OGC® DOCUMENT: 22-043

External identifier of this OGC® document: http://www.opengis.net/doc/PER/eoep-Hack2018

Open Geospatial Consortium

JOINT OGC AND ISO CODE SPRINT 2022 SUMMARY ENGINEERING REPORT

ENGINEERING REPORT

DRAFT

Submission Date: 2018-12-20
Approval Date: 2018-07-10
Publication Date: 2029-06-12
Editor: Gobe Hobona, Joana Simoes

License Agreement

Copyright notice

Note

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

CONTENTS

LIST OF FIGURES

I EXECUTIVE SUMMARY

Over the past two decades, standards such as ISO 19115:2003 and the OGC Catalogue Services for the Web (CSW) have been integrated into several Spatial Data Infrastructure (SDI) initiatives at national and international level. These standards leveraged the Extensible Markup Language (XML) which, at the time, was the primary encoding for data exchange across much of Information Technology. In recent times, however, the increasing use of JavaScript Object Notation (JSON) and the uptake of Web Application Programming Interface (API) has meant that modernisation of metadata and catalogues approaches is necessary.

In November 2021, the Open Geospatial Consortium (OGC) and the International Organization for Standardization (ISO) held their first joint code sprint. The success of that first joint code sprint provided the foundation for a second joint code sprint. This Engineering Report (ER) summarizes the main achievements of the second joint code sprint, conducted between September 14th and 16th, 2022. The second code sprint, named the 2022 Joint OGC and ISO Code Sprint — The Metadata Code Sprint, served to accelerate the support of open geospatial standards that relate to geospatial metadata and catalogues. The code sprint was sponsored by Ordnance Survey (OS) at the Gold-level and Geonovum at the Silver-level. The code sprint was held as a hybrid event, with the face-to-face element hosted at the Geovation Hub in London, United Kingdom.

The code sprint focused on the following group of APIs and encodings:

- OGC API — Records candidate standard
- ISO 19115 metadata Standards (i.e., ISO 19115-1, ISO 19115-2, ISO 19115-3)
- OGC Features and Geometries JSON (JSON-FG) candidate standard
- Spatio-Temporal Asset Catalog (STAC), which leverages the OGC API — Features Standard

The OGC is an international consortium of more than 500 businesses, government agencies, research organizations, and universities driven to make geospatial (location) information and services FAIR — Findable, Accessible, Interoperable, and Reusable. The consortium consists of Standards Working Groups (SWGs) that have responsibility for designing a candidate standard prior to approval as an OGC Standard and for making revisions to an existing OGC Standard. The sprint objectives for the SWGs were to:

- Develop prototype implementations of OGC standards, including implementations of draft OGC Application Programming Interface (API) standards
- Test the prototype implementations
- Provide feedback to the Editor about what worked and what did not
- Provide feedback about the specification document

Technical Committee 211 (TC 211) of ISO is responsible for the development and publication of standards that relate to geographic information. As with other ISO committees, TC 211 consists of member nations, as well as liaison partner organizations. TC 211 and OGC have a liaison partnership that enables the organizations to participate in each others activities and also to collaborate of standards development initiatives. The sprint objectives for ISO/TC 211 were:

- Support the development of ISO Standards
- Fix open issues
- Develop new features
- Encourage the implementation of ISO Standards

This engineering report makes the following recommendations for future innovation work items:

- Initiatives to facilitate implementation of JSON-FG (e.g. 3D, cadastral data, etc)
- Initiatives to facilitate implementation of catalogues
- Prototyping of tools for creating metadata (e.g. the automated STAC metadata crawler demonstrated during the sprint)

The engineering report also makes the following recommendations for things that the Standards Working Groups should consider:

- Outreach for JSON-FG
- Code Sprint for designing profiles of JSON-FG for different communities of interest
- Documentation of the different roles of catalogues and API, as well as guidance on when to use them
- Code Sprint on versioning, possibly combining an OGC API Features Part 4 with OGC API Records
- Exploring how to move GeoDCAT forward within OGC

II KEYWORDS

The following are keywords to be used by search engines and document catalogues.

hackaton, application-to-the-cloud, testbed, docker, web service

III SECURITY CONSIDERATIONS

No security considerations have been made for this document.

