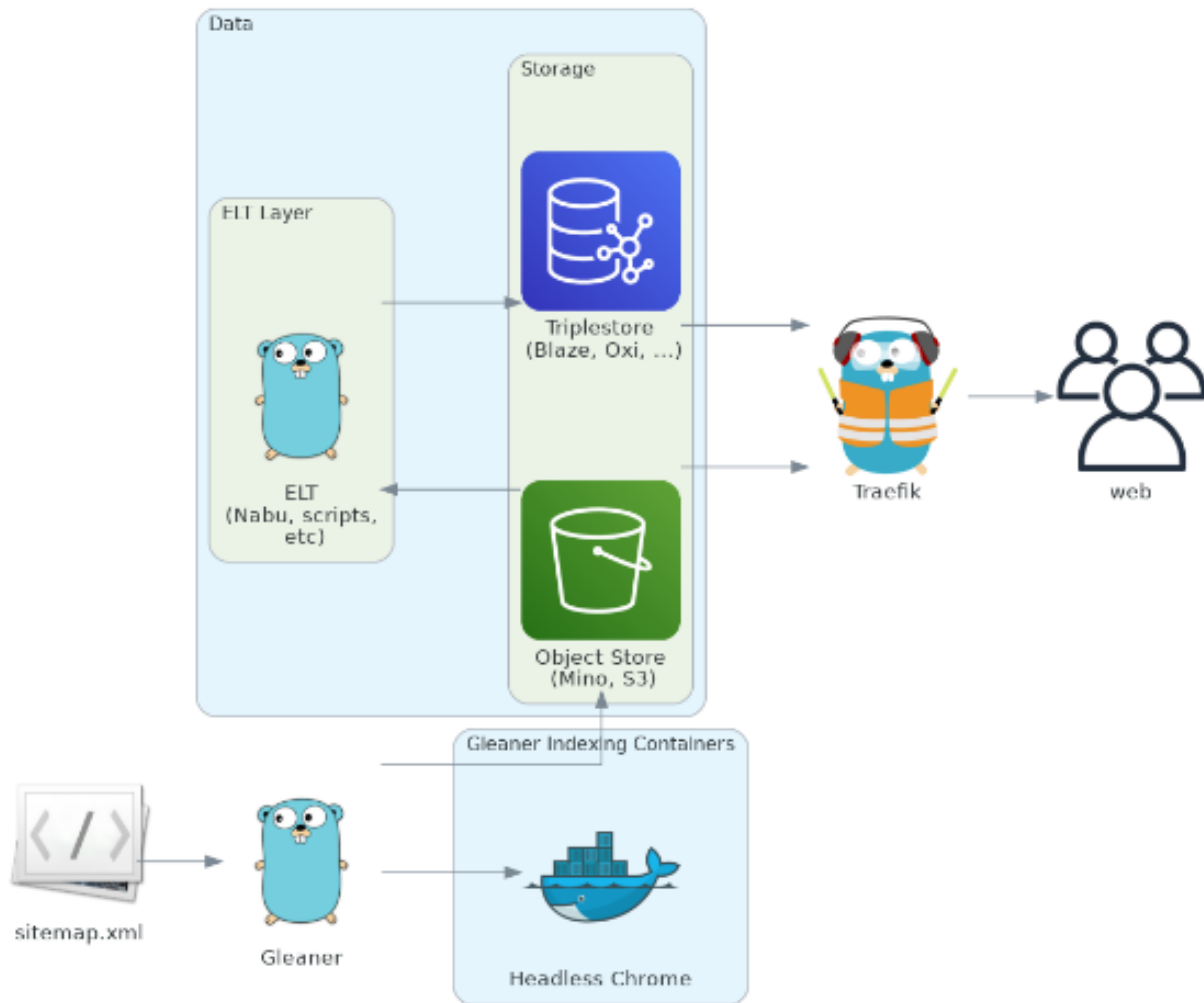


Fig. 2: Gleaner Indexing and Data Service Combined

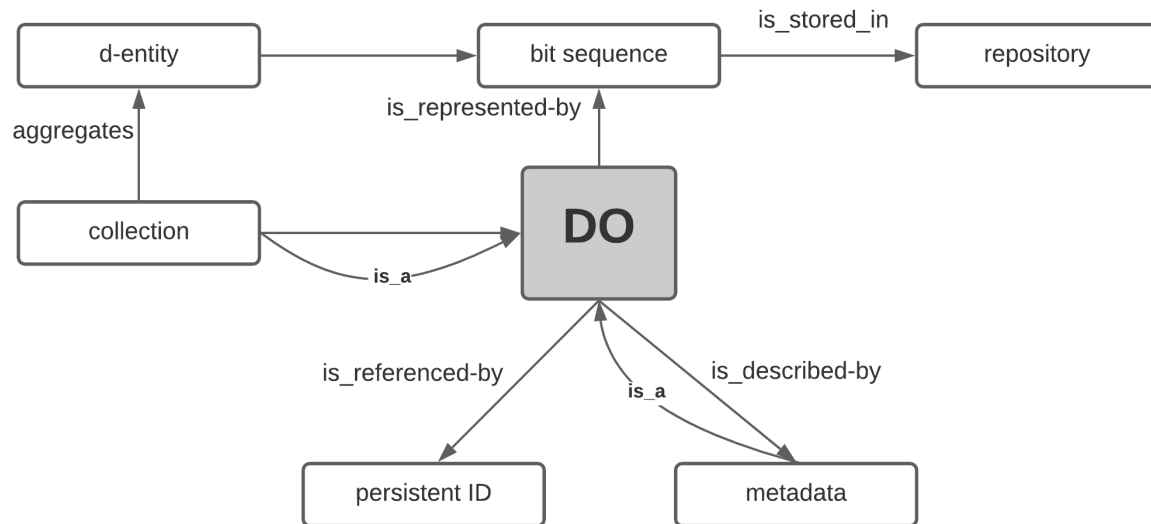


Figure 1: Research Data Alliance Digital Object Cloud Pattern

Fig. 3: Gleaner Digital Object Pattern

```

1  #!/bin/bash
2
3  # domains
4  export GLEANER_ADMIN_DOMAIN=admin.local.dev
5  export GLEANER_OSS_DOMAIN=oss.local.dev
6  export GLEANER_GRAPH_DOMAIN=graph.local.dev
7  export GLEANER_WEB_DOMAIN=web.local.dev
8  export GLEANER_WEB2_DOMAIN=web2.local.dev
9
10 # Object store keys
11 export MINIO_ACCESS_KEY=worldsbestaccesskey
12 export MINIO_SECRET_KEY=worldsbestsecretkey
13
14 # local data volumes
15 export GLEANER_BASE=/tmp/gleaner/
16 export GLEANER_TRAEFIK=${GLEANER_BASE}/config
17 export GLEANER_OBJECTS=${GLEANER_BASE}/datavol/s3
18 export GLEANER_GRAPH=${GLEANER_BASE}/datavol/graph
19

```

– Break down the compose file here

```

1  version: '3'
2
3  # ${GLEANER_ADMIN_DOMAIN}
4  # ${GLEANER_OSS_DOMAIN}
5  # ${GLEANER_GRAPH_DOMAIN}
6  # ${GLEANER_WEB_DOMAIN}
7  # ${GLEANER_WEB2_DOMAIN}

```

(continues on next page)

(continued from previous page)

```

8  # ${MINIO_ACCESS_KEY}
9  # ${MINIO_SECRET_KEY}
10 #
11 # ${GLEANER_TRAEFIK}
12 # ${GLEANER_OBJECTS}
13 # ${GLEANER_GRAPH}
14
15 services:
16   triplestore:
17     image: nower/blazegraph
18     environment:
19       JAVA_XMS: 2g
20       JAVA_XMX: 8g
21       JAVA_OPTS: -Xmx6g -Xms2g --XX:+UseG1GC
22     ports:
23       - 9999:9999
24     labels:
25       - "traefik.enable=true"
26       - "traefik.http.routers.triplestore.entrypoints=http"
27       - "traefik.http.routers.triplestore.rule=Host(`${GLEANER_GRAPH_DOMAIN}`)"
28       - "traefik.http.middlewares.triplestore-https-redirect.redirectscheme.
↪scheme=https"
29       - "traefik.http.routers.triplestore.middlewares=triplestore-https-redirect"
30       - "traefik.http.routers.triplestore-secure.entrypoints=https"
31       - "traefik.http.routers.triplestore-secure.rule=Host(`${GLEANER_GRAPH_DOMAIN}`)"
32       - "traefik.http.routers.triplestore-secure.tls=true"
33       - "traefik.http.routers.triplestore-secure.tls.certresolver=http"
34       - "traefik.http.routers.triplestore-secure.service=triplestore"
35       - "traefik.http.middlewares.triplestore-secure.headers.
↪accesscontrolallowmethods=GET,OPTIONS,PUT,POST"
36       - "traefik.http.middlewares.triplestore-secure.headers.
↪accesscontrolalloworigin=*"
37       - "traefik.http.middlewares.triplestore-secure.headers.accesscontrolmaxage=200"
38       - "traefik.http.middlewares.triplestore-secure.headers.addvaryheader=true"
39       - "traefik.http.middlewares.triplestore-secure.headers.
↪accesscontrolallowcredentials=true"
40       - "traefik.http.middlewares.triplestore-secure.headers.
↪accesscontrolallowheaders=Authorization,Origin,Content-Type,Accept"
41       - "traefik.http.middlewares.triplestore-secure.headers.customresponseheaders.
↪Access-Control-Allow-Headers=Authorization,Origin,Content-Type,Accept"
42       - "traefik.http.routers.triplestore-secure.middlewares=triplestore-secure@docker
↪"
43       - "traefik.http.services.triplestore.loadbalancer.server.port=9999"
44       - "traefik.docker.network=traefik_default"
45     volumes:
46       - ${GLEANER_GRAPH}:/var/lib/blazegraph
47     networks:
48       - traefik_default
49
50   s3system:
51     image: minio/minio:latest
52     ports:
53       - 9000:9000
54     labels:
55       - "traefik.enable=true"
56       - "traefik.http.routers.s3system.entrypoints=http"
57       - "traefik.http.routers.s3system.rule=Host(`${GLEANER_OSS_DOMAIN}`)"

```

(continues on next page)

(continued from previous page)

```

58     - "traefik.http.middlewares.s3system-https-redirect.redirectscheme.scheme=https"
59     - "traefik.http.routers.s3system.middlewares=s3system-https-redirect"
60     - "traefik.http.routers.s3system-secure.entrypoints=https"
61     - "traefik.http.routers.s3system-secure.rule=Host(`${GLEANER_OSS_DOMAIN}`)"
62     - "traefik.http.routers.s3system-secure.tls=true"
63     - "traefik.http.routers.s3system-secure.tls.certresolver=http"
64     - "traefik.http.routers.s3system-secure.service=s3system"
65     - "traefik.http.services.s3system.loadbalancer.server.port=9000"
66     - "traefik.docker.network=traefik_default"
67 volumes:
68     - ${GLEANER_OBJECTS}:/data
69 environment:
70     - MINIO_ACCESS_KEY=${MINIO_ACCESS_KEY}
71     - MINIO_SECRET_KEY=${MINIO_SECRET_KEY}
72 networks:
73     - traefik_default
74 command: ["server", "/data"]
75
76 features:
77     image: fils/grow-general:latest
78     ports:
79         - 8080:8080
80     environment:
81         - S3ADDRESS=s3system:9000
82         - S3BUCKET=sites
83         - S3PREFIX=domain
84         - DOMAIN=https://${GLEANER_WEB_DOMAIN}/
85         - S3KEY=${MINIO_ACCESS_KEY}
86         - S3SECRET=${MINIO_SECRET_KEY}
87     labels:
88         - "traefik.enable=true"
89         - "traefik.http.routers.features.entrypoints=http"
90         - "traefik.http.routers.features.rule=Host(`${GLEANER_WEB_DOMAIN}`, `${GLEANER_
↪WEB2_DOMAIN}`)"
91         - "traefik.http.middlewares.features-https-redirect.redirectscheme.scheme=https"
92         - "traefik.http.routers.features.middlewares=features-https-redirect"
93         - "traefik.http.routers.features-secure.entrypoints=https"
94         - "traefik.http.routers.features-secure.rule=Host(`${GLEANER_WEB_DOMAIN}`, `${
↪GLEANER_WEB2_DOMAIN}`)"
95         - "traefik.http.routers.features-secure.tls=true"
96         - "traefik.http.routers.features-secure.tls.certresolver=http"
97         - "traefik.http.routers.features-secure.service=features"
98         - "traefik.http.services.features.loadbalancer.server.port=8080"
99         - "traefik.docker.network=traefik_default"
100        - "traefik.http.middlewares.features.headers.accesscontrolallowmethods=GET,
↪OPTIONS,PUT,POST"
101        - "traefik.http.middlewares.features.headers.accesscontrolalloworigin=*"
102        - "traefik.http.middlewares.features.headers.accesscontrolmaxage=100"
103        - "traefik.http.middlewares.features.headers.addvaryheader=true"
104        - "traefik.http.middlewares.features-secure.headers.accesscontrolallowheaders=*"
105        - "traefik.http.middlewares.features-secure.headers.customresponseheaders.
↪Access-Control-Allow-Headers=*"
106    networks:
107        - traefik_default
108
109 networks:
110     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)

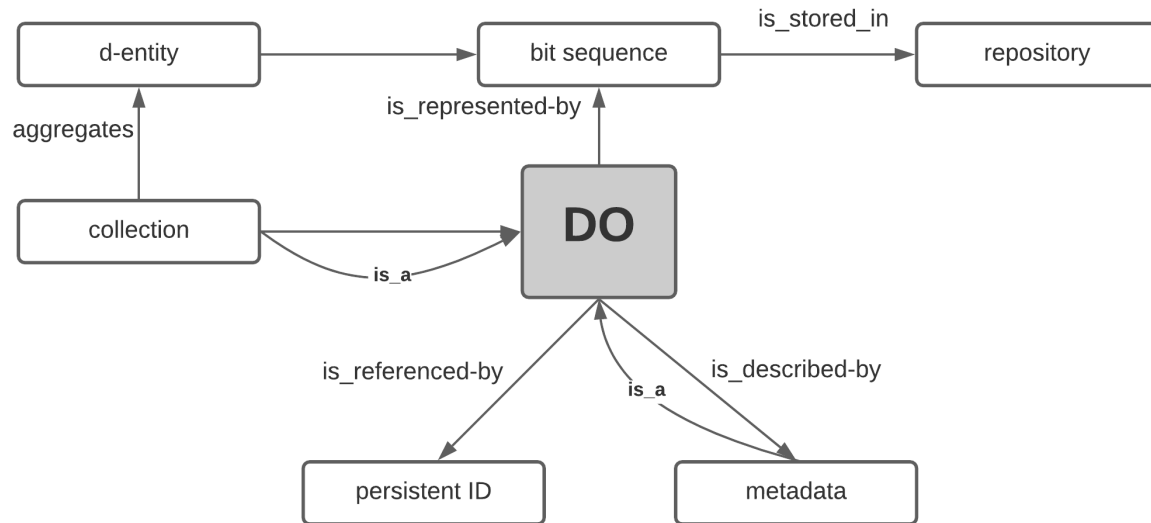


Figure 1: Research Data Alliance Digital Object Cloud Pattern

INTERFACES

23.1 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

23.2 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 augment 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.

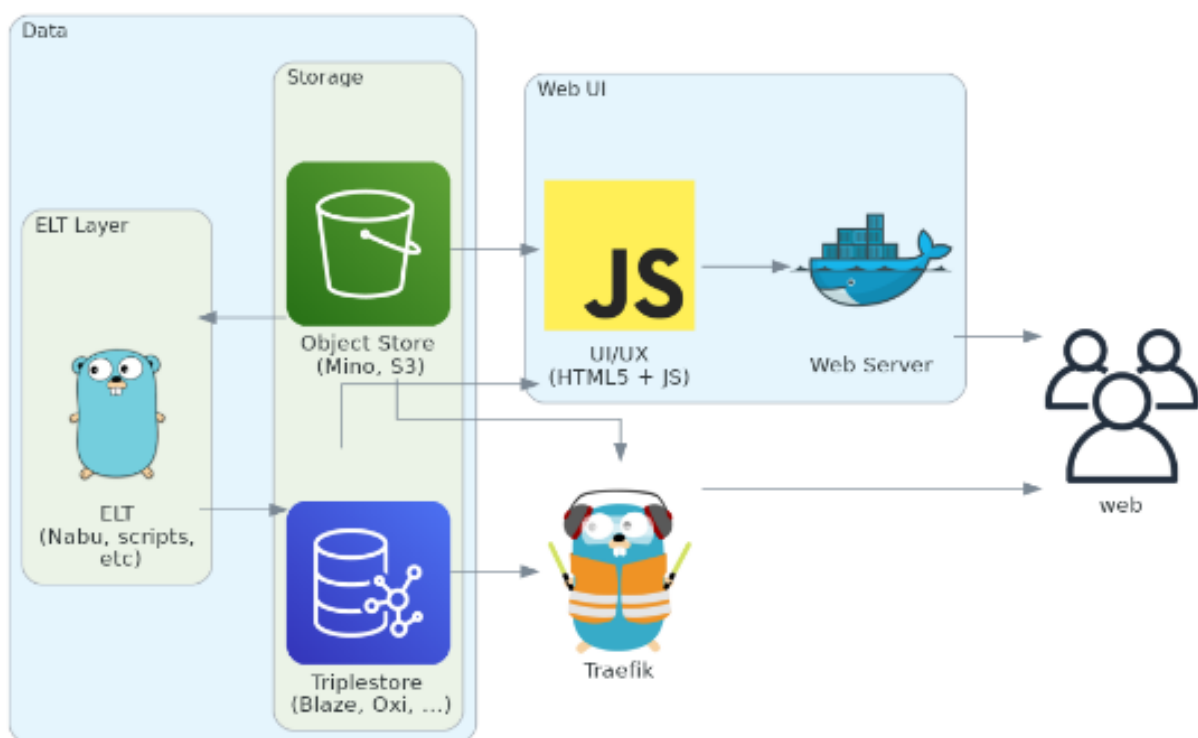


Fig. 1: Gleaner Optional Web UI

GRAPH FIRST APPROACH

24.1 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.

24.2 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](https://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:DataCatalogue 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.

24.3 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.

24.4 Publishing and referencing

24.5 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

25.1 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.

25.2 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.