IV SUBMITTERS

All questions regarding this document should be directed to the editor or the contributors:

NAME	ORGANIZATION	ROLE
Gobe Hobona	Open Geospatial Consortium	Editor
Joana Simoes	Open Geospatial Consortium	Editor
Panagiotis Vretanos	CubeWerx Inc.	Contributor
Núria Julià Selvas	UAB-CREAF	Contributor
Joan Maso	UAB-CREAF	Contributor
Clemens Portele	interactive instruments GmbH	Contributor
Matthias Mohr	WWU Münster	Contributor
Tom Kralidis	Meteorological Service of Canada	Contributor
Byron Cochrane	OpenWork Ltd	Contributor
Andreas Matheus	Secure Dimensions	Contributor
Jeroen Ticheler	GeoCat	Contributor

V ABSTRACT

The subject of this Engineering Report (ER) is a code sprint that was held from the 14th to the 16th of September 2022 to advance open standards that relate to geospatial metadata and catalogues. The code sprint was hosted by the Open Geospatial Consortium (OGC) and the International Organization for Standardization (ISO). The code sprint was sponsored by Ordnance Survey (OS) and Geonovum, and held as a hybrid event with the face-to-face element hosted at the Geovation Hub in London, United Kingdom.

1

SCOPE

1 SCOPE

This Engineering Report (ER) summarizes the main achievements of the Joint OGC and ISO Code Sprint, conducted between September 14th and 16th, 2022. Sponsored by Ordnance Survey (OS) and Geonovum, the code sprint was hosted by the OGC and ISO with the goal of advancing the development and implementation open standards that relate to geospatial metadata and catalogues.

A Code Sprint is a collaborative and inclusive event driven by innovative and rapid programming with minimal process and organization constraints to support the development of new applications and open standards. Code Sprints experiment with emerging ideas in the context of geospatial standards, help improve interoperability of existing standards by experimenting with new extensions or profiles, and are used for building proofs of concept to support standards development activities and enhancement of software products.

2

NORMATIVE REFERENCES

2 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Open API Initiative: OpenAPI Specification 3.0.3, https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md

Berners-Lee, T., Fielding, R., Masinter, L: IETF RFC 3896, Uniform Resource Identifier (URI): Generic Syntax, https://tools.ietf.org/rfc/rfc3896.txt

ISO: ISO 19115-1, *Geographic information — Metadata — Part 1: Fundamentals*. International Organization for Standardization, Geneva https://www.iso.org/standard/53798.html.

ISO: ISO 19115-2, *Geographic information — Metadata — Part 2: Extensions for acquisition and processing*. International Organization for Standardization, Geneva https://www.iso.org/standard/67039.html.

ISO: ISO/TS 19115-3, *Geographic information — Metadata — Part 3: XML schema implementation for fundamental concepts*. International Organization for Standardization, Geneva https://www.iso.org/standard/32579.html.

3

TERMS, DEFINITIONS AND ABBREVIATED TERMS

3 TERMS, DEFINITIONS AND ABBREVIATED TERMS

This document uses the terms defined in OGC Policy Directive 49, which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this document and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications (OGC 08-131r3), also known as the ‘ModSpec’. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

For the purposes of this document, the following additional terms and definitions apply.

3.1. API

An Application Programming Interface (API) is a standard set of documented and supported functions and procedures that expose the capabilities or data of an operating system, application, or service to other applications (adapted from ISO/IEC TR 13066-2:2016).

3.2. coordinate reference system

A coordinate system that is related to the real world by a datum term name (source: ISO 19111)

3.3. OpenAPI Document

A document (or set of documents) that defines or describes an API. An OpenAPI definition uses and conforms to the OpenAPI Specification (https://www.openapis.org)

3.4. **Metadata**

information about a resource [source: ISO 19115-1:2014, Amendment 2]

3.5. **Web API**

API using an architectural style that is founded on the technologies of the Web [source: OGC API — Features — Part 1: Core]

3.6. Abbreviated terms

API	Application Programming Interface
CRS	Coordinate Reference System
GIS	Geographic Information System
OGC	Open Geospatial Consortium
OWS	OGC Web Services
REST	Representational State Transfer

4

HIGH-LEVEL ARCHITECTURE

4 HIGH-LEVEL ARCHITECTURE

As illustrated in Figure 1, the sprint architecture was designed with the view of enabling client applications to connect to different servers that implement open geospatial standards that relate to metadata and catalogues. Implementations of JSON-FG, ISO 19115, STAC, and OGC API — Records were deployed in participants' own infrastructure in order to build a solution with the architecture shown below in Figure 1.