```
1 {
2   "@context": {
3     "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
4     "prov": "http://www.w3.org/ns/prov#",
5     "rdfs": "http://www.w3.org/2000/01/rdf-schema#"
6   },
7   "@graph": [
8     {
9       "@id": "https://www.re3data.org/repository/obis",
10      "@type": "prov:Organization",
11      "rdf:name": "Ocean Biodiversity Information System",
12      "rdfs:seeAlso": "https://obis.org"
13    },
14    {
15      "@id": "https://obis.org/dataset/9381239f-3d64-48b4-80c9-b9ebb674edc2",
16      "@type": "prov:Entity",
17      "prov:wasAttributedTo": {
18        "@id": "https://www.re3data.org/repository/obis"
19      },
20      "prov:value": "https://obis.org/dataset/9381239f-3d64-48b4-80c9-
21      ↪b9ebb674edc2"
22    },
23    {
24      "@id": "https://gleaner.io/id/collection/
25      ↪7c1eaa1aaed95861330109026c42e57a31ecae55",
```

(continues on next page)

(continued from previous page)

```

24         "@type": "prov:Collection",
25         "prov:hadMember": {
26             "@id": "https://obis.org/dataset/9381239f-3d64-48b4-80c9-b9ebb674edc2"
27         }
28     },
29     {
30         "@id": "urn:gleaner:milled:obis:7c1eaa1aaed95861330109026c42e57a31ecae55",
31         "@type": "prov:Entity",
32         "prov:value": "7c1eaa1aaed95861330109026c42e57a31ecae55.jsonld"
33     },
34     {
35         "@id": "https://gleaner.io/id/run/7c1eaa1aaed95861330109026c42e57a31ecae55
36     ↪",
37         "@type": "prov:Activity",
38         "prov:endedAtTime": {
39             "@value": "2021-04-20",
40             "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
41         },
42         "prov:generated": {
43             "@id":
44 ↪"urn:gleaner:milled:obis:7c1eaa1aaed95861330109026c42e57a31ecae55"
45         },
46         "prov:used": {
47             "@id": "https://gleaner.io/id/collection/
48 ↪7c1eaa1aaed95861330109026c42e57a31ecae55"
49         }
50     }
51 ]
52 }

```

```

{
  "@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"
    }
}
}
}

```

25.3 Nano Prov

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

```

1 {
2   "@context": {
3     "gleaner": "https://voc.gleaner.io/id/",
4     "np": "http://www.nanopub.org/nschema#",
5     "prov": "http://www.w3.org/ns/prov#",
6     "xsd": "http://www.w3.org/2001/XMLSchema#"
7   },
8   "@set": [
9     {
10      "@id": "gleaner:nanopub/XID",
11      "@type": "np:NanoPublication",
12      "np:hasAssertion": {
13        "@id": "gleaner:nanopub/XID#assertion"
14      },
15      "np:hasProvenance": {
16        "@id": "gleaner:nanopub/XID#provenance"
17      },
18      "np:hasPublicationInfo": {
19        "@id": "gleaner:nanopub/XID#pubInfo"
20      }
21    },
22    {
23      "@id": "gleaner:nanopub/XID#assertion",
24      "@graph": {
25        "@id": "DataSetURI",
26        "@type": "schema:Dataset",
27        "description": "This is where you would put corrections or annotations
28        ",
29        "identifier": [
30          {
31            "@type": "schema:PropertyValue",
32            "name": "GraphSHA",
33            "description": "A SHA256 sha stamp on the harvested data_
34            graph from a URL",
35            "value": "{SHA256 HASH HERE}"
36          },
37          {
38            "@type": "schema:PropertyValue",
39            "name": "ProviderID",
40            "description": "The id provided with the data graph by the_
41            provider",

```

(continues on next page)

(continued from previous page)

```

39         "value": "{{re3 or URL noted in config}}"
40     },
41     {
42         "@type": "schema:PropertyValue",
43         "name": "URL",
44         "description": "The URL harvested by gleaner",
45         "value": "{{The URL the JSON-LD came from}}"
46     }
47 ]
48 }
49 },
50 {
51     "@id": "gleaner:nanopub/XID#provenance",
52     "@graph": {
53         "@id": "URIforprovondataset",
54         "prov:wasGeneratedAtTime": {
55             "@value": "dateDone",
56             "@type": "xsd:dateTime"
57         },
58         "prov:wasDerivedFrom": {
59             "@id": "IDHERE"
60         },
61         "prov:wasAttributedTo": {
62             "@id": "IDHERE"
63         }
64     }
65 },
66 {
67     "@id": "gleaner:nanopub/XID#pubInfo",
68     "@graph": {
69         "@id": "IDHERE",
70         "prov:wasAttributedTo": {
71             "@id": "gleaner:tool/gleaner"
72         },
73         "prov:generatedAtTime": {
74             "@value": "2019-10-23T14:38:00Z",
75             "@type": "xsd:dateTime"
76         }
77     }
78 }
79 ]
80 }

```

```
<graphviz.dot.Digraph at 0x7fece09b02e0>
```

25.4 Refs

Nanopubs Guidance

ALTERNATIVES

26.1 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

TOOLING

27.1 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

27.1.1 OpenRefine

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

27.1.2 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.

27.2 On-line tooling

[Schema.org Validator](#)

[json-ld playground](#)

[SHACL Playground](#)

[json-lint](#)

[f-ujj](#)

27.3 Dev

[json-ld](#)

[f-ujj dev](#)

[jq](#)

[jello \(python\)](#) and [Practical JSON](#) at the command line using [jello](#)

[Python Extruct](#)

27.4 OpenRefine

27.4.1 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.

27.4.2 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)}}
}
```

27.4.3 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 exporter](#) to 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 column, 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)}}
}

```

27.5 OIH Notebooks

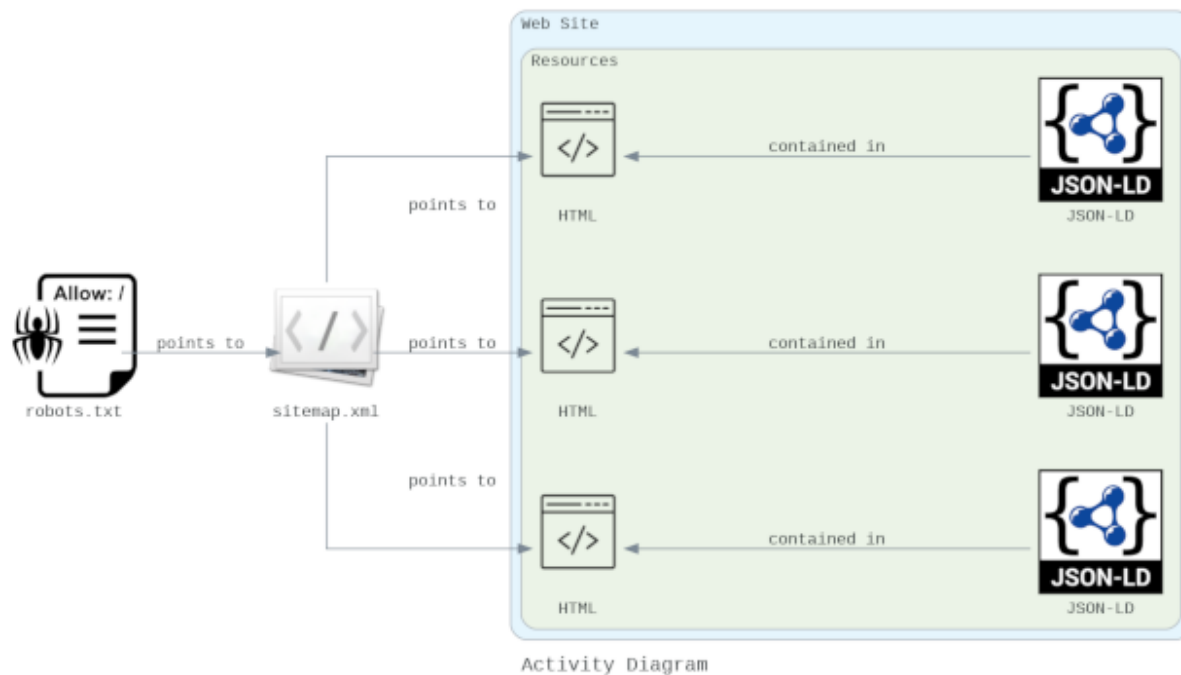
27.5.1 About

This section contains a collection of 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>). They will likely run on Google Colab or Binder but additional installs may be required. We will attempt to document all requirements in the notebooks and the instructions.

27.5.2 Indexing Concepts

Review

Before we starting pulling data lets revist the publishing overview presented in *Publisher* section.



Briefly, a sitemap is made available that points the resources we will be indexing. This can optionally be in a robots.txt file as well. This sitemap will provide a URL for each resource we will be indexing. It is fine if it provides more too, those resources simply won't express any JSON-LD content. You can also have multiple sitemaps or ones specifically focused on the resources to index.

Each URL or page represented in the index are then accessed and parsed for the JSON-LD content.

Command Line Tooling

We can start exploring the indexing of data on the command line. Here we will use the `curl` command which should be installed on all Mac OS X and Linux systems and should be found on the Linux Subsystem for Windows.

This will give us a low level feel for what is going on.

We will start by exploring a sitemap.

```
curl -s https://samples.earth/sitemap0.xml
```

We can parse out the URLs from the sitemap with the use of the UNIX `grep` command

```
curl -s https://samples.earth/sitemap0.xml | grep -oP '<loc>\K[^\<]*'
```

In doing this we see that our sitemap is really just a feed of URLs. The sitemap provides us with the ability to add some extra information for our URLs. It also provides a machine readable XML format we can work with. There are many libraries for working with XML and several for working with the sitemap data model in XML as well.

Now lets pull down the URL resource and parse out the JSON-LD we find in the `<script>` tag of type `application/ld+json`.

Note, it is possible that there are many of these tags and also that this tag might be placed in by Javascript which means we would not see it here. We will talk more about this as we explore. For now we will just look for the first one and a static example, which we know this to be.

```
curl -s --header "Accept: text/html" https://samples.earth/id/documents/
c1pnht3h2h44frv6igfg | sed -n '/<script type="application\/ld+json">/,/<\/script>/
p'
```

Let's get rid of the `<script>` tags and then parse the JSON-LD.

```
curl -s --header "Accept: text/html" https://samples.earth/id/documents/
c1pnht3h2h44frv6igfg | sed -n '/<script type="application\/ld+json">/,/<\/script>/
p' | sed 's/<\/script>/' | sed 's/<script type="application\/ld+json">/'
```

TODO WORK ON THIS At this point we could copy this JOSN-LD and visit something like the JSON-LD playground. We could also visit the Structure Data Linter. We can then play a bit with the JSON-LD there.

Now, lets pass this throgh another app. This is the `jsonld.js` app. A Javascript app and library that can be found on GitHub at [digitalbazaar/jsonld.js](https://github.com/digitalbazaar/jsonld.js). There are many similar libraries, so you can feel free to try out others.

```
curl -s --header "Accept: text/html" https://samples.earth/id/documents/
c1pnht3h2h44frv6igfg | sed -n '/<script type="application\/ld+json">/,/<\/script>/
p' | sed 's/<\/script>/' | sed 's/<script type="application\/ld+json">/' |
jsonld format -q
```

If all goes well we should see the following output.

```
<https://samples.earth/id/documents/c1pnht3h2h44frv6igfg> <http://www.w3.org/1999/02/
22-rdf-syntax-ns#type> <https://schema.org/Dataset> .
<https://samples.earth/id/documents/c1pnht3h2h44frv6igfg> <https://schema.org/
description> "of data assignments." .
<https://samples.earth/id/documents/c1pnht3h2h44frv6igfg> <https://schema.org/
distribution> _:b0 .
<https://samples.earth/id/documents/c1pnht3h2h44frv6igfg> <https://schema.org/
maintainer> <https://samples.earth> .
<https://samples.earth/id/documents/c1pnht3h2h44frv6igfg> <https://schema.org/name>
"Fake: technical and" .
<https://samples.earth> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://
schema.org/Organization> .
<https://samples.earth> <https://schema.org/description> "DEMO SITE: fake data for
testing" .
_:b0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://schema.org/
DataDownload> .
_:b0 <https://schema.org/contentUrl> "https://samples.earth/id/documents/
c1pnht3h2h44frv6igfg.tsv" .
_:b0 <https://schema.org/encodingFormat> "text/tab-separated-values" .
```

So, this was just a simple set of command line calls to give us a feel for the process. We have seen a sitemap, the URLs that make it up and pulled back those URLs and parsed the JSON-LD. We then wen ahead and converted the JSON-LD into abother RDF represenatikon (triples) that make loading into a graph database easier.

We could easily take these commands and roll them in a simple bash script. This might not be a production level approach, but it's a good exercise and a good way to get started. We will explore more advanced ways of doing this later.

Python

Let's look at how this might be done in Python. Python is a very popular language and has many solid libraries for working with JSON-LD. Again, we will use this more to get a feel for the process.

27.5.3 Imports and defs

```
import json

import requests
from bs4 import BeautifulSoup

def get_ld_json(url: str) -> dict:
    parser = "html.parser"
    req = requests.get(url)
    soup = BeautifulSoup(req.text, parser)
    return json.loads("".join(soup.find("script", {"type": "application/ld+json"}).
    contents))
```

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

def show_graph(url, size=10):
```

(continues on next page)

(continued from previous page)

```

"""
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

```

27.5.4 Parse a search result

Let's do a search at the OIH test web site and then see if we can work with some of the results there.

```
ld = get_ld_json("https://obis.org/dataset/46005357-02b8-4f17-b028-065bfc1cd384")
```

```
json_formatted_str = json.dumps(ld, indent=2)
print(json_formatted_str)
```

```

{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@type": "Dataset",
  "name": "Coral Reef Evaluation and Monitoring Project Dry Tortugas 1999",
  "description": "The purpose of the Coral Reef Evaluation and Monitoring Project
(CREMP) is to monitor the status and trends of selected reefs in the Florida Keys
National Marine Sanctuary(FKNMS). CREMP assessments have been conducted annually at
fixed sites since 1996 and data collected provides information on the temporal
changes in benthic cover and diversity of stony corals and associated marine flora
and fauna. The core field methods continue to be underwater videography and timed
coral species inventories. Findings presented in this report include data from 109
stations at 37 sites sampled from 1996 through 2008 in the Florida Keys and 1999
through 2008 in the Dry Tortugas. The report describes the annual differences
(between 2007 and 2008) in the percent cover of major benthic taxa (stony corals,
octocorals, sponges, and macroalgae), mean coral species richness and the incidence
of stony coral conditions. Additionally, it examines the long-term trends of the
major benthic taxa, five coral complex, Montastraea cavernosa, Colpophyllia natans,
Siderastrea siderea, and Porites astreoides) and the clionaid sponge, Cliona
delitrix. It is one of the longest running coral reef monitoring projects in south
Florida and has been extremely important in documenting the temporal changes that
have occurred in recent years",

```

(continues on next page)

(continued from previous page)

```

"url": "https://obis.org/dataset/46005357-02b8-4f17-b028-065bfc1cd384",
"sameAs": [
  "https://www1.usgs.gov/obis-usa/ipt/resource?r=crempdrytortugas1999"
],
"license": "To the extent possible under law, the publisher has waived all rights_
to these data and has dedicated them to the Public Domain (CC0 1.0)",
"citation": null,
"version": "2020-10-07T22:52:17.000Z",
"keywords": [
  "Samplingevent",
  "about",
  "absence",
  "accepted",
  "accepted_aphia_database_identification_number",
  "accepted_authorship_information",
  "accepted_name_usage",
  "acceptedNameAuthorship",
  "acceptedNameUsage",
  "acceptedNameUsageID",
  "aphia",
  "aphia_database_identification_number",
  "area",
  "array",
  "array-data",
  "associatedreferences",
  "authorship",
  "authorship_information",
  "basis",
  "basis of record",
  "basisOfRecord",
  "biogeographic",
  "biology",
  "class",
  "commission",
  "common",
  "comprehensive",
  "conservation",
  "coral",
  "coral monitoring",
  "coverage",
  "cremp",
  "data",
  "data_set_identification",
  "data_set_name",
  "database",
  "dataset",
  "datasetName",
  "date",
  "dry",
  "dry tortugas",
  "evaluation",
  "event",
  "eventdate",
  "eventID",
  "family",
  "fish",
  "florida",

```

(continues on next page)

(continued from previous page)

```
"florida keys",
"fwc",
"fwc-fwri",
"fwri",
"genus",
"gulf of mexico",
"habitatid",
"hierarchy",
"identification",
"identifier",
"information",
"institute",
"kingdom",
"large",
"latitude",
"longitude",
"marine",
"meters",
"monitoring",
"name",
"number",
"obis",
"occurrence",
"occurrence_identification",
"occurrenceID",
"occurrenceStatus",
"ocean",
"order",
"organism",
"organism_per_sample_area",
"organismQuantity",
"organismQuantityType",
"per",
"percent",
"percent_coverage",
"phylum",
"presence",
"project",
"rank",
"record",
"recorded",
"recordedBy",
"reef",
"register",
"research",
"revisited",
"sample",
"samples",
"scientific",
"scientific_name",
"scientificname",
"scientificNameAuthorship",
"scientificnameid",
"set",
"sitcode",
"siteid",
"species",
```

(continues on next page)

(continued from previous page)

```

    "species_name",
    "specificEpithet",
    "square",
    "statement",
    "station",
    "status",
    "stewardship",
    "subregionid",
    "system",
    "taxon",
    "taxon_rank",
    "taxon_status",
    "taxonomic",
    "taxonomic_status",
    "taxonomicStatus",
    "taxonomy",
    "taxonRank",
    "time",
    "tortugas",
    "unaccepted",
    "usage",
    "v2.3",
    "value",
    "vernacular",
    "vernacular_name",
    "vernacularName",
    "wildlife",
    "world",
    "worms",
    "year"
  ],
  "variableMeasured": [],
  "includedInDataCatalog": {
    "@id": "https://obis.org",
    "@type": "DataCatalog",
    "url": "https://obis.org"
  },
  "temporalCoverage": "1999/1999",
  "distribution": {
    "@type": "DataDownload",
    "contentUrl": "https://www1.usgs.gov/obis-usa/ipt/archive.do?
↪r=crempdrytortugas1999",
    "encodingFormat": "application/zip"
  },
  "spatialCoverage": {
    "@type": "Place",
    "geo": {
      "@type": "GeoShape",
      "polygon": "-83.0022 24.6117,-83.0022 24.6993,-82.8702 24.6993,-82.8702 24.6117,
↪-83.0022 24.6117"
    },
    "additionalProperty": {
      "@type": "PropertyValue",
      "propertyID": "http://dbpedia.org/resource/Spatial_reference_system",
      "value": "http://www.w3.org/2003/01/geo/wgs84_pos#lat_long"
    }
  }
},

```

(continues on next page)

(continued from previous page)

```

"provider": [
  {
    "@id": "https://oceanexpert.org/institution/5852",
    "@type": "Organization",
    "legalName": "Texas A&M University, College Station \u2013 Department of Oceanography",
    "name": "Texas A&M University, College Station \u2013 Department of Oceanography",
    "url": "https://oceanexpert.org/institution/5852"
  },
  {
    "@id": "https://oceanexpert.org/institution/6238",
    "@type": "Organization",
    "legalName": "Florida Fish and Wildlife Conservation Commission, Fish and Wildlife Research Institute",
    "name": "Florida Fish and Wildlife Conservation Commission, Fish and Wildlife Research Institute",
    "url": "https://oceanexpert.org/institution/6238"
  },
  {
    "@id": "https://oceanexpert.org/institution/12976",
    "@type": "Organization",
    "legalName": "U.S. Geological Survey HQ",
    "name": "U.S. Geological Survey HQ",
    "url": "https://oceanexpert.org/institution/12976"
  }
]

```

That's big

There is a lot here, especially in the keywords section. That's great, this will give us a lot to work with the graph. For this demo though it might be nice to parse it down a bit. We can take advantage of a feature of JSON-LD called "framing". Here will make a JSON-LD frame that allows us to parse out only those elements of the document we want to work with. Framing is very powerful, but we will work with a simple frame for now. One like this:

```

{
  "@context": {"@vocab": "https://schema.org/"},
  "@explicit": "true",
  "@type": "Dataset",
  "name": "",
  "description": "",
  "url": "",
  "sameAs": ""
}

```

We will run this frame then look at the results and do a quick visualization using the defined function we placed at the beginning of this notebook.

```

from pyld import jsonld
import json

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

```

(continues on next page)

(continued from previous page)

```

    "@type": "Dataset",
    "name": "",
    "description": "",
    "url": "",
    "sameAs": "",
}

framed = jsonld.frame(ld, frame)

json_formatted_str = json.dumps(framed, indent=2)
print(json_formatted_str)

show_graph(framed)

```

```

{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@type": "Dataset",
  "description": "The purpose of the Coral Reef Evaluation and Monitoring Project_
↳ (CREMP) is to monitor the status and trends of selected reefs in the Florida Keys_
↳ National Marine Sanctuary (FKNMS). CREMP assessments have been conducted annually at_
↳ fixed sites since 1996 and data collected provides information on the temporal_
↳ changes in benthic cover and diversity of stony corals and associated marine flora_
↳ and fauna. The core field methods continue to be underwater videography and timed_
↳ coral species inventories. Findings presented in this report include data from 109_
↳ stations at 37 sites sampled from 1996 through 2008 in the Florida Keys and 1999_
↳ through 2008 in the Dry Tortugas. The report describes the annual differences_
↳ (between 2007 and 2008) in the percent cover of major benthic taxa (stony corals,_
↳ octocorals, sponges, and macroalgae), mean coral species richness and the incidence_
↳ of stony coral conditions. Additionally, it examines the long-term trends of the_
↳ major benthic taxa, five coral complex, Montastraea cavernosa, Colpophyllia natans,_
↳ Siderastrea siderea, and Porites astreoides) and the clionaid sponge, Cliona_
↳ delitrix. It is one of the longest running coral reef monitoring projects in south_
↳ Florida and has been extremely important in documenting the temporal changes that_
↳ have occurred in recent years",
  "name": "Coral Reef Evaluation and Monitoring Project Dry Tortugas 1999",
  "sameAs": "https://www1.usgs.gov/obis-usa/ipt/resource?r=crempdrytortugas1999",
  "url": "https://obis.org/dataset/46005357-02b8-4f17-b028-065bfc1cd384"
}

```

```
<graphviz.dot.Digraph at 0x7f2674380b20>
```

27.5.5 Conclusion

Using these approaches you could explore data from other sources you know are publishing JSON-LD. You could improve either of the bash script or Python code to loop on the resources and store the results in files or load them directly into a triples store / graph database.

As we continue, we will move on to the Gleaner package that is used by Ocean InfoHub and see some of the edge cases that can be addressed with it and how it fits into a more automated process.

27.5.6 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 -q pyshacl
!pip install -q 'PyLD>=2.0.3'
!pip install -q flatten_json
!pip install -q 'fsspec>=0.3.3'
!pip install -q s3fs
!pip -q install SPARQLWrapper
# !pip install -q 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 as kg

oih = "https://graph.collaborium.io/blazegraph/namespace/oih/sparql"
oihdev = "https://graph.collaborium.io/blazegraph/namespace/oihdev/sparql"
oihad = "https://graph.collaborium.io/blazegraph/namespace/aquadocs/sparql"
```

Support 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)
```

Queries

What follows is a set of queries designed to provide a feel for the OIH graph

Simple Count

How many triples are there?

```
rq_count = """SELECT (COUNT(*) as ?Triples)
WHERE
{
    { ?s ?p ?o }
}
"""
```

```
dfsc = get_sparql_dataframe(oih, rq_count)
dfsc.head()
```

```
  Triples
0  159766
```

Predicate Counts

This gives an overview of unique predicates that connect a subject to an object. This gives us both an idea of the properties we are using on things and count of their usage.

```
rq_pcount = """SELECT ?p (COUNT(?p) as ?pCount)
WHERE
{
    ?s ?p ?o .
}
GROUP BY ?p
"""
```

```
dfc = get_sparql_dataframe(oih, rq_pcount)
dfc['pCount'] = dfc["pCount"].astype(int) # convert count to int
# dfc.set_index('p', inplace=True)
```

```
dfc_sorted = dfc.sort_values('pCount', ascending=False)
dfc_sorted.head(10)
```

	p	pCount
33	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	32625
5	https://schema.org/keywords	17359
4	https://schema.org/url	14236
2	https://schema.org/name	12594
17	https://schema.org/provider	5136
0	https://schema.org/description	4895
6	https://schema.org/sameAs	4889
27	https://schema.org/legalName	4823
3	https://schema.org/propertyID	4265
7	https://schema.org/value	4265

```
rcount = len(dfc_sorted)
print(rcount)
```

```
34
```

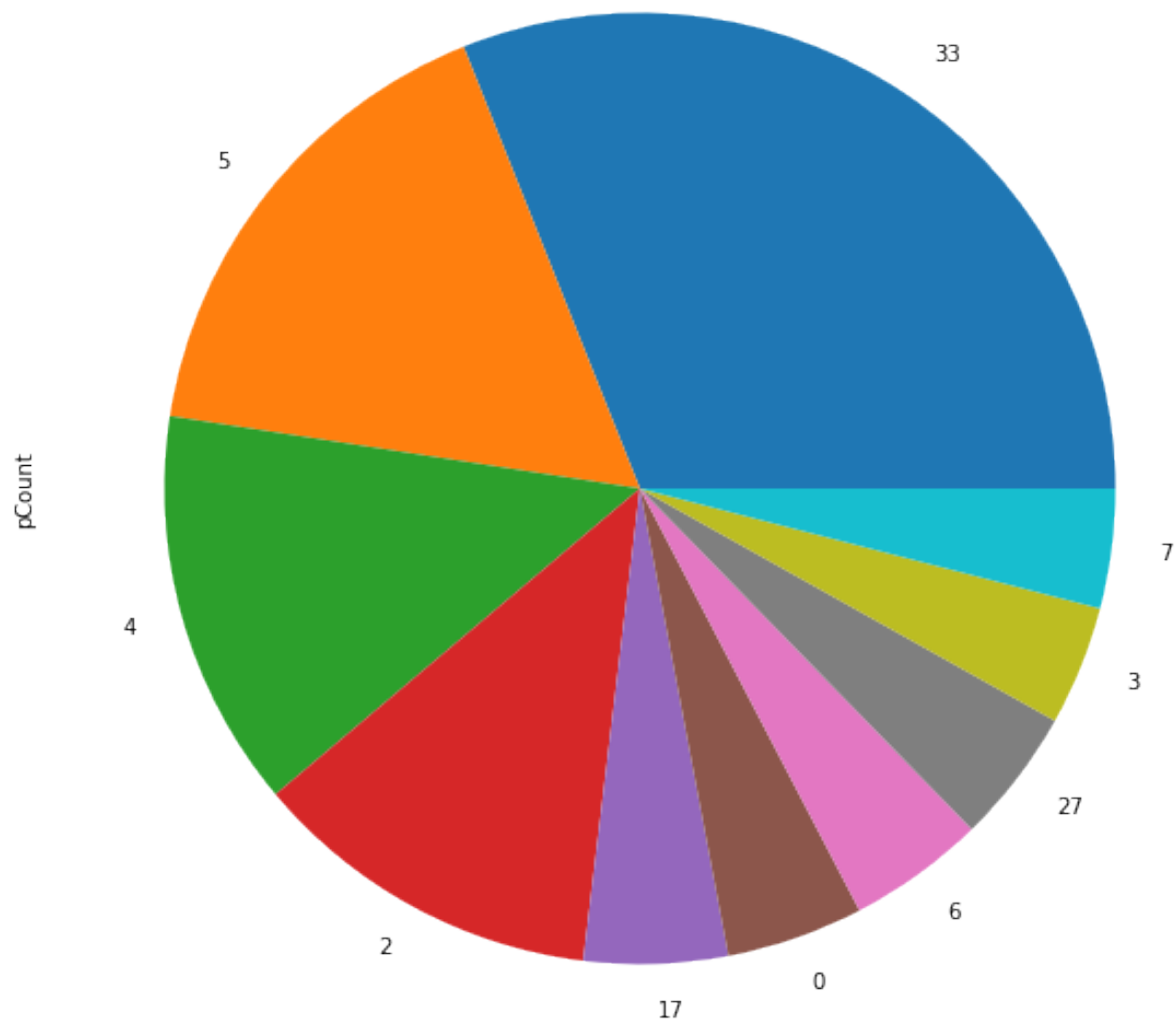
```
ts = dfc_sorted.tail(38) ['pCount'].sum()
print(ts)
```

```
159766
```

```
hs = dfc_sorted.head(10)
hs.append({'p': 'Other', 'pCount': ts}, ignore_index=True)
# hs.set_index('p', inplace=True)
hs.head(15)
```

	p	pCount
33	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	32625
5	https://schema.org/keywords	17359
4	https://schema.org/url	14236
2	https://schema.org/name	12594
17	https://schema.org/provider	5136
0	https://schema.org/description	4895
6	https://schema.org/sameAs	4889
27	https://schema.org/legalName	4823
3	https://schema.org/propertyID	4265
7	https://schema.org/value	4265

```
plot = hs.plot.pie(y='pCount', x='p', legend=False, figsize=(10, 10))
```



OIH Base Query

```
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 ?type ?score ?name ?url ?lit ?description ?headline
WHERE
```

(continues on next page)

(continued from previous page)

```
{
  ?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 . }
  }
}
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...  t670214
1  urn:gleaner:milled:obis:18d1180a74c200d06f9114...  t671476
2  urn:gleaner:milled:obis:24bac898cda34444176ec4...  t673848
3  urn:gleaner:milled:obis:24d453e3a4ea6d1f117e5c...  t673875
4  urn:gleaner:milled:obis:2bf98aa888d856b8706176...  t675633

                                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  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/431f96f7-521c-4182-ae...  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  The purpose of the Coral Reef Evaluation and M...  None
```

OIH Gleaner Query

```
rq_maingl = """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(oihad, rq_maingl)
df.head(5)
```

```

g \
0 urn:gleaner:milled:obis:c18f3c68e05fc5820e5daf...
1 urn:gleaner:milled:obis:4411c25cf985f8253c0a13...
2 urn:gleaner:summoned:aquadocs:bac548e654f7097a...
3 urn:gleaner:summoned:aquadocs:bac548e654f7097a...
4 urn:gleaner:summoned:aquadocs:bac548e654f7097a...

s wat \
0 t3333357 https://www.re3data.org/repository/obis
1 t3323787 https://www.re3data.org/repository/obis
2 oai:aquadocs.org:1834/1987 https://www.re3data.org/repository/aquadocs
3 oai:aquadocs.org:1834/10551 https://www.re3data.org/repository/aquadocs
4 oai:aquadocs.org:1834/12281 https://www.re3data.org/repository/aquadocs

orgname domain \
```

(continues on next page)

(continued from previous page)

0	Ocean Biodiversity Information System	https://obis.org	
1	Ocean Biodiversity Information System	https://obis.org	
2	AquaDocs	https://aquadocs.org	
3	AquaDocs	https://aquadocs.org	
4	AquaDocs	https://aquadocs.org	
	type score \		
0	https://schema.org/Dataset	1.0	
1	https://schema.org/Dataset	1.0	
2	https://schema.org/CreativeWork	1.0	
3	https://schema.org/CreativeWork	1.0	
4	https://schema.org/CreativeWork	1.0	
	name \		
0	Nematoda from Kenya and Zanzibar		
1	Kenyan reef corals sampled in November 1990		
2	Is competition for space between the encrustin...		
3	Study of biotic communities for artificial ree...		
4	Molecular diversity of Symbiodinium spp. withi...		
	url lit \		
0	https://obis.org/dataset/aa9787d6-c4db-4fde-8e...	Coral	
1	https://obis.org/dataset/447b61a2-4a04-4394-af...	Coral	
2	https://www.oceandocs.org/handle/1834/1987	Coral	
3	https://www.oceandocs.org/handle/1834/10551	Coral	
4	https://www.oceandocs.org/handle/1834/12281	Coral	
	description headline		
0	Data on the species and trophic composition of...	None	
1	This dataset contains a taxonomic list of cora...	None	
2	- Temperaturas del agua de mar por encima del...	None	
3	- Persian Gulf waters (Hormuzgan Province) we...	None	
4	- Reef-building coral harbor communities of p...	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 focus 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 .

```

(continues on next page)

(continued from previous page)

```

    ?wat rdfs:seeAlso ?domain
}
GROUP BY ?wat ?orgname ?domain
"""

```

```

rq_prov2 = """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) ?wat ?orgname ?domain
WHERE
{
  graph ?g {
    VALUES (?type) { ( schema:CreativeWork ) ( schema:Map ) ( schema:Person
    ↪) ( schema:Organization ) ( schema:Dataset ) ( schema:Course ) }
    ?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
}

GROUP BY ?wat ?orgname ?domain

"""

```

```

dfp = get_sparql_dataframe(oihad, rq_prov2)
dfp['count'] = dfp["count"].astype(int) # convert count c to int
dfp.set_index('orgname', inplace=True)
dfp.head(10)

```

```

orgname                                count \
Marine Training EU                      944
Ocean Biodiversity Information System    9036
OceanExpert UNESCO/IOC Project Office for IODE    502
Ocean Best Practices                     8223
AquaDocs                                155498

↪      wat \
orgname
Marine Training EU                https://www.re3data.org/repository/
↪marinetraining

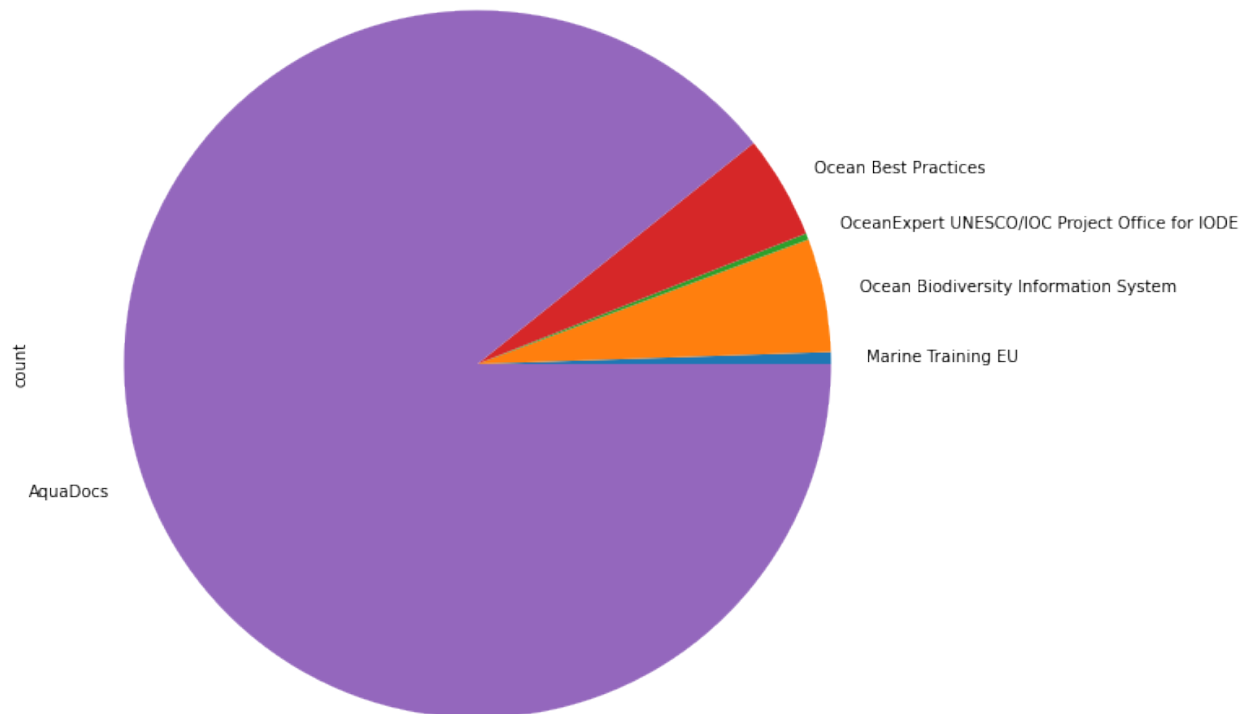
```

(continues on next page)

(continued from previous page)

Ocean Biodiversity Information System	https://www.re3data.org/
↪ repository/obis	
OceanExpert UNESCO/IOC Project Office for IODE	https://www.re3data.org/repository/
↪ oceanexpert	
Ocean Best Practices	https://www.re3data.org/
↪ repository/obps	
AquaDocs	https://www.re3data.org/
↪ repository/aquadocs	
	domain
orgname	
Marine Training EU	https://marinettraining.eu/
Ocean Biodiversity Information System	https://obis.org
OceanExpert UNESCO/IOC Project Office for IODE	https://oceanexpert.org/
Ocean Best Practices	https://oih.oceanbestpractices.org
AquaDocs	https://aquadocs.org

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



```
rq_provide = """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#>
```

(continues on next page)

(continued from previous page)