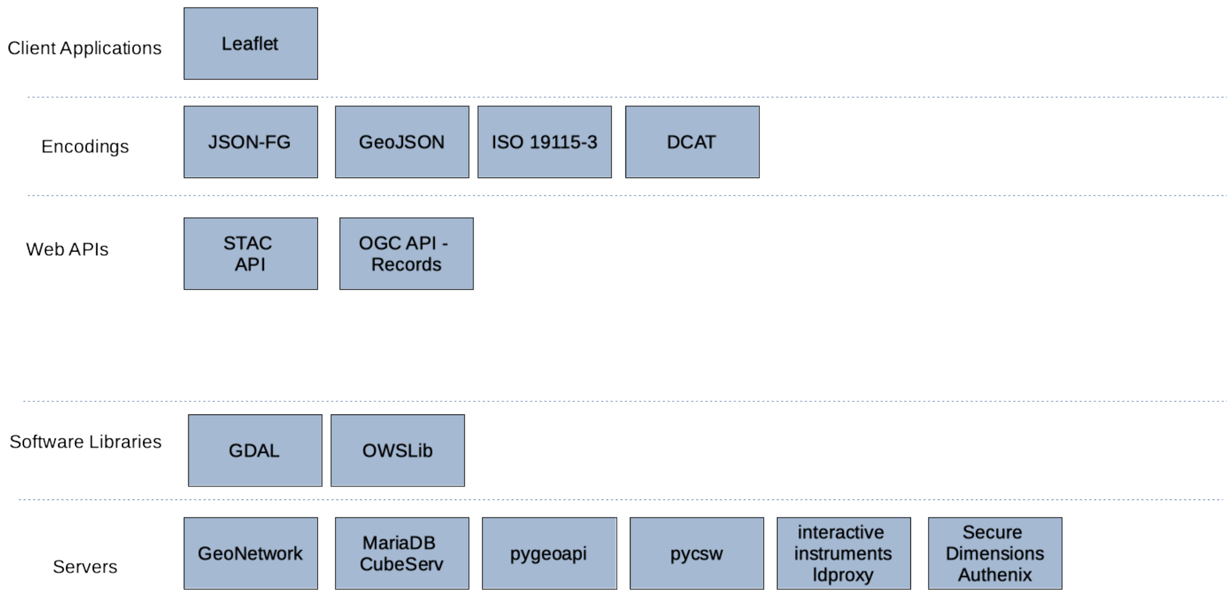

Figure 1 — High Level Overview of the Sprint Architecture

The rest of this section describes the software deployed and standards implemented during the code sprint.

4.1. Approved and Draft Standards

This section describes the approved and draft standards implemented during the code sprint.

4.1.1. OGC API — Records

The OGC API — Records candidate standard provides discovery and access to metadata records about resources such as features, coverages, tiles / maps, models, assets, services or widgets. The candidate standard enables the discovery of geospatial resources by standardizing the way collections of descriptive information about the resources (metadata) are exposed. The candidate standard also enables the discovery and sharing of related resources that may be referenced from geospatial resources or their metadata by standardizing the way all kinds of records are exposed and managed. OGC API — Records can be considered the future successor to the widely implemented Catalogue Services for the Web (CSW) standard.

The candidate standard specifies the information content of a record. A record contains summary descriptive information about a resource that a provider wishes to make discoverable. Records are organized into collections. A record represents resource characteristics that can be presented for evaluation and further processing by both humans and software. Examples of resources include: a data collection, a service, a process, a style, a code list, an Earth observation asset, a machine learning model, a code list and so on.

4.1.2. JSON-FG

JSON-FG (Features and Geometry JSON) extends GeoJSON to support a limited set of additional capabilities that are out-of-scope for GeoJSON, but that are important for a variety of use cases involving feature data.

This candidate standard extends GeoJSON in the following minimal ways:

- The ability to use Coordinate Reference Systems (CRSs) other than WGS 84
- The ability to use non-Euclidean metrics, in particular ellipsoidal metrics
- Support for solids and prisms as geometry types
- The ability to encode temporal characteristics of a feature
- The ability to declare the type and the schema of a feature

Where it is possible to represent information as GeoJSON, the candidate standard allows for the information to be represented as such. Additional information is supported through the use of additional members which are represented using keys that should not conflict with those specified by GeoJSON. This means that JSON-FG enabled clients will be able to parse and understand the additional members.

4.1.3. ISO 19115

ISO 19115 Standards define the schema required for describing geographic information and services by means of metadata. Metadata is information about a resource such as a dataset, web service, or API. This multi-part International Standard is applicable to the cataloguing of datasets, clearinghouse activities, geographic datasets, dataset series, individual geographic features, and feature properties.

The individual parts of ISO 19115 that each serve as an approved standard include:

- ISO 19115-1:2014 defines the schema required for describing geographic information and services by means of metadata.

- ISO 19115-2:2019 extends ISO 19115-1:2014 by defining the schema required for an enhanced description of the acquisition and processing of geographic information, including imagery.
- ISO/TS 19115-3:2016 defines an integrated XML implementation of ISO 19115-1, ISO 19115-2, and concepts from ISO/TS 19139

4.1.4. STAC

The SpatioTemporal Asset Catalog (STAC) is a specification that offers a language for describing geospatial information, so it can be worked with, indexed, and discovered. The STAC API offers an interface that implements OGC API — Features. Although STAC has been developed outside of the OGC, in the long term it is envisaged that the STAC API specification will be developed into an OGC Community Standard that implements OGC API building blocks that are relevant for the STAC use cases.