```
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
"""
```

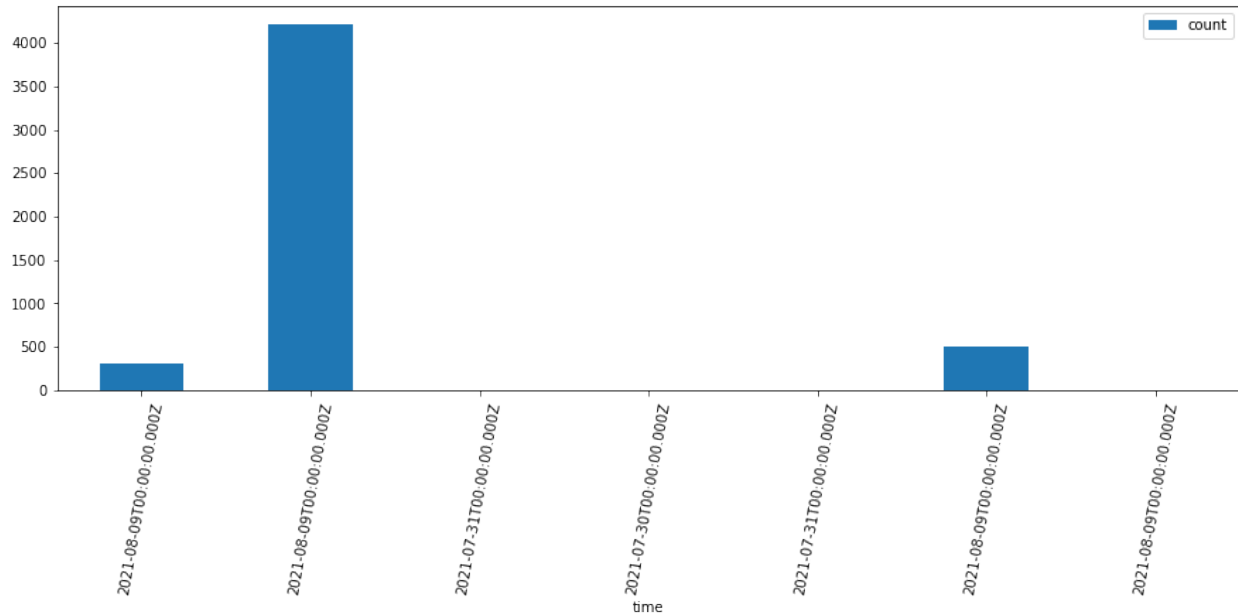
```
dfpd = get_sparql_dataframe(oihad, rq_provddate)
dfpd.head(10)
```

	count	time \	orgname
0	314	2021-08-09T00:00:00.000Z	Marine Training EU
1	4214	2021-08-09T00:00:00.000Z	Ocean Biodiversity Information System
2	1	2021-08-09T00:00:00.000Z	AquaDocs
3	502	2021-08-09T00:00:00.000Z	Ocean Best Practices
4	1	2021-08-09T00:00:00.000Z	OceanExpert UNESCO/IOC Project Office for IODE

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

time	count	orgname
2021-08-09T00:00:00.000Z	314	Marine Training EU
2021-08-09T00:00:00.000Z	4214	Ocean Biodiversity Information System
2021-07-31T00:00:00.000Z	1	AquaDocs
2021-07-30T00:00:00.000Z	2	Ocean Best Practices
2021-07-31T00:00:00.000Z	2	Ocean Best Practices

```
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_provdatalist = ""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#>
```

(continues on next page)

(continued from previous page)

```

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_provdatalist)
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.marinettraining.eu/node/4059
1  https://www.marinettraining.eu/node/4191
2  https://www.marinettraining.eu/node/4347
3  https://www.marinettraining.eu/node/4158
4  https://www.marinettraining.eu/node/3910
5  https://www.marinettraining.eu/node/4332
6  https://www.marinettraining.eu/node/4115
7  https://www.marinettraining.eu/node/4094
8  https://www.marinettraining.eu/node/4343
9  https://www.marinettraining.eu/node/4139

```

```

           memname  \
0  Coastal Benthic Ecology
1  Methods for Implementation of Surveillance Pro...
2  Adaptation Planning
3  Business Development and Innovation (Aquatic R...
4  Machine Vision
5  Arctic Ocean Governance
6  Mathematical Biology
7  Methodology and Sampling in Environmental Mana...

```

(continues on next page)

(continued from previous page)

```

8           Global Fisheries and Seafood
9           Biological Oceanography

                                memdesc
0 Contents\n\nThe course provides a thorough int...
1 General course objectives\n\nThe course aims t...
2 Course Description:\n\nThe course examines cha...
3 General course objectives\n\nThe objective of ...
4 Course content\n\nIntroduction to machine visi...
5 Course Description:\n\nThe course will discuss...
6 General course objectives\n\nThe course is an ...
7 Practical sampling of soil, water and bioindic...
8 Course Description:\n\nThe course offers a det...
9 General course objectives\n\nBiological oceano...

```

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)
    """

```

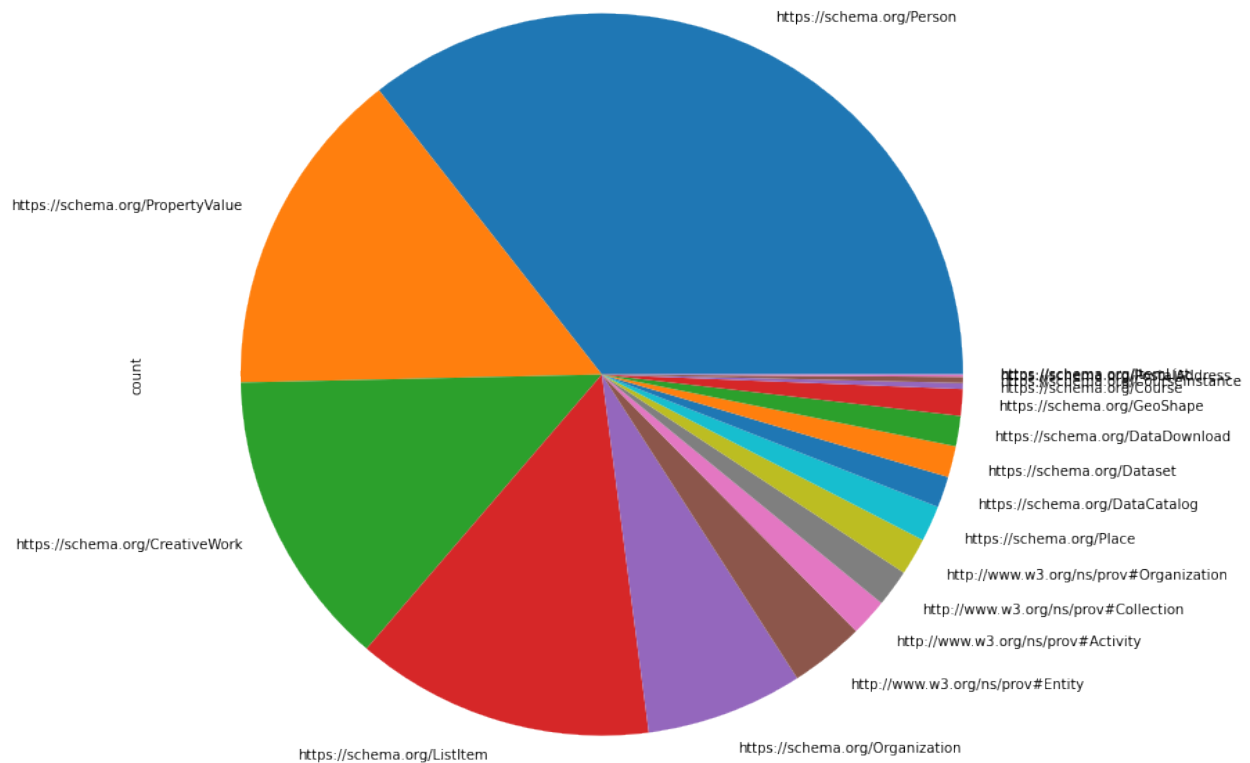
```

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

```

type	count
https://schema.org/Person	107563
https://schema.org/PropertyValue	44658
https://schema.org/CreativeWork	40389
https://schema.org/ListItem	40387
https://schema.org/Organization	21224
http://www.w3.org/ns/prov#Entity	10064
http://www.w3.org/ns/prov#Activity	5032
http://www.w3.org/ns/prov#Collection	5032
http://www.w3.org/ns/prov#Organization	5032
https://schema.org/Place	4848

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



27.5.7 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 datetime import datetime

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

# hide_solution()
```

```
Current Time = 13:16:51
```

```
# 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	47147	0	--:--:-- --:--:-- --:--:-- 47147

```
{
  "@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"
    ↪""}
```

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

(continues on next page)

(continued from previous page)

```

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)

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 0x7efef405bb20>
```

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": "",
}

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 0x7efef405b2b0>
```

Parse out the keywords

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

```
#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 anything 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()
```

	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]

(continues on next page)

(continued from previous page)

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

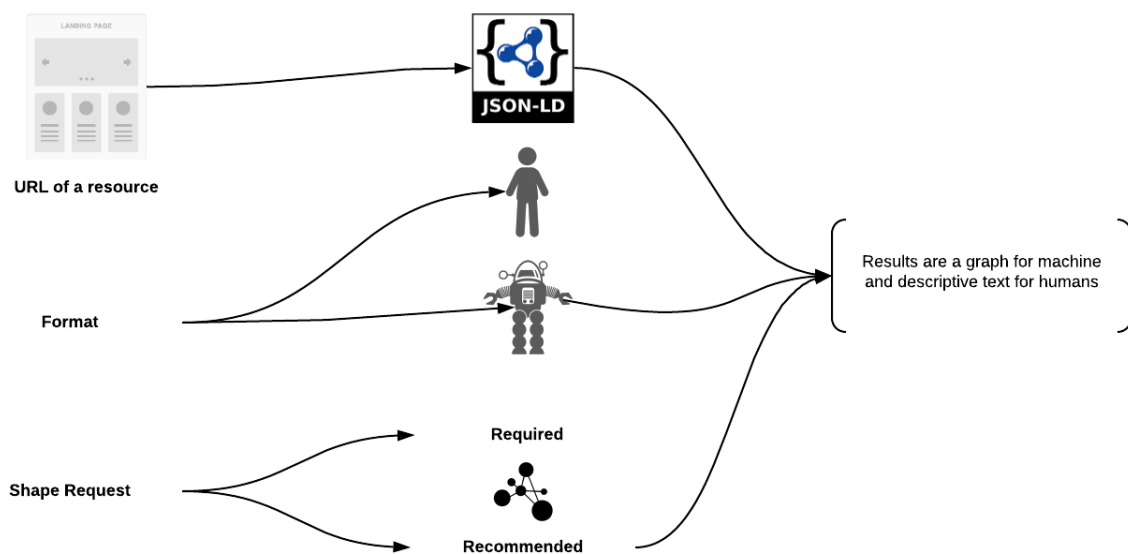
27.5.8 Validation

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 image 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()
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

```

27.5.9 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)

```

(continues on next page)

(continued from previous page)

```
# 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',
```

(continues on next page)

(continued from previous page)

```
'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/",
}

# compact a document according to a particular context
# see: http://json-ld.org/spec/latest/json-ld/#compact-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
```

27.5.10 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
```

(continues on next page)

(continued from previous page)

```
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)
    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 . }
    }
}
```

(continues on next page)

(continued from previous page)

```

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
"""

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

```

(continues on next page)

(continued from previous page)

```

                                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
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 CÃspedes 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

```

(continues on next page)

(continued from previous page)

```
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)
# 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
```

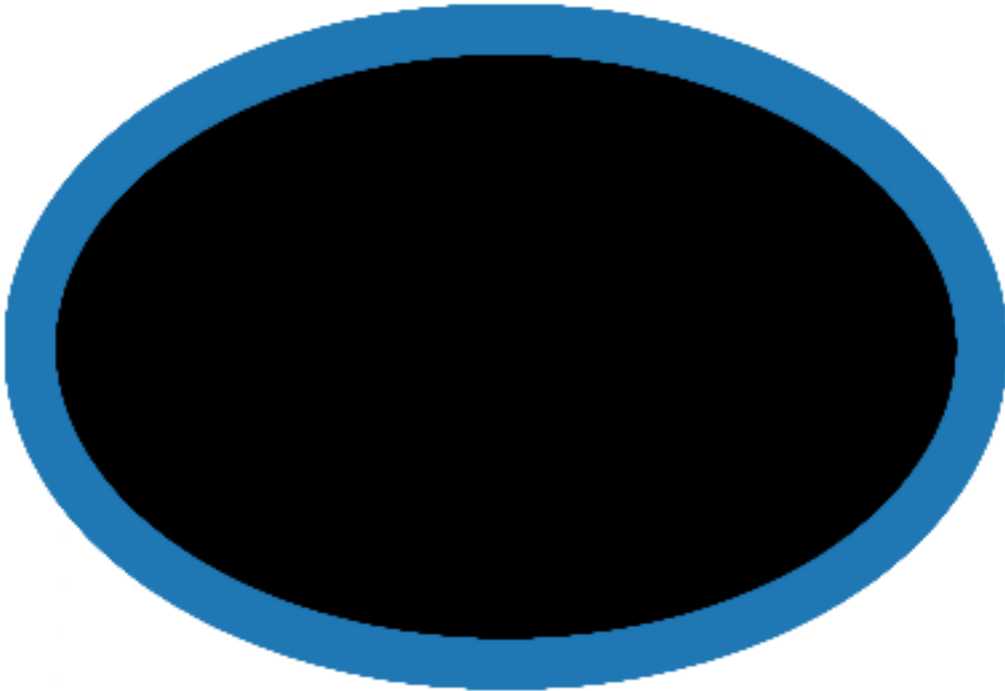
```
0      float64
dtype: object
```

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

```

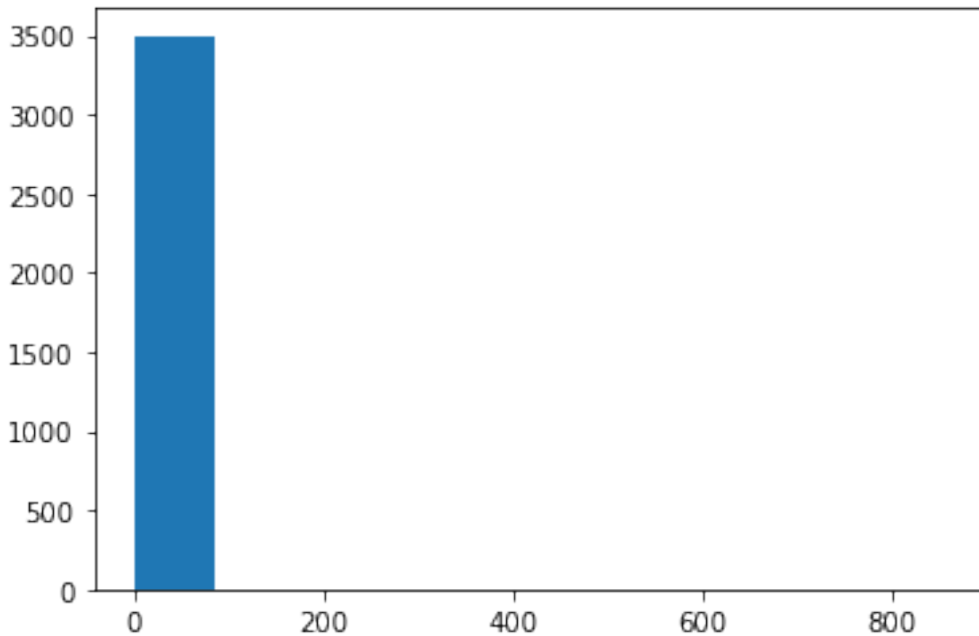
0
https://schema.org/Place      0.058482
https://schema.org/CourseInstance  0.018446
https://schema.org/Course      0.016571
UNESCO/IOC Project Office for IODE Wandelaarka...  0.007863
Russia      0.007497
UNESCO/IOC Project Office for IODE Wandelaarka...  0.006203
Wandelaarkaai 7 8400 Oostende Belgium      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
```



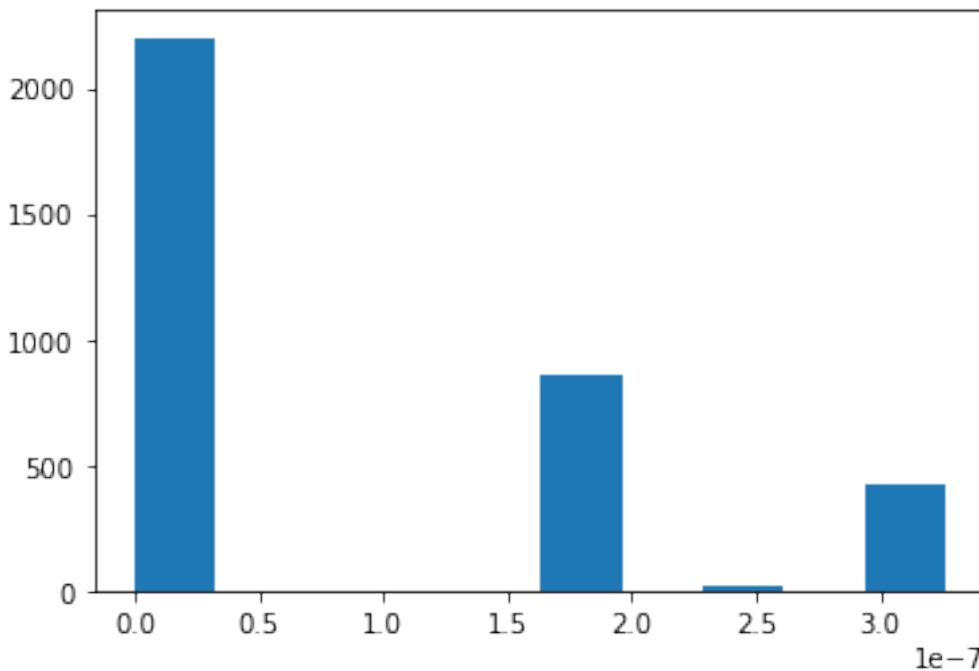
```
plt.hist([v for k,v in nx.degree(G)])
```

```
(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>)
```



27.5.11 SHACL Simple Validation

Ocean Info Hub SHACL validation examples

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 described in <https://www.w3.org/TR/shacl/>.

- SHACL Playground
- kglab SHACL validation with pySHACL

```
%%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
!pip install rdflib-jsonld
!pip install kglab
!pip install numba
!pip install pyld
```

```
import kglab as kg
from rdflib import Graph, plugin
from rdflib.serializer import Serializer
```

```
-----
AttributeError                                Traceback (most recent call last)
<ipython-input-3-138915545c57> in <module>
----> 1 import kglab as kg
      2 from rdflib import Graph, plugin
      3 from rdflib.serializer import Serializer

~/conda/envs/rapids-0.17/lib/python3.7/site-packages/kglab/__init__.py in <module>
      3 # see license https://github.com/DerwenAI/kglab#license-and-copyright
      4
----> 5 from .kglab import KnowledgeGraph
      6
      7 from .subg import Subgraph, SubgraphMatrix, SubgraphTensor

~/conda/envs/rapids-0.17/lib/python3.7/site-packages/kglab/kglab.py in <module>
      9 from kglab.pkg_types import PathLike, IOPathLike, GraphLike, RDF_Node
     10 from kglab.gpviz import GPViz
--> 11 from kglab.util import get_gpu_count
     12 from kglab.version import _check_version
     13 _check_version()

~/conda/envs/rapids-0.17/lib/python3.7/site-packages/kglab/util.py in <module>
     31
     32 if get_gpu_count() > 0:
--> 33     import cudf # type: ignore # pylint: disable=E0401
     34
     35
```

(continues on next page)

(continued from previous page)

```
~/.../conda/envs/rapids-0.17/lib/python3.7/site-packages/cudf/__init__.py in <module>
    9 import rmm
    10
--> 11 from cudf import core, datasets, testing
    12 from cudf._version import get_versions
    13 from cudf.api.extensions import (

~/.../conda/envs/rapids-0.17/lib/python3.7/site-packages/cudf/core/__init__.py in
-><module>
    1 # Copyright (c) 2018-2020, NVIDIA CORPORATION.
    2
----> 3 from cudf.core import buffer, column, common
    4 from cudf.core.buffer import Buffer
    5 from cudf.core.dataframe import DataFrame, from_pandas, merge

~/.../conda/envs/rapids-0.17/lib/python3.7/site-packages/cudf/core/column/__init__.py in
-><module>
    1 # Copyright (c) 2020, NVIDIA CORPORATION.
    2
----> 3 from cudf.core.column.categorical import CategoricalColumn
    4 from cudf.core.column.column import (
    5     ColumnBase,

~/.../conda/envs/rapids-0.17/lib/python3.7/site-packages/cudf/core/column/categorical.py in
->in <module>
    6
    7 import cudf
----> 8 from cudf import _lib as libcudf
    9 from cudf._lib.transform import bools_to_mask
    10 from cudf.core.buffer import Buffer

~/.../conda/envs/rapids-0.17/lib/python3.7/site-packages/cudf/_lib/__init__.py in
-><module>
    2 import numpy as np
    3
----> 4 from . import (
    5     avro,
    6     binaryop,

cudf/_lib/gpuarrow.pyx in init cudf._lib.gpuarrow()

AttributeError: module 'pyarrow.lib' has no attribute '_CRecordBatchReader'
```

```
# set up the files we will use here
dg = './datagraphs/datagraph.json'
sg = './shapes/oih_learning.ttl'
```

```
import json
from pyld import jsonld
import os, sys

currentdir = os.path.dirname(os.path.abspath('__'))
parentdir = os.path.dirname(currentdir)
sys.path.insert(0, currentdir)
from lib import jbutils
```

(continues on next page)

(continued from previous page)

```
g = Graph().parse(sg, format='ttl')

context = {"schema": "https://schema.org/",
          "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
          "shacl": "http://www.w3.org/ns/shacl#" ,
          "oihval": "https://oceans.collaborium.io/voc/validation/1.0.1/shacl#"
          }

sgjld = g.serialize(format='json-ld', context=context, indent=4)

doc = json.loads(sgjld)

frame = {
    "@context": {"schema": "https://schema.org/",
                "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
                "shacl": "http://www.w3.org/ns/shacl#" ,
                "oihval": "https://oceans.collaborium.io/voc/validation/1.0.1/shacl#"
                },
    "@explicit": "false",
    "@requireAll": "true",
    "@type": {},
    "shacl:message": "",
    "shacl:path": "",
}

compactd = jsonld.compact(doc, context)

framed = jsonld.frame(compactd, frame)
jd = json.dumps(framed, indent=4)
# print(jd)

jbutils.show_graph(framed)
```

```
-----
ModuleNotFoundError                                Traceback (most recent call last)
<ipython-input-8-88c94132ad8a> in <module>
      1 import json
----> 2 from pyld import jsonld
      3 import os, sys
      4
      5 currentdir = os.path.dirname(os.path.abspath('.'))

ModuleNotFoundError: No module named 'pyld'
```

```
# pyshack sends output to log along with the vars. This suppresses that
import logging, sys
logging.disable(sys.maxsize)
```

```
import kglab

namespaces = {
    "schema": "https://schema.org/",
    "shacl": "http://www.w3.org/ns/shacl#" ,
}
```

(continues on next page)

(continued from previous page)

```
kg = kglab.KnowledgeGraph(
    name = "Schema.org based datagraph",
    base_uri = "https://example.org/id/",
    namespaces = namespaces,
)

kg.load_jsonld(dg)
```

```
<kglab.kglab.KnowledgeGraph at 0x7fcd58ffa7f0>
```

```
conforms, report_graph, report_text = kg.validate(
    shacl_graph=sg,
    shacl_graph_format="ttl"
)
```

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

sparql = """
SELECT ?path ?value ?constraint ?message ?id ?focus
WHERE {
    ?id rdf:type shacl:ValidationResult .
    ?id shacl:focusNode ?focus .
    ?id shacl:resultMessage ?message .
    ?id shacl:sourceConstraintComponent ?constraint .
    OPTIONAL {
        ?id shacl:resultPath ?path .
    }
    OPTIONAL {
        ?id shacl:value ?value .
    }
}
"""

df = report_graph.query_as_df(sparql)
df.head(10)
```

```

      path                                constraint \
0  schema:provider  shacl:MinCountConstraintComponent
1              NaN  shacl:NodeKindConstraintComponent
2  schema:name      shacl:MinCountConstraintComponent
3  schema:url       shacl:MinCountConstraintComponent

      message \
0          A provider must be noted
1          Graph must have an ID
2          Name is required
3  URL required for the location of the resource ...

      id \
0  _:nc0c23de19cda41839d796f82d56c39b4b2
1  _:nc0c23de19cda41839d796f82d56c39b4b4
2  _:nc0c23de19cda41839d796f82d56c39b4b5
3  _:nc0c23de19cda41839d796f82d56c39b4b6
```

(continues on next page)

(continued from previous page)

	focus	\
0	_:nc0c23de19cda41839d796f82d56c39b4b3	
1	_:nc0c23de19cda41839d796f82d56c39b4b3	
2	_:nc0c23de19cda41839d796f82d56c39b4b3	
3	_:nc0c23de19cda41839d796f82d56c39b4b3	
	value	
0	NaN	
1	_:nc0c23de19cda41839d796f82d56c39b4b3	
2	NaN	
3	NaN	

```

VIS_STYLE = {
    "schema": {
        "color": "green",
        "size": 20,
    },
    "shacl": {
        "color": "red",
        "size": 20,
    },
    "_": {
        "color": "orange",
        "size": 20,
    },
}

subgraph = kglab.SubgraphTensor(report_graph)
pyvis_graph = subgraph.build_pyvis_graph(notebook=True, style=VIS_STYLE)
pyvis_graph.force_atlas_2based()
pyvis_graph.show("tmp.fig05.html")

```

```
<IPython.lib.display.IFrame at 0x7fcd5916d220>
```

27.5.12 SHACL Validation on Object Store

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 facilitates. Further this validation follows W3C recommendations as described 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'

```

(continues on next page)

(continued from previous page)

```
!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 = []
```

(continues on next page)

(continued from previous page)

```

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',
                                                       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)

```

(continues on next page)

(continued from previous page)

```

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

(continues on next page)

(continued from previous page)

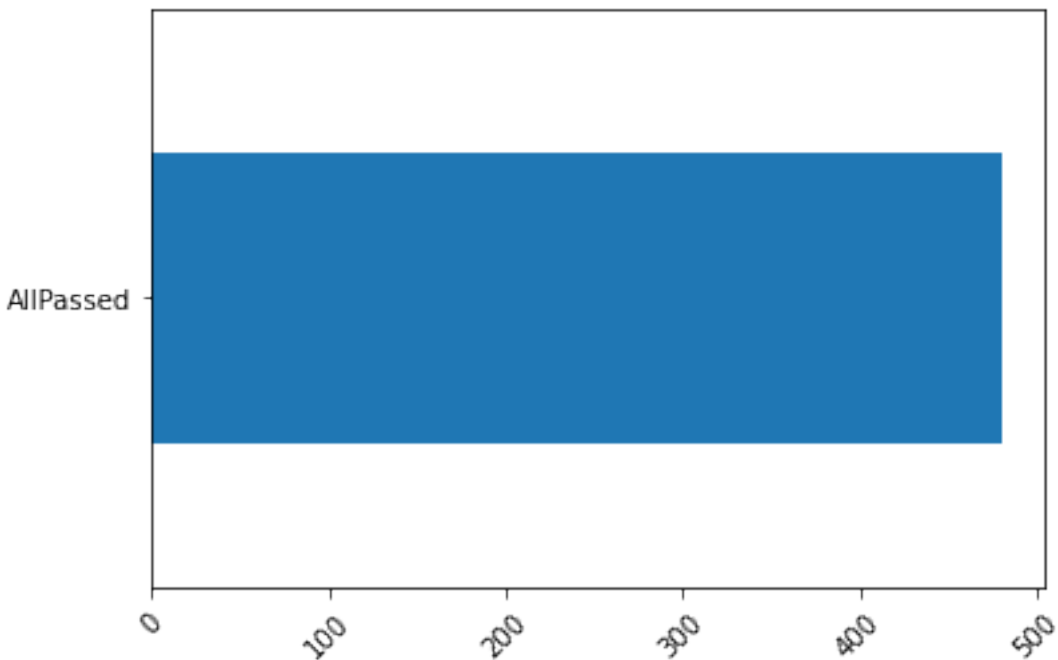
```
3 gleaner/summoned/oceanexperts/0223a997319c102b... Valid AllPassed
4 gleaner/summoned/oceanexperts/022ac35a670a36a3... Valid AllPassed
```

```
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, ''),
  Text(0, 0, 'AllPassed')])
```



27.5.13 Mapping to Schema.org

About

Notebooks related to mapping from other vocabularies to SDO

Mapping - ISO 19139

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 alternative 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
Length: 79137 (77K) [text/plain]
Saving to: 'ISO19139ToSDODatasetStandalone1.0.xslt.1'

ISO19139ToSDODataset 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:urnSDNCIDILOCALMARIS-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:urnSDNCIDILOCALMARIS-TEST.",
  "creator":
  [{
    "@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"
    }
  },
  {
    "@type": "Role",
    "roleName": "originator",
    "creator": {

      "@type": "Role",
      "roleName": "originator"
    }
  }
]
```

(continues on next page)

(continued from previous page)

```

    "@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 wherever 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 "
    },
    {
      "@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"
        }
      }
    }
  ],

```

(continues on next page)

(continued from previous page)

```

    "fileFormat": [
      "Ocean Data View ASCII input v.0.3", "MEDATLAS ASCII v.1"]
    ],
    "identifier":
    [
      {
        "@type": "PropertyValue",
        "propertyID": "dataset identifier",
        "value": "urn:urnSDNCIDILOCALMARIS-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: "
      }
    ],
    "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"
        }
      ]
    }
  ]

```

(continues on next page)

(continued from previous page)

```
}}
```

Mapping - DCAT

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 JOSH-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 otherwise 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 .
}
```

(continues on next page)

(continued from previous page)

```

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

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

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

# for row in gres:
#     print("-----")
#     print(row)

```

```

b'{"\n      "@context": {\n          "@language": "en",\n          "@vocab": "https://schema.\norg/"\n      },\n      "@id": "https://data.gov.au/dataset/67ca5de1-8774-4678-9d1b-\n8b1cb70ab33c",\n      "identifier": {\n          "@value": "67ca5de1-8774-4678-9d1b-\n8b1cb70ab33c"\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(
    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 .
}

```

(continues on next page)

(continued from previous page)

```
"""

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

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

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

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)
```

(continues on next page)

(continued from previous page)

```
# 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 source IRI that 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
```

(continues on next page)

(continued from previous page)

```
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 ;
    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" .
```

27.5.14 Exploration

About

The `exploration` directory contains notebooks that demonstrate how to map existing approaches to generate the patterns and data assets that align to OIH guidance.

Geo KG Completion

About

This is a simple test notebook to explore approaches 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 shape file that is converted to WKT and loaded into a geopandas dataframe.

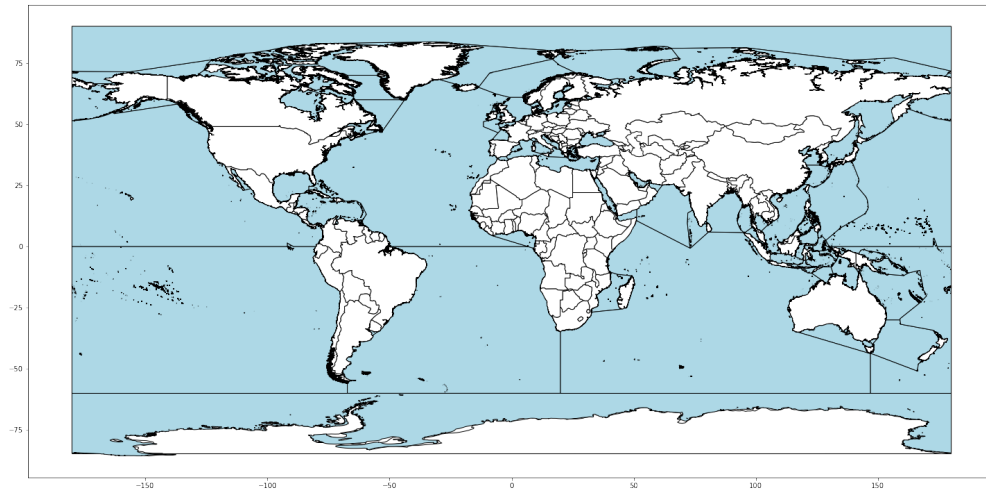
We then create a few points to compare against this. In this test we are only comparing points to the polygons. So this is what is called a “point in polygon” test. Later revisions to this could do polygon intersection or other tests. This is only a proof of concept notebook.

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 could be fed back into the graph as keywords or items from a known list of terms for more explicate relation mapping.

Specifically, something 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” comes from the official Marine Regions file.

```
##@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:
```

(continues on next page)

(continued from previous page)

```
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)
```

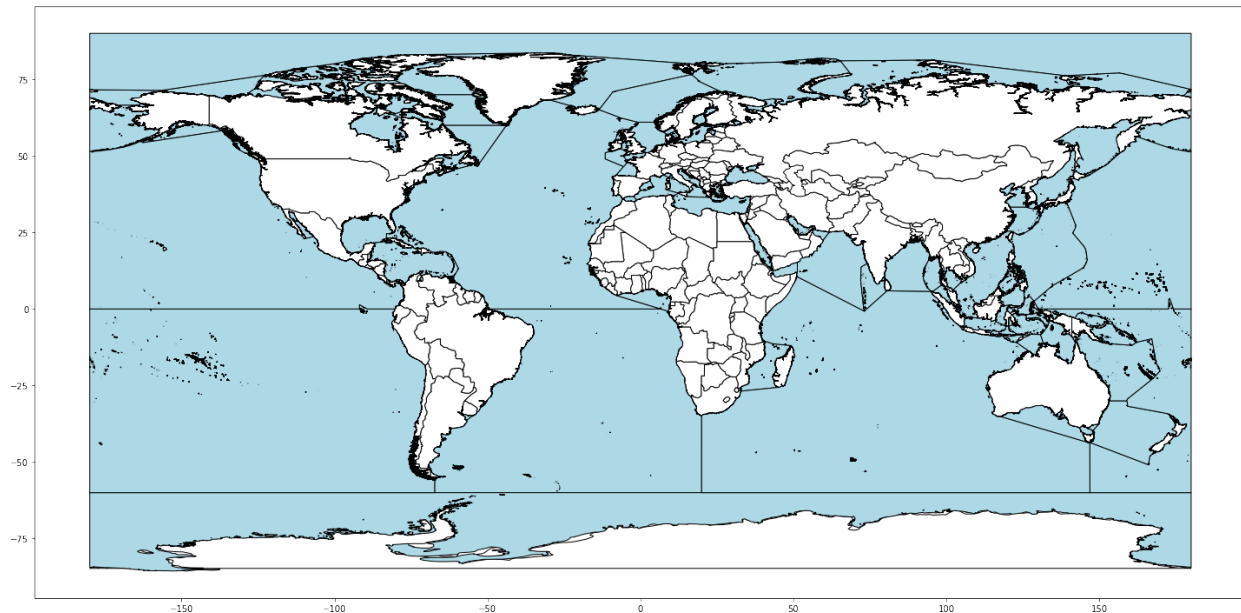
```
# Load and plot the ocean regions
import contextily as ctx

# 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
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 could be fed back into the graph as keywords or items from a known list of terms for more explicate relation mapping.

Specifically, something 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” comes from the official 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

California DWR sprint

- <https://earthref.org/GERM/metadata/papers/hellyetal2002.htm>
- https://ezid.cdlib.org/id/doi:10.4246/CA-DWR-GROUNDWATER-LEVELS_20171109_LEVEL0-MD5B8CA0D403FCE165F9E180CC73C648800.TAR
- <https://californiacoastalatlantlas.net/california-department-of-water-resources-water-balance-library>
- ftp://owia.sdsc.edu/Project-OWIA-Library-WaterBalance/CA-DWR-WaterBalance_20190505_CA-DWR-WaterBalance-2011-md5a3f47721f42c4f06075856218dd043ef.zip
- https://ezid.cdlib.org/manage/display_xml/doi:10.4246/CA-DWR-GROUNDWATER-LEVELS_20171109_LEVEL0-MD5B8CA0D403FCE165F9E180CC73C648800.TAR
- https://ezid.cdlib.org/manage/display_xml/doi:10.4246/CA-DWR-GROUNDWATER-LEVELS_20171109_LEVEL0-MD5B8CA0D403FCE165F9E180CC73C648800.TAR

Next steps

DF to look at MIF files; Will contact John as needed. JH and PLB / Doug to follow up on publishing tools

Marine Protected Areas API exploration

References:

- <https://www.protectedplanet.net/en>

```
from urllib.request import urlopen
import ssl, json, os, sys
from pyld import jsonld
```

```
apicall = "https://mpa.protectedseas.net/php/mpa_search.php?lat=33.90689555128866&
lon=-119.46258544921874&high=true&mid=true&low=true&min=true&v=1&lang=en_EN"
detailscall = "https://mpa.protectedseas.net/php/mpaDetail.php?gid=156&complexity=118&
rid=0&v=1&lang=en_EN"
```

```
ssl._create_default_https_context = ssl._create_unverified_context

f = urlopen(detailscall)
jr = f.read()
json_object = json.loads(jr)
json_formatted_str = json.dumps(json_object, indent=2)
```


Notes

There are two ways we can look at this. We need to convert this likely to something like a Creative Works / Map. Something like we have done at: <https://book.oceaninfohub.org/thematics/docs/maps.html>. We could also do this at a Dataset, but that would take a bit of a broad interpretation.

To convert from JSON to JSON-LD we can either parse and build or look at some sort of generic conversion or workflow like OpenRefine.

```
data = {}

data['https://schema.org/name'] = json_object["site_name"]
data['@id'] = "http://example.org/1" # json_object["url"]
data['https://schema.org/description'] = json_object["purpose"]
```

```
currentdir = os.path.dirname(os.path.abspath('__'))
parentdir = os.path.dirname(currentdir)
sys.path.insert(0, currentdir)
from lib import jbutils

context = {"@vocab": "https://schema.org/"}
compacted = jsonld.compact(data, context)

jbutils.show_graph(compacted)
```

```
<graphviz.dot.Digraph at 0x7eff8c7803d0>
```

```
print(json_object)
```

```
{'site_id': 'NMF960', 'site_name': 'Footprint (Anacapa Channel) Federal Marine Reserve',
  'url': 'MPA Website|https://www.wildlife.ca.gov/Conservation/Marine/MPAs/Network/Southern-California',
  'country': 'USA', 'state': 'CA ', 'managing_authority': 'NOAA National Marine Fisheries Service',
  'designation': 'Federal Marine Reserve',
  'purpose': 'To protect the rich kelp forest ecosystem that is home to abalone, crabs, eel, Garibaldi, rock fish, sea bass, sea lions, eel, lobster, sea urchins and many other marine residents in the kelp forest ecosystem.',
  'restrictions': '1. Take of all living marine resources is prohibited. <br>2. Possessing fishing gear onboard a vessel unless such gear is stowed and not available for immediate use. <br>3. Possessing any Sanctuary resource, except legally-taken fish onboard a vessel at anchor or in transit.',
  'allowed': 'None.', 'regulation_name': '15 C.F.R. § 922.73',
  'regulation_url': 'Regulations Text|https://www.ecfr.gov/cgi-bin/text-id?SID=d2d103de414f2056167c2fa1b50a3fa5&mc=true&node=pt15.3.922&rgn=div5#se15.3.922_173',
  'season': 'Year-round', 'effective_from': '', 'effective_to': '',
  'report_violations': 'CalTIP 1 (888) 334-2258', 'latest_updates': 'For the latest fishing regulation information:<br>Call or drop by your local Marine Region CDFW office.<br>In-season changes to commercial and recreational fishing regulations: <a href="https://www.wildlife.ca.gov/Fishing/Ocean/Regulations/Inseason" target="_blank">https://www.wildlife.ca.gov/Fishing/Ocean/Regulations/Inseason</a><br>Sport Fishing: <a href="https://www.dfg.ca.gov/marine/fishing_map.asp" target="_blank">DFG Sportfishing Regulations</a><br>Groundfish fishing regulations: (831) 649-2801<br>Ocean salmon fishing regulations: (707) 576-3429<br>CDFW News Room at <a href="https://www.wildlife.ca.gov/news" target="_blank">CDFW News</a><br>Marine Region News Page at <a href="https://www.dfg.ca.gov/marine/news.asp" target="_blank">Marine News</a><br>Marine Region News Service (email notification of ocean-related news) <a href="https://www.dfg.ca.gov/marine/subscribe.asp" target="_blank">Marine Email Subscription</a><br>For all questions, contact the CDFW Marine Region Headquarters: (831) 649-2870, <a href="mailto:AskMarine@wildlife.ca.gov">AskMarine@wildlife.ca.gov</a>',
  'clarification_needed': 'null', 'bounds': {'type': 'MultiPolygon', 'coordinates': [[[[[-119.433254, 33.954493, 0], [-119.43311, 33.901981, 0], [-119.51609, 33.901981, 0], [-119.51649, 33.958432, 0], [-119.516489, 33.958432, 0], [-119.515691, 33.959, 0], [-119.514905, 33.959579, 0], [-119.514131, 33.960169, 0], [-119.51337, 33.960771, 0], [-119.512621, 33.961383, 0], [-119.511886, 33.962006, 0], [-119.511163, 33.96264, 0], [-119.510454, 33.963285, 0], [-119.509759, 33.963939, 0], [-119.509077, 33.964604, 0], [-119.508409, 33.965278, 0],
```

(continues on next page)

27.5. OIH Notebooks

(continued from previous page)

```
print(json_formatted_str)
```

```
{
  "site_id": "NMF960",
  "site_name": "Footprint (Anacapa Channel) Federal Marine Reserve",
  "url": "MPA Website|https://www.wildlife.ca.gov/Conservation/Marine/MPAs/Network/
↳Southern-California",
  "country": "USA",
  "state": "CA ",
  "managing_authority": "NOAA National Marine Fisheries Service",
  "designation": "Federal Marine Reserve",
  "purpose": "To protect the rich kelp forest ecosystem that is home to abalone,
↳crabs, eel, Garibaldi, rock fish, sea bass, sea lions, lobster, sea urchins.
↳and many other marine residents in the kelp forest ecosystem.",
  "restrictions": "1. Take of all living marine resources is prohibited. <br>2.
↳Possessing fishing gear onboard a vessel unless such gear is stowed and not
↳available for immediate use. <br>3. Possessing any Sanctuary resource, except
↳legally-taken fish onboard a vessel at anchor or in transit.",
  "allowed": "None.",
  "regulation_name": "15 C.F.R. \u00a7 922.73",
  "regulation_url": "Regulations Text|https://www.ecfr.gov/cgi-bin/text-idx?
↳SID=d2d103de414f2056167c2fa1b50a3fa5&mc=true&node=pt15.3.922&rgn=div5#se15.3.922_173
↳",
  "season": "Year-round",
  "effective_from": "",
  "effective_to": "",
  "report_violations": "CalTIP 1 (888) 334-2258",
  "latest_updates": "For the latest fishing regulation information:<br>Call or drop
↳by your local Marine Region CDFW office.<br>In-season changes to commercial and
↳recreational fishing regulations: <a href=\"https://www.wildlife.ca.gov/Fishing/
↳Ocean/Regulations/Inseason\" target=\"_blank\">https://www.wildlife.ca.gov/Fishing/
↳Ocean/Regulations/Inseason</a><br>Sport Fishing: <a href=\"https://www.dfg.ca.gov/
↳marine/fishing_map.asp\" target=\"_blank\">DFG Sportfishing Regulations</a><br>
↳Groundfish fishing regulations: (831) 649-2801<br>Ocean salmon fishing regulations:
↳(707) 576-3429<br>CDFW News Room at <a href=\"https://www.wildlife.ca.gov/news\"
↳target=\"_blank\">CDFW News</a><br>Marine Region News Page at <a href=\"https://www.
↳dfg.ca.gov/marine/news.asp\" target=\"_blank\">Marine News</a><br>Marine Region
↳News Service (email notification of ocean-related news) <a href=\"https://www.dfg.
↳ca.gov/marine/subscribe.asp\" target=\"_blank\">Marine Email Subscription</a><br>
↳For all questions, contact the CDFW Marine Region Headquarters: (831) 649-2870, <a
↳href=\"mailto:AskMarine@wildlife.ca.gov\">AskMarine@wildlife.ca.gov</a>",
  "clarification_needed": "null",
  "bounds": {
    "type": "MultiPolygon",
    "coordinates": [
      [
        [
          [
            -119.433254,
            33.954493,
            0
          ],
          [
            -119.43311,
```

(continues on next page)

(continued from previous page)

```

33.901981,
0
],
[
-119.51609,
33.901981,
0
],
[
-119.51649,
33.958432,
0
],
[
-119.516489,
33.958432,
0
],
[
-119.515691,
33.959,
0
],
[
-119.514905,
33.959579,
0
],
[
-119.514131,
33.960169,
0
],
[
-119.51337,
33.960771,
0
],
[
-119.512621,
33.961383,
0
],
[
-119.511886,
33.962006,
0
],
[
-119.511163,
33.96264,
0
],
[
-119.510454,
33.963285,
0

```

(continues on next page)

(continued from previous page)

```
],  
[  
  -119.509759,  
  33.963939,  
  0  
],  
[  
  -119.509077,  
  33.964604,  
  0  
],  
[  
  -119.508409,  
  33.965278,  
  0  
],  
[  
  -119.507756,  
  33.965962,  
  0  
],  
[  
  -119.507116,  
  33.966655,  
  0  
],  
[  
  -119.506492,  
  33.967358,  
  0  
],  
[  
  -119.505882,  
  33.968069,  
  0  
],  
[  
  -119.505288,  
  33.968789,  
  0  
],  
[  
  -119.505105,  
  33.969016,  
  0  
],  
[  
  -119.504908,  
  33.969173,  
  0  
],  
[  
  -119.504159,  
  33.969786,  
  0  
],  
[
```

(continues on next page)

(continued from previous page)

```
-119.503424,  
33.970409,  
0  
],  
[  
-119.502701,  
33.971043,  
0  
],  
[  
-119.501992,  
33.971687,  
0  
],  
[  
-119.501296,  
33.972342,  
0  
],  
[  
-119.500614,  
33.973006,  
0  
],  
[  
-119.499947,  
33.973681,  
0  
],  
[  
-119.499293,  
33.974365,  
0  
],  
[  
-119.498654,  
33.975058,  
0  
],  
[  
-119.498029,  
33.97576,  
0  
],  
[  
-119.49742,  
33.976472,  
0  
],  
[  
-119.496825,  
33.977192,  
0  
],  
[  
-119.496245,  
33.977921,
```

(continues on next page)

(continued from previous page)

```
0
],
[
  -119.495681,
  33.978658,
  0
],
[
  -119.495132,
  33.979403,
  0
],
[
  -119.494599,
  33.980156,
  0
],
[
  -119.494082,
  33.980917,
  0
],
[
  -119.493581,
  33.981685,
  0
],
[
  -119.493096,
  33.98246,
  0
],
[
  -119.492627,
  33.983242,
  0
],
[
  -119.492588,
  33.98331,
  0
],
[
  -119.492372,
  33.983077,
  0
],
[
  -119.491719,
  33.982393,
  0
],
[
  -119.491051,
  33.981719,
  0
],
],
```

(continues on next page)

(continued from previous page)

```
[
  -119.490369,
  33.981054,
  0
],
[
  -119.489673,
  33.9804,
  0
],
[
  -119.488964,
  33.979755,
  0
],
[
  -119.488241,
  33.979122,
  0
],
[
  -119.487506,
  33.978498,
  0
],
[
  -119.487104,
  33.978167,
  0
],
[
  -119.487087,
  33.978153,
  0
],
[
  -119.486338,
  33.977541,
  0
],
[
  -119.485577,
  33.976939,
  0
],
[
  -119.485068,
  33.976549,
  0
],
[
  -119.48471,
  33.976278,
  0
],
[
  -119.483924,
```

(continues on next page)

(continued from previous page)

```
33.975699,  
0  
],  
[  
-119.483126,  
33.975132,  
0  
],  
[  
-119.482316,  
33.974576,  
0  
],  
[  
-119.481494,  
33.974032,  
0  
],  
[  
-119.480662,  
33.9735,  
0  
],  
[  
-119.479818,  
33.97298,  
0  
],  
[  
-119.478964,  
33.972473,  
0  
],  
[  
-119.478099,  
33.971978,  
0  
],  
[  
-119.477223,  
33.971495,  
0  
],  
[  
-119.476338,  
33.971026,  
0  
],  
[  
-119.475444,  
33.970569,  
0  
],  
[  
-119.474539,  
33.970126,  
0
```

(continues on next page)

(continued from previous page)

```
],  
[  
  -119.473626,  
  33.969696,  
  0  
],  
[  
  -119.472704,  
  33.969279,  
  0  
],  
[  
  -119.471773,  
  33.968875,  
  0  
],  
[  
  -119.471299,  
  33.968676,  
  0  
],  
[  
  -119.470642,  
  33.968316,  
  0  
],  
[  
  -119.469757,  
  33.967847,  
  0  
],  
[  
  -119.468862,  
  33.96739,  
  0  
],  
[  
  -119.467958,  
  33.966947,  
  0  
],  
[  
  -119.467044,  
  33.966516,  
  0  
],  
[  
  -119.466122,  
  33.966099,  
  0  
],  
[  
  -119.465191,  
  33.965696,  
  0  
],  
[
```

(continues on next page)

(continued from previous page)

```
-119.464252,  
33.965306,  
0  
],  
[  
-119.464079,  
33.965236,  
0  
],  
[  
-119.463988,  
33.965199,  
0  
],  
[  
-119.463041,  
33.964823,  
0  
],  
[  
-119.462086,  
33.96446,  
0  
],  
[  
-119.461124,  
33.964112,  
0  
],  
[  
-119.460154,  
33.963778,  
0  
],  
[  
-119.459178,  
33.963457,  
0  
],  
[  
-119.458195,  
33.963151,  
0  
],  
[  
-119.457373,  
33.962908,  
0  
],  
[  
-119.457138,  
33.96277,  
0  
],  
[  
-119.456273,  
33.962275,
```

(continues on next page)

(continued from previous page)

```

0
],
[
  -119.455398,
  33.961793,
  0
],
[
  -119.454513,
  33.961323,
  0
],
[
  -119.453618,
  33.960867,
  0
],
[
  -119.452714,
  33.960423,
  0
],
[
  -119.451801,
  33.959993,
  0
],
[
  -119.450879,
  33.959576,
  0
],
[
  -119.449948,
  33.959173,
  0
],
[
  -119.449009,
  33.958783,
  0
],
[
  -119.448062,
  33.958407,
  0
],
[
  -119.447107,
  33.958044,
  0
],
[
  -119.446145,
  33.957696,
  0
],
],

```

(continues on next page)

(continued from previous page)

```
[
  -119.445176,
  33.957362,
  0
],
[
  -119.4442,
  33.957041,
  0
],
[
  -119.443217,
  33.956735,
  0
],
[
  -119.442228,
  33.956444,
  0
],
[
  -119.441233,
  33.956166,
  0
],
[
  -119.440232,
  33.955904,
  0
],
[
  -119.439226,
  33.955656,
  0
],
[
  -119.438215,
  33.955422,
  0
],
[
  -119.4372,
  33.955203,
  0
],
[
  -119.436313,
  33.955025,
  0
],
[
  -119.435981,
  33.95496,
  0
],
[
  -119.434957,
```

(continues on next page)

(continued from previous page)

```

        33.954771,
        0
    ],
    [
        -119.433929,
        33.954597,
        0
    ],
    [
        -119.433254,
        33.954493,
        0
    ]
    ]
    ]
    ],
    "boundary_source": "California Department of Fish and Wildlife",
    "modification_level": "0",
    "references": []
}

```

SPREP Data Testing

Notes

We are close on this one. Really just a few mods to the context and the sitemap and we are ready to index.

- https://pacificdata.org/data/api/3/action/package_list
- https://pacificdata.org/data/api/3/action/package_show?id=SPC-DF_POP_COAST
- <https://github.com/PacificCommunity/ckanext-spc-pdh/blob/master/ckanext/spc/schemas/dataset.json>
- <http://dev.d9fsm.sprep.wod.by/>
- <http://dev.d9fsm.sprep.wod.by/dataset/151/jsonld>

```

import kglab as kg
from rdflib import Graph, plugin
from rdflib.serializer import Serializer
import pyshacl
import logging, sys

```

```

import json
from pyld import jsonld
import os, sys

currentdir = os.path.dirname(os.path.abspath('__'))
parentdir = os.path.dirname(currentdir)
sys.path.insert(0, currentdir)
from lib import jbutils

context = {"schema": "https://schema.org/",
           "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
           "shacl": "http://www.w3.org/ns/shacl#" ,

```

(continues on next page)

(continued from previous page)

```

        "oihval": "https://oceans.collaborium.io/voc/validation/1.0.1/shacl#"
    }

# docurl = "http://dev.d9fsm.sprep.wod.by/dataset/151/jsonld"
# g = Graph().parse("http://dev.d9fsm.sprep.wod.by/dataset/151/jsonld", format='json-ld')
# sgjld = g.serialize(format='json-ld', context=context, indent=4)
# doc = json.loads("http://dev.d9fsm.sprep.wod.by/dataset/151/jsonld")

# load file
g = Graph().parse("./sprep1.json", format='json-ld')
sgjld = g.serialize(format='json-ld', context=context, indent=4)
doc = json.loads(sgjld)

# print(doc)

# compacted = jsonld.compact(docurl, context)
compacted = jsonld.compact(doc, context)

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

jbutils.show_graph(compacted)

```

```
<graphviz.dot.Digraph at 0x7f1f31de9130>
```

Validation call

```

# set up the files we will use here
dg = './sprep1.json'
sg = '../validation/shapes/oih_dataset.ttl'

```

```

# pyshack sends output to log along with the vars. This suppresses that
logging.disable(sys.maxsize)

```

```

import kglab

namespaces = {
    "schema": "https://schema.org/",
    "shacl": "http://www.w3.org/ns/shacl#" ,
}

kg = kglab.KnowledgeGraph(
    name = "Schema.org based datagraph",
    base_uri = "https://example.org/id/",
    namespaces = namespaces,
)

kg.load_jsonld(dg)

```

```
<kglab.kglab.KnowledgeGraph at 0x7f1e24730a90>
```

```
conforms, report_graph, report_text = kg.validate(
    shacl_graph=sg,
    shacl_graph_format="ttl"
)
```

```
print(report_text)
```

```
Validation Report
Conforms: False
Results (2):
Constraint Violation in MinCountConstraintComponent (http://www.w3.org/ns/shacl
↳#MinCountConstraintComponent):
    Severity: shacl:Violation
    Source Shape: oihval:urlResourceProperty
    Focus Node: <http://dev.d9fsm.sprep.wod.by/dataset/151>
    Result Path: schema:url
    Message: URL required for the location of the resource described by this.
↳metadata
Constraint Violation in MinCountConstraintComponent (http://www.w3.org/ns/shacl
↳#MinCountConstraintComponent):
    Severity: shacl:Violation
    Source Shape: oihval:identifierResourceProperty
    Focus Node: <http://dev.d9fsm.sprep.wod.by/dataset/151>
    Result Path: schema:identifier
    Message: Resource must have an identifier node
```

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

sparql = """
SELECT ?path ?value ?constraint ?message ?id ?focus ?desc
WHERE {
    ?id rdf:type shacl:ValidationResult .
    ?id shacl:focusNode ?focus .
    ?id shacl:resultMessage ?message .
    ?id shacl:sourceConstraintComponent ?constraint .
    OPTIONAL {
        ?id shacl:resultPath ?path .
    }
    OPTIONAL {
        ?id shacl:value ?value .
    }
    OPTIONAL {
        ?id shacl:description ?desc .
    }
}
"""

df = report_graph.query_as_df(sparql)
df.head(10)
```

```

           path                                     constraint \
0      schema:url  shacl:MinCountConstraintComponent
1  schema:identifier  shacl:MinCountConstraintComponent
                                     message \
```

(continues on next page)

(continued from previous page)

```
0 URL required for the location of the resource ...
1 Resource must have an identifier node

                                id \
0 _:n03a8abebbbe694760844b154392a85363b2
1 _:n03a8abebbbe694760844b154392a85363b3

                                focus
0 <http://dev.d9fsm.sprep.wod.by/dataset/151>
1 <http://dev.d9fsm.sprep.wod.by/dataset/151>
```

Maspawio

Dublin Core to Schema.org workflow testing

An experiment to see if we can convert a Dublin Core record to Schema.org. The focus will be on OIH type <https://book.oceaninfohub.org/thematics/docs/README.html>

The source location is: https://maspawio.net/layers/geonode:locally_managed_marine_areas_kenya

Need somethnig like documented at: <https://book.oceaninfohub.org/thematics/docs/maps.html> We will need to add into this a spital geometry to scope the bounding box.

https://docs.google.com/spreadsheets/d/1OS_DPrPppxkWR7kU3vjRJJaDSCq3XF74/edit#gid=1139001206

Note OIH Themtic mapping

What existing (or not) patterns do we align to

Document Risk Pattens

- URI pattern (URI governance)
- Sitemap update patterns

Steps to alignment with Structured Data on the Web

- ...

```
# !pip install -q kglab
!pip install -q graphviz
```

```
import json
import rdflib
from pyld import jsonld
import os, sys
# import kglab as kg
from rdflib import Graph, plugin
from rdflib.serializer import Serializer
from icecream import ic
```

(continues on next page)

(continued from previous page)

```
from pathlib import Path
import xml.etree.ElementTree as ET
```

```
# Source Dublin core in XML
dcxml = "https://maspawio.net/catalogue/csw?outputschema=http%3A%2F%2Fwww.opengis.net
↪%2Fcat%2Fcsrw%2F2.0.2&service=CSW&request=GetRecordById&version=2.0.2&
↪elementsetname=full&id=3046b5fc-18e0-11eb-894f-0a735f7a740c"
```

```
tree = ET.parse('dublin.xml')
root = tree.getroot()
# print(root.tag)
```

```
r = root.find('{http://www.opengis.net/cat/csw/2.0.2}Record')
```

```
# id
id = r.find('{http://purl.org/dc/elements/1.1/}identifier')
print(id.text)

# name
# This can be used to form the URL: https://maspawio.net/layers/geonode%3Alocally_
↪managed_marine_areas_kenya
name = r.find('{http://purl.org/dc/elements/1.1/}title')
print(name.text)

# description
description = r.find('{http://purl.org/dc/terms/}abstract')
print(description.text)

# keywords
subjects = r.findall('{http://purl.org/dc/elements/1.1/}subject')
for s in subjects:
    print(s.text)

# url https://maspawio.net/layers/geonode:locally_managed_marine_areas_kenya
```

```
3046b5fc-18e0-11eb-894f-0a735f7a740c
locally managed marine areas Kenya
This resource/layer shows the spatial location of locally managed marine areas (LMMA) ↪
↪along the Kenyan coast. The information comes from a report done by CORDIO in 2015 ↪
↪that sought to review a detailed history of LMMA development in Kenya. For more ↪
↪information, please contact jkawaka@cordioea.net

kenya
LMMA
```

```
data = {}

data['https://schema.org/name'] = name.text
data['@id'] = "http://example.org/1" #id.text
data['https://schema.org/description'] = description.text
```

```
currentdir = os.path.dirname(os.path.abspath(''))
parentdir = os.path.dirname(currentdir)
```

(continues on next page)

(continued from previous page)

```
sys.path.insert(0, currentdir)
from lib import jbutils

context = {"@vocab": "https://schema.org/"}
compacted = jsonld.compact(data, context)

jbutils.show_graph(compacted)
```

```
<graphviz.dot.Digraph at 0x7efbdde35430>
```

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

```
{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@id": "http://example.org/1",
  "description": "This resource/layer shows the spatial location of locally managed_
↳marine areas (LMMA) along the Kenyan coast. The information comes from a report_
↳done by CORDIO in 2015 that sought to review a detailed history of LMMA development_
↳in Kenya. For more information, please contact jkawaka@cordioea.net\r\n\r\n",
  "name": "locally managed marine areas Kenya"
}
```

```
for actor in root.findall('{http://www.opengis.net/cat/csw/2.0.2}Record'):
#     print(actor)

    for child in actor:
        print(child.tag, child.text)
```

```
{http://purl.org/dc/elements/1.1/}identifier 3046b5fc-18e0-11eb-894f-0a735f7a740c
{http://purl.org/dc/elements/1.1/}title locally managed marine areas Kenya
{http://purl.org/dc/elements/1.1/}type dataset
{http://purl.org/dc/elements/1.1/}subject kenya
{http://purl.org/dc/elements/1.1/}subject LMMA
{http://purl.org/dc/elements/1.1/}format vector
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/geonode/wfs
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/geonode/wms
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/wms?
↳request=GetLegendGraphic&format=image/png&WIDTH=20&HEIGHT=20&LAYER=geonode:locally_
↳managed_marine_areas_kenya&legend_options=fontAntiAliasing:true;fontSize:12;
↳forceLabels:on
{http://purl.org/dc/terms/}references http://maspawio.net/uploaded/thumbs/layer-
↳3046b5fc-18e0-11eb-894f-0a735f7a740c-thumb.png
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/wms/reflect?
↳layers=geonode:locally_managed_marine_areas_kenya&width=200&height=150&format=image/
↳png8
{http://purl.org/dc/terms/}references http://maspawio.net/layers/geonode%3Alocally_
↳managed_marine_areas_kenya
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/gwc/service/gmaps?
↳layers=geonode:locally_managed_marine_areas_kenya&zoom={z}&x={x}&y={y}&format=image/
↳png8
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/wms/kml?
↳layers=geonode%3Alocally_managed_marine_areas_kenya&mode=refresh
```

(continues on next page)

(continued from previous page)

```
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/wms/kml?
↳layers=geonode%3Alocally_managed_marine_areas_kenya&mode=download
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/wfs?srsName=EPSG
↳%3A4326&typename=geonode%3Alocally_managed_marine_areas_kenya&outputFormat=json&
↳version=1.0.0&service=WFS&request=GetFeature
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/wfs?
↳typename=geonode%3Alocally_managed_marine_areas_kenya&outputFormat=excel&version=1.
↳0.0&request=GetFeature&service=WFS
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/wfs?
↳typename=geonode%3Alocally_managed_marine_areas_kenya&outputFormat=csv&version=1.0.
↳0&request=GetFeature&service=WFS
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/wfs?
↳typename=geonode%3Alocally_managed_marine_areas_kenya&outputFormat=text%2Fxml
↳%3B&subtype%3Dgml%2F3.1.1&version=1.0.0&request=GetFeature&service=WFS
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/wfs?
↳typename=geonode%3Alocally_managed_marine_areas_kenya&outputFormat=gml2&version=1.0.
↳0&request=GetFeature&service=WFS
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/wfs?format_
↳options=charset%3AUTF-8&typename=geonode%3Alocally_managed_marine_areas_kenya&
↳outputFormat=SHAPE-ZIP&version=1.0.0&service=WFS&request=GetFeature
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/wms?layers=geonode
↳%3Alocally_managed_marine_areas_kenya&width=400&bbox=39.235491%2C-4.674234%2C41.
↳1162229999999%2C-2.094547&service=WMS&format=image%2Fpng&srs=EPSG%3A4326&
↳request=GetMap&height=550
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/wms?layers=geonode
↳%3Alocally_managed_marine_areas_kenya&width=400&bbox=39.235491%2C-4.674234%2C41.
↳1162229999999%2C-2.094547&service=WMS&format=application%2Fpdf&srs=EPSG%3A4326&
↳request=GetMap&height=550
{http://purl.org/dc/terms/}references http://maspawio.net/geoserver/wms?layers=geonode
↳%3Alocally_managed_marine_areas_kenya&width=400&bbox=39.235491%2C-4.674234%2C41.
↳1162229999999%2C-2.094547&service=WMS&format=image%2Fjpeg&srs=EPSG%3A4326&
↳request=GetMap&height=550
{http://purl.org/dc/terms/}modified 2020-10-28
{http://purl.org/dc/terms/}abstract This resource/layer shows the spatial location of
↳locally managed marine areas (LMMMA) along the Kenyan coast. The information comes
↳from a report done by CORDIO in 2015 that sought to review a detailed history of
↳LMMMA development in Kenya. For more information, please contact jkawaka@cordioea.net

{http://purl.org/dc/elements/1.1/}date 2020-10-28
{http://purl.org/dc/elements/1.1/}language eng
{http://www.opengis.net/ows}BoundingBox
```

```
# for actor in root.findall('{http://www.opengis.net/ows}BoundingBox'):
#     for child in actor:
#         print(child.tag, child.text)

# bb = root.find('{http://www.opengis.net/ows}BoundingBox')
# print(bb)
```

Marine Regions LD Feed

Exploring working with the data from these commands

Current thoughts

- Getting the data graph is a simple call and then we can frame it and use similar methods to get our URL set.
- At that point it's the classic Gleaner workflow, no changes seen.
- As we get new feed updates we can leverage the prov graph generated by Gleaner combined with the feed time stamps (nothing hard there)
- This is a nice way to maintain updates in the OIH graph vs the prov pattern (complements likely, not replaces)
- put the feed URL in the robots.txt?

Data graphs via curl

```
curl -L --url http://marineregions.org/mrgid/2795.jsonld
curl -L --url http://marineregions.org/mrgid/2795 -H Accept:application/ld+json
curl -L --url http://marineregions.org/feed -HAccept:application/ld+json
```

refs:

- LDF
- LDES specification
- TREE and the TREEcg repo
- <https://www.marineregions.org/gazetteer.php?p=webservices>

my installs and imports..

```
!pip install -q 'PyLD>=2.0.3'
```

```
from pyld import jsonld
import json
```

Problems

In the following, if I frame with “framenc”, I get results like:

```
{
  "@id": "mrgid/103?t=1626519699",
  "dcterms:isVersionOf": {
    "@id": "mrgid/103"
  },
  "dcterms:modified": {
    "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
```

(continues on next page)

(continued from previous page)

```

    "@value": "2021-07-17T11:01:39Z"
  },
},

```

which lack the full domain name based @id values. I was hoping to get results with a full qualified domain name from the frame and then I would simply place that into a native Go struct that I could pass along to my regular indexing code without changes.

If I use “frame” which should be the full context. I get an error “list index out of range”

Also, why does the context contain both

- "dcterms": "http://purl.org/dc/terms/",
- "dct": "http://purl.org/dc/terms//dup/",

Both are used in the data graph.. when I tried to align them though, I got some other errors.. likely I had an error in the context.

```

docurl = "http://marineregions.org/feed.jsonld"

# with open('./data/mregFeedv2.json', 'rb') as f:
#     doc = f.read()

# this is not used.. had it for when playing with compact and expand
context = {
    "tree": "https://w3id.org/tree#",
    "ldes": "https://w3id.org/ldes#",
    "ldes:versionKey": {
        "@type": "@id",
        "@container": "@list"
    },
    "dcterms": "http://purl.org/dc/terms/",
    "dcterms:isVersionOf": {
        "@type": "@id"
    },
    "dcterms:modified": {
        "@type": "xsd:dateTime"
    },
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "sh": "http://www.w3.org/ns/shacl#",
    "schema": "https://schema.org/",
    "skos": "http://www.w3.org/2004/02/skos/core#",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "mr": "http://marineregions.org/ns/ontology#",
    "dct": "http://purl.org/dc/terms//dup/",
    "gsp": "http://www.opengis.net/ont/geosparql#",
    "dcat": "http://www.w3.org/ns/dcat#"
}

frame = {
    "@context": {
        "tree": "https://w3id.org/tree#",
        "ldes": "https://w3id.org/ldes#",
        "ldes:versionKey": {
            "@type": "@id",
            "@container": "@list"
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    },
    "dcterms": "http://purl.org/dc/terms/",
    "dcterms:isVersionOf": {
        "@type": "@id"
    },
    "dcterms:modified": {
        "@type": "xsd:dateTime"
    },
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "sh": "http://www.w3.org/ns/shacl#",
    "schema": "https://schema.org/",
    "skos": "http://www.w3.org/2004/02/skos/core#",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "mr": "http://marineregions.org/ns/ontology#",
    "dct": "http://purl.org/dc/terms/dup/",
    "gsp": "http://www.opengis.net/ont/geosparql#",
    "dcat": "http://www.w3.org/ns/dcat#"
  },
  "@explicit": "true",
  "dcterms:isVersionOf": "",
  "dcterms:modified": ""
}

framenc = {
  "@context": {"schema": "https://schema.org/",
               "dcterms": "http://purl.org/dc/terms/"},
  "@explicit": "true",
  "dcterms:isVersionOf": "",
  "dcterms:modified": ""
}

# rg = jsonld.expand(doc, context) # would need to define context for this code
rg = jsonld.frame(docurl, framenc) ## framenc: works.. but no FQDN. frame: ⚠
    ↪ gives errors

jd = json.dumps(rg, indent=4)
print(jd)

```

```

{
  "@context": {
    "schema": "https://schema.org/",
    "dcterms": "http://purl.org/dc/terms/"
  },
  "@graph": [
    {
      "@id": "mrgid/1053?t=1629807612",
      "dcterms:isVersionOf": {
        "@id": "mrgid/1053"
      },
      "dcterms:modified": {
        "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
        "@value": "2021-08-24T12:20:12Z"
      }
    },
    {
      "@id": "mrgid/63408?t=1629807612",

```

(continues on next page)

(continued from previous page)

```
    "dcterms:isVersionOf": {
      "@id": "mrgid/63408"
    },
    "dcterms:modified": {
      "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
      "@value": "2021-08-24T12:20:12Z"
    }
  }
]
```


Part V

Validation

VALIDATION

28.1 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.

28.1.1 Well Formed JSON-LD

[SDO Validator](#)

[Structured data Linter](#)

[JSON-LD Playground](#)

28.1.2 Shape Validation

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

28.2 Implementation

Work this implementation can be found in the notebooks section *SHACL testing*.


Part VI

Interfaces

29.1 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](#)

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

SPARQL

30.1 Introduction

This notebook will go over some concepts for querying the graph that is generated by the Ocean InfoHub.

The product of the publishing and indexing process is a graph. This graph follows the W3C Resource Description Framework (RDF). RDF is a data model that can be represented in various text encodings. Among these is the JSON-LD format we have been using in the publishing process.

Not all graph libraries and databases support JSON-LD yet, so as part of the aggregation process one step is to simply translate this format into a format that is supported by the graph database. Typically we use something n-triples or turtle for this. There are many tools for translating RDF from one format to another.

We can load the graphs into a graph database and this was documented in the Aggregation section, so we will not revisit it here. In this notebook, we will load the graph into a local graph database in the Python library that also supports the same query language our graph databases use. This will make the demonstration easier and allow you to download and run this notebook on your own machine if you wish. The query language we will use is SPARQL.

30.2 SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) is the query language that is used to query RDF graphs. It is a W3C recommendation documented at [SPARQL 1.1 Query Language](#).

30.3 Load a Graph

For this demonstration we will be using Python to load the graph and do our queries. However, there are many languages and libraries for working with graphs and SPARQL. Also, you can often query a graph using web UIs in front of your graph database or even with simple direct HTTP requests. Examples of some of these can be found in other sections of the OIH documentation.

```
import rdflib

g = rdflib.Graph()
g.parse("../tooling/notebooks/data/oceanexperts_graph.ttl", format="ttl")
```

```
<Graph identifier=N0ab13478b6604d5fba3237e6070b1fc7 (<class 'rdflib.graph.Graph'>)>
```


LET'S DEFINE OUR FIRST QUERY

This is a simple query that will return all the triples in the graph. Recall an RDF graph is made of triples defined by a subject, predicate and object. You can review RDF at the W3C RDF Primer (<https://www.w3.org/TR/rdf11-concepts/>). SPARQL allows us to do a type of pattern matching on the graph by defining a pattern we are looking for.

Here we define `?s ?p ?o` as the pattern we are looking for. The `?` is a variable that can be used to match any value. In this case, we are looking for all the triples in the graph. We have used `s`, `p` and `o` as simple names for our subject, predicate, object elements of the RDF triple.

Since this would match ALL the values in the graph, which would be a lot, we have added a `LIMIT` option to reduce the number of results returned.

```
s1 = """
    SELECT ?s ?p ?o
    WHERE {
        ?s ?p ?o .
    }
    LIMIT 3
    """
```


RUN THE QUERY

We will run this query, and print out the results

```
from icecream import ic

for row in g.query(s1):
    ic(row.asdict())
```

```
ic| row.asdict(): {'o': rdflib.term.URIRef('https://schema.org/Place'),
                  'p': rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns
↪#type'),
                  's': rdflib.term.BNode('nb8eae3ca1f664b1c9bf65c29290416bfb68')}
ic| row.asdict(): {'o': rdflib.term.Literal(' Wandelaarkaai 7 Oostende Belgium '),
                  'p': rdflib.term.URIRef('https://schema.org/name'),
                  's': rdflib.term.BNode('nb8eae3ca1f664b1c9bf65c29290416bfb1456')}
ic| row.asdict(): {'o': rdflib.term.BNode('nb8eae3ca1f664b1c9bf65c29290416bfb932'),
                  'p': rdflib.term.URIRef('https://schema.org/hasCourseInstance'),
                  's': rdflib.term.BNode('nb8eae3ca1f664b1c9bf65c29290416bfb931')}
```


NICER FORMATTING

We can do a bit of formatting to see our results in a nicer way.

```
for row in g.query(s1):  
    s = row["s"].n3()  
    p = row["p"].n3()  
    o = row["o"].n3()  
  
    ic(s, p, o)
```

```
ic| s: '_:nb8eae3ca1f664b1c9bf65c29290416bfb68'  
    p: '<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>'  
    o: '<https://schema.org/Place>'  
ic| s: '_:nb8eae3ca1f664b1c9bf65c29290416bfb1456'  
    p: '<https://schema.org/name>'  
    o: '" Wandelaarkaai 7 Oostende Belgium "'  
ic| s: '_:nb8eae3ca1f664b1c9bf65c29290416bfb931'  
    p: '<https://schema.org/hasCourseInstance>'  
    o: '_:nb8eae3ca1f664b1c9bf65c29290416bfb932'
```


As mentioned, this intro is based on the nice work done by KGLab both in their library and documentation. We will import `kglab` here and take advantage of some of their nice features. In particular some nice functions to load our graph and query it with results into a `Dataframe`.

```
import kglab

namespaces = {
    "schema": "https://schema.org/",
}

kg = kglab.KnowledgeGraph(
    name = "Demonstration Graph",
    base_uri = "https://oceaninfohub.org/id/",
    namespaces = namespaces,
)

kg.load_rdf("../tooling/notebooks/data/oceanexperts_graph.ttl")
```

```
/home/files/.conda/envs/kglab/lib/python3.8/site-packages/owlrl/__init__.py:177:
↳UserWarning: Code: dateTimeStamp is not defined in namespace XSD
    from . import DatatypeHandling, Closure
/home/files/.conda/envs/kglab/lib/python3.8/site-packages/owlrl/RDFSClosure.py:40:
↳UserWarning: Code: dateTimeStamp is not defined in namespace XSD
    from owlrl.AxiomaticTriples import RDFS_Axiomatic_Triples, RDFS_D_Axiomatic_Triples
/home/files/.conda/envs/kglab/lib/python3.8/site-packages/owlrl/RDFSClosure.py:40:
↳UserWarning: Code: length is not defined in namespace XSD
    from owlrl.AxiomaticTriples import RDFS_Axiomatic_Triples, RDFS_D_Axiomatic_Triples
/home/files/.conda/envs/kglab/lib/python3.8/site-packages/owlrl/RDFSClosure.py:40:
↳UserWarning: Code: maxExclusive is not defined in namespace XSD
    from owlrl.AxiomaticTriples import RDFS_Axiomatic_Triples, RDFS_D_Axiomatic_Triples
/home/files/.conda/envs/kglab/lib/python3.8/site-packages/owlrl/RDFSClosure.py:40:
↳UserWarning: Code: maxInclusive is not defined in namespace XSD
    from owlrl.AxiomaticTriples import RDFS_Axiomatic_Triples, RDFS_D_Axiomatic_Triples
/home/files/.conda/envs/kglab/lib/python3.8/site-packages/owlrl/RDFSClosure.py:40:
↳UserWarning: Code: maxLength is not defined in namespace XSD
    from owlrl.AxiomaticTriples import RDFS_Axiomatic_Triples, RDFS_D_Axiomatic_Triples
/home/files/.conda/envs/kglab/lib/python3.8/site-packages/owlrl/RDFSClosure.py:40:
↳UserWarning: Code: minExclusive is not defined in namespace XSD
    from owlrl.AxiomaticTriples import RDFS_Axiomatic_Triples, RDFS_D_Axiomatic_Triples
/home/files/.conda/envs/kglab/lib/python3.8/site-packages/owlrl/RDFSClosure.py:40:
↳UserWarning: Code: minInclusive is not defined in namespace XSD
    from owlrl.AxiomaticTriples import RDFS_Axiomatic_Triples, RDFS_D_Axiomatic_Triples
/home/files/.conda/envs/kglab/lib/python3.8/site-packages/owlrl/RDFSClosure.py:40:
↳UserWarning: Code: minLength is not defined in namespace XSD
```

(continues on next page)

(continued from previous page)

```
from owlrl.AxiomaticTriples import RDFS_Axiomatic_Triples, RDFS_D_Axiomatic_Triples
/home/files/.conda/envs/kglab/lib/python3.8/site-packages/owlrl/RDFSClosure.py:40:␣
↳UserWarning: Code: pattern is not defined in namespace XSD
    from owlrl.AxiomaticTriples import RDFS_Axiomatic_Triples, RDFS_D_Axiomatic_Triples
/home/files/.conda/envs/kglab/lib/python3.8/site-packages/owlrl/OWLRL.py:53:␣
↳UserWarning: Code: dateTimeStamp is not defined in namespace XSD
    from .XsdDatatypes import OWL_RL_Datatypes, OWL_Datatype_Subsumptions
/home/files/.conda/envs/kglab/lib/python3.8/site-packages/owlrl/OWLRLExtras.py:64:␣
↳UserWarning: Code: dateTimeStamp is not defined in namespace XSD
    from .RestrictedDatatype import extract_faceted_datatypes
```

```
<kglab.kglab.KnowledgeGraph at 0x7f16ee072fa0>
```

PREFIX

Let's do another query. We will use the PREFIX keyword to define a prefix for the namespace. This is a way to leverage namespace in our query. Note the use of schema:name vs the <https://schema.org/location>. The latter is a URI and the former is a prefix. The prefix is defined in the PREFIX keyword. The schema:name is the name of the property that we are querying, the prefix can expand it to a URI. The <https://schema.org/location> is the URI of the property directly, without use of a prefix. Both work, the prefix can simply make life easier for you and also help avoid hard to find typos where you may miss-type a URI.

Let's do a query to see all the location names in our group for training.

```
s2 = """
PREFIX schema: <https://schema.org/>
SELECT DISTINCT ?locname
  WHERE {
    ?s rdf:type <https://schema.org/Course> .
    ?s <https://schema.org/location> ?location .
    ?location schema:name ?locname .
  }
"""
```

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

df = kg.query_as_df(s2)
df.head(5)
```

	locname
0	Lima Peru
1	Qingdao China
2	Russia
3	Playa del Secreto Puerto Morelos, Mexico
4	University of Ghent Ghent Belgium

LOCATION COUNTS

The above is nice, but it doesn't tell us much about the number of events in each location. We can modify our query to do this. We can leverage the COUNT and GROUP BY capacity of SPARQL to do this. We can then add in ORDER BY to sort the results.

```
s3 = """
SELECT DISTINCT ?locname (COUNT(?locname) AS ?count)
  WHERE {
    ?s rdf:type <https://schema.org/Course> .
    ?s <https://schema.org/location> ?location .
    ?location <https://schema.org/name> ?locname .
  }
GROUP BY ?locname
ORDER BY DESC(?count)
"""
```

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

df = kg.query_as_df(s3)
df.head(5)
```

	locname	count
0	UNESCO/IOC Project Office for IODE Wandelaark...	28
1	Russia	27
2	UNESCO/IOC Project Office for IODE Wandelaark...	22
3	Wandelaarkaai 7 8400 Oostende Belgium	13
4	Belgium	10

FILTER

Graphs are great for showing relations between objects but our approaches to searching or selecting data can often leverage pattern matching on text strings. We do have some tools in SPARQL for this in the form of FILTER. We can use regex pattern in the FILTER to match text. Let's use this to find all the locations with "Peru" in the name.

In the following we will do a simple FILTER with a regular expression, but you can also do things like:

```
SELECT DISTINCT ?s
WHERE {
    ?s rdf:type <https://schema.org/Course> .
    FILTER NOT EXISTS { ?s <https://schema.org/hasCourseInstance> ?instance . }
}
```

which in a SPARQL query would require that a hasCourseInstance property is not defined for the Course. That is, Courses that have had no instances made yet.

```
s4 = """
SELECT DISTINCT ?locname (COUNT(?locname) AS ?count)
WHERE {
    ?s rdf:type <https://schema.org/Course> .
    ?s <https://schema.org/location> ?location .
    ?location <https://schema.org/name> ?locname .
    FILTER regex(?locname, ".Peru.", "i") .
}
GROUP BY ?locname
"""
```

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

df4 = kg.query_as_df(s4)
df4.head(5)
```

	locname	count
0	Lima Peru	1
1	Dirección Hidrografía y Navegación de la Arma...	1
2	Hotel Palmetto Lima Perú Peru	1

CHAPTER THIRTYEIGHT

PERU

So we see we have three entries in our graph that have Peru in the name. This is where connecting our group to a shared concept of what Peru is like the WikiData entry for Peru at <https://www.wikidata.org/wiki/Q419> would help. That's a topic for another day.

DATES

Let's look at the dates that are available in the graph.

```
s5 = """
SELECT DISTINCT ?sdate (COUNT(?sdate) AS ?count)
WHERE {
    ?s rdf:type <https://schema.org/Course> .
    ?s <https://schema.org/hasCourseInstance> ?instance .
    ?instance <https://schema.org/startDate> ?sdate .
}
GROUP BY ?sdate
"""
```

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

df4 = kg.query_as_df(s5)
df4.head(5) # show some results
```

	sdate	count
0	2020-06-08	1
1	2011-05-03	1
2	2012-07-16	1
3	2007-07-12	1
4	2019-04-02	1

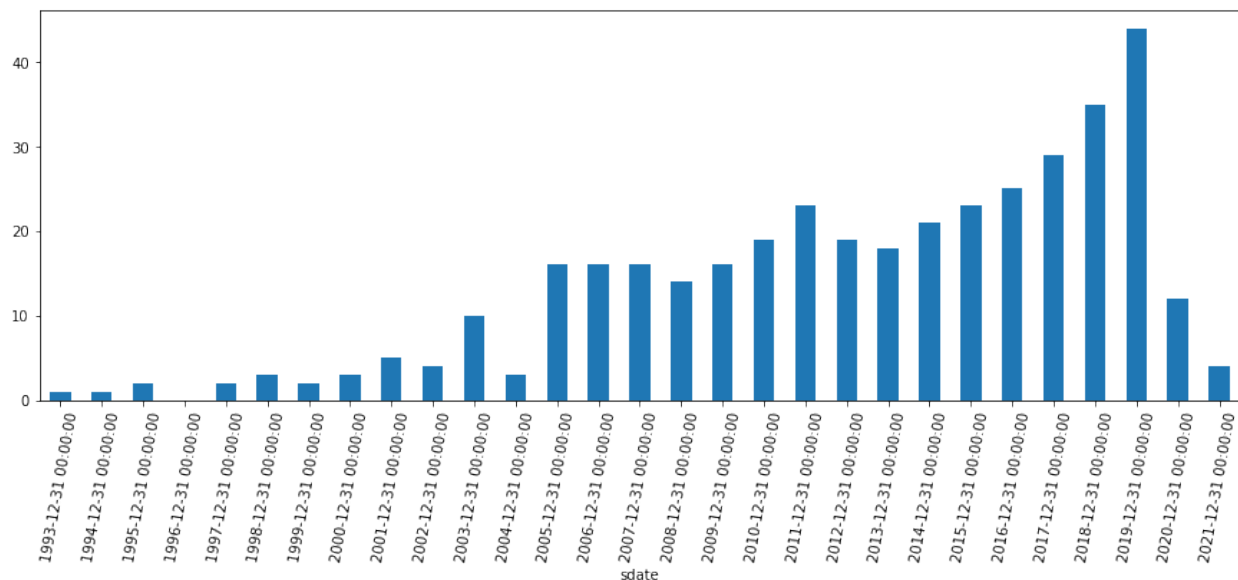
A BIT OF PANDAS

Here we will mix in a bit of Pandas to convert our dates to Pandas dates, group and plot. This is not SPARQL but rather just some post processing in Pandas. It does still give us an idea of some of the information we can extract from our graph.

Also, when the graph is composed of multiple source we can still conduct the same search across a shared concept like <https://scheme.org/Course> with CourseInstance connected to a date via startDate.

This shared approach by Ocean InfoHub is developed to support the shared data and integration goals among the participants.

```
df4['count'] = df4["count"].astype(int) # convert count c to int
df4pd = df4.to_pandas() # convert cudf dataframe to pandas dataframe
df4pd['sdate'] = pd.to_datetime(df4pd['sdate'], format='%Y-%m-%d') # convert date to
↳datetime
courseByYear = df4pd.groupby(pd.Grouper(key='sdate', freq='Y')).size() # group by year
ax = courseByYear.plot.bar(rot=80, stacked=True, figsize=(15, 5)) # plot
```



MORE ON QUERY

Here we have gone through an introduction to the SPARQL query language and how to use it to query our graph. We can look at the SPARQL we use on the larger Ocean Infohub graph.

REFERENCES

This is a just a quick overview of the SPARQL language. You can find a large amount of information online.

- https://derwen.ai/docs/kg1/ex4_0/
- <https://www.stardog.com/tutorials/getting-started-1/>
- <https://www.dataversity.net/introduction-to-sparql/>
- <https://www.slideshare.net/olafhartig/an-introduction-to-sparql>

OIH SPARQL

43.1 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 Creative-works, but to the identity element for People and Organizations.

```
1 prefix prov: <http://www.w3.org/ns/prov#>
2 PREFIX con: <http://www.ontotext.com/connectors/lucene#>
3 PREFIX luc: <http://www.ontotext.com/owlim/lucene#>
4 PREFIX con-inst: <http://www.ontotext.com/connectors/lucene/instance#>
5 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
6 PREFIX schema: <https://schema.org/>
7 PREFIX schemaold: <http://schema.org/>
8 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
9 SELECT DISTINCT ?g ?s ?wat ?orgname ?domain ?type ?score ?name ?url ?lit ?
  ?description ?headline
10 WHERE
11   ?lit bds:search "coral" .
12   ?lit bds:matchAllTerms "false" .
13   ?lit bds:relevance ?score .
14   graph ?g {
15     ?s ?p ?lit .
16     ?s rdf:type ?type .
17     OPTIONAL { ?s schema:name ?name . }
18     OPTIONAL { ?s schema:headline ?headline . }
19     OPTIONAL { ?s schema:url ?url . }
20     OPTIONAL { ?s schema:description ?description . }
21   }
22   ?sp prov:generated ?g .
23   ?sp prov:used ?used .
24   ?used prov:hadMember ?hm .
25   ?hm prov:wasAttributedTo ?wat .
26   ?wat rdf:name ?orgname .
27   ?wat rdfs:seeAlso ?domai
28 }
29 ORDER BY DESC(?score)
30 LIMIT 30
31 OFFSET 0
```

43.2 Lines 11-13

It should be noted that the above SPARQL is not standards compliant. It leverages some vendor specific syntax that is not part of the SPARQL standard. This is not uncommon as groups will often add their own syntax to offer additional functionality.

A common one is what is seen here where a full text index is leveraged to allow for more complex and faster searches than can be done with FILTER regex. These three lines will only work in the current OIH triplestore (Blazegraph). Other triplestores like Jena and others do similar built in function extensions.

43.3 Lines 17-20

These lines demonstrate the use of the OPTIONAL keyword. These triples are not required to be present in a resource. If they are, they will be returned.

43.4 Lines 22-27

These lines are standard SPARQL but are searching across triples not from the provider graphs. Rather, they are looking at triples generated by the OIH indexing program used, Gleaner.

Note, that Gleaner is not a dependency of this project and other indexing approaches and software could be used. As pointed out in the documentation, this approach is based on structured data on the web and web architecture approaches. So, any indexing system following those approaches can be used.

These triples are used to track the indexing event and the sources indexed. It provides some additional provenance to the information collected, but does not change or even extend what the providers are publishing.

As such, these statements could be removed and all that would be lost of indexing activity information.

43.5 Lines 29-31

These lines represent three specific SPARQL parameters.

First is the ORDER BY directive. This is used to order the results by one of the returned variables. In this case we are using the ?score variable which comes from the vendor specific syntax noted in lines 11-13. This score is the ranking score for a resource search against the full text index. However, this could be any variable coming from standards compliant SPARQL calls too. Sorting can be done on alphanumeric values in ascending (ASC) or descending (DESC) order.

The LIMIT is used to limit the number of results returned. We follow this with, OFFSET which is used to skip the first n results. These two are useful for pagination when combined with the ORDER BY directive.

44.1 About

The Ocean InfoHub graph is accessible through several approaches. The graph is implemented in RDF and expressed through a standards compliant triplestore.

This triplestore exposes a SPARQL endpoint that can be queried using the [SPARQL 1.1 Query Language](#). To do this you can visit a web based query interface as discussed in other sections of this document.

It is also possible to access this service following RESTful principles.

44.2 SPARQL HTTP Protocol

One approach to this RESTful approach is to use the [SPARQL 1.1 Graph Store HTTP Protocol](#). This is a simple HTTP protocol that allows you to query the graph using SPARQL and obtain the results in JSON.

The OIH triplestore exposes the graph following this pattern for queries.

44.2.1 Examples

```
curl -X POST https://graph.collaborium.io/blazegraph/namespace/aquadocs/sparql --data-  
urlencode 'query=SELECT * { ?s ?p ?o } LIMIT 1' -H 'Accept:application/sparql-  
results+json'
```

If run this from the command line we will get something like the following.

```
❏ curl -X POST https://graph.collaborium.io/blazegraph/namespace/aquadocs/sparql --  
data-urlencode 'query=SELECT * { ?s ?p ?o } LIMIT 1' -H 'Accept:application/sparql-  
results+json'
```

```
{  
  "head" : {  
    "vars" : [ "s", "p", "o" ]  
  },  
  "results" : {  
    "bindings" : [ {  
      "s" : {  
        "type" : "uri",  
        "value" : "https://hdl.handle.net/1834/10030"  
      },  
      "p" : {  
        "type" : "uri",  
        "value" : "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"  
      },  
      "o" : {  
        "type" : "uri",  
        "value" : "http://www.w3.org/2002/07/owl#Thing"  
      }  
    ]  
  }  
}
```

(continues on next page)

(continued from previous page)

```
"p" : {
  "type" : "uri",
  "value" : "https://schema.org/propertyID"
},
"o" : {
  "type" : "literal",
  "value" : "https://hdl.handle.net/"
}
} ]
}
```

While this is unlikely how you will want to interact with the graph, it demonstrates the HTTP based access API that can be used in tools, notebooks or other applications.

This is the same basic approach the used in the web client. There the axios library (<https://axios-http.com/>) is used with a code snippet like:

```
axios.get(url.toString())
  .then(function (response) {
    // handle success
    console.log(response);
    const el = document.querySelector('#container2');
    render(showresults(response), el);
  })
  .catch(function (error) {
    // handle error
    console.log(error);
  })
  .then(function () {
    // always executed
  });
```

Part VII

Appendix

45.1 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.

45.2 Known Issues

45.2.1 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.

45.3 References

A broad collection of references.

45.3.1 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

45.3.2 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

45.3.3 Content Management and Web Server support

Drupal Support

Wordpress Claim Review

45.3.4 Organizations

Google Open Source

DataCommons & DataCommons REST

45.3.5 Indexers

Gleaner BMUSE

45.3.6 Graph tools

Wikipedia SPARQL implementation list

RDFlib

Any23

rdfjs

shemaram

shemarama demo

validatingrdf

Structured Data Linter

45.3.7 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

45.3.8 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

Yeast Structured Data Guide

Schema App

Springer: Scigraph

iPhylo Biodiversity KG

ozymandias

RDA group meeting notes

RDA Plenary meeting

45.4 Registries

45.4.1 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.

Datcite

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](#)

45.4.2 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.

45.4.3 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.

45.4.4 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.

45.5 Controlled Vocabularies

45.5.1 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.

45.5.2 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](#)

BIBLIOGRAPHY

- [1] Omar Benjelloun, Shiyu Chen, and Natasha Noy, editors. *Google Dataset Search by the Numbers*, 2020. URL: <https://arxiv.org/abs/2006.06894>.
- [2] Google search central dataset. <https://developers.google.com/search/docs/advanced/structured-data/dataset>. URL: <https://developers.google.com/search/docs/advanced/structured-data/dataset>.