STAC is a multi-part specification that includes the following consituent parts:

- STAC Item is a representation of a single spatio-temporal asset, encoded as a GeoJSON feature with datetime and links properties.
- STAC Catalog is a JSON-encoded representation of links that provides a structure for organizing and browsing STAC Items.
- STAC Collection extends the STAC Catalog to offer additional information such as the extents, keywords, license, providers, and other elements that describe STAC Items that grouped within the Collection.
- STAC API provides a RESTful interface that conforms to the OGC API — Features standard, described in an OpenAPI definition document, and supports search of STAC Items.

Each of the above-listed parts can be used on its own, however the parts have been designed to offer optimal capabilities when used together.

4.2. Open Source Software Projects

This section describes open source software products that were deployed during the code sprint.

4.2.1. OSGeo GeoNetwork

GeoNetwork is a catalog application for managing spatially referenced resources. It provides metadata editing and search functions as well as an interactive web map viewer.

GeoNetwork is used for (meta)-data management by governments, local communities and private sector. It is also used to discover geospatial (and other) (open) data supporting multiple metadata standards and multiple catalog interfaces.

OGC standards have been core to the GeoNetwork project and the community is now working on the implementation of the OGC API Records specification.

4.2.2. ldproxy

ldproxy is an implementation of the OGC API family of specifications, inspired on the W3C/OGC Spatial Data on the Web Best Practices. ldproxy is developed by interactive instruments GmbH, written in Java (Source Code) and is typically deployed using docker (DockerHub). The software originally started in 2015 as a Web API for feature data based on WFS 2.0 capabilities. In addition to the JSON/XML encodings, an emphasis is placed on an intuitive HTML representation.

The current version supports WFS 2.0 instances as well as PostgreSQL/PostGIS databases as backends. It implements all conformance classes and recommendations of "OGC API — Features — Part 1: Core" and "OGC API — Features- Part 2: Coordinate Reference Systems By Reference", as well as the draft extensions (that is Part 3 and Part 4). ldproxy also has draft implementations for additional resource types (Tiles, Styles).

4.2.3. OSGeo Leaflet

Leaflet is an open-source JavaScript library for mobile-friendly interactive maps. It works across all major desktop and mobile platforms, can be extended with a variety of plugins, and offers a well-documented API.

4.2.4. OSGeo pygeoapi

pygeoapi is a Python server implementation of the OGC API suite of standards. The project emerged as part of the next generation OGC API efforts in 2018 and provides the capability for organizations to deploy a RESTful OGC API endpoint using OpenAPI, GeoJSON, and HTML. pygeoapi is open source and released under an MIT license. pygeoapi is an official OSGeo Project as well as an OGC Reference Implementation.

pygeoapi supports numerous OGC API standards. The official documentation provides an overview of all supported standards.

4.2.5. OSGeo pycsw

pycsw is an OGC API — Records and OGC CSW server implementation written in Python. Started in 2010 (more formally announced in 2011), pycsw allows for the publishing and discovery of geospatial metadata via numerous APIs (CSW 2/CSW 3, OpenSearch, OAI-PMH, SRU), providing a standards-based metadata and catalogue component of spatial data infrastructures. pycsw is Open Source, released under an MIT license, and runs on all major platforms (Windows, Linux, Mac OS X). pycsw is an official OSGeo Project as well as an OGC Reference Implementation.

pycsw supports numerous metadata content and API standards, including OGC API — Records — Part 1.0: Core and its associated specifications. The official documentation provides an overview of all supported standards.

4.2.6. OSGeo pygeometa

pygeometa provides a lightweight and Pythonic approach for users to easily create geospatial metadata in standards-based formats using simple configuration files (affectionately called metadata control files [MCF]). Leveraging the simple but powerful YAML format, pygeometa can generate metadata in numerous standards. Users can also create their own custom metadata formats which can be plugged into pygeometa for custom metadata format output.

For developers, pygeometa provides a Pythonic API that allows developers to tightly couple metadata generation within their systems and integrate nicely into metadata production pipelines.

The project supports various metadata formats out of the box including ISO 19115, the WMO Core Metadata Profile, and the WIGOS Metadata Standard. The project also supports the OGC API — Records core record model as well as STAC (Item).

pygeometa has minimal dependencies (install is less than 50 kB), and provides a flexible extension mechanism leveraging the Jinja2 templating system.

pygeometa is open source and released under an MIT license.

4.2.7. OSGeo OWSLib

OWSLib is a Python client for OGC Web Services and their related content models. The project is an OSGeo Community project and is released under a BSD 3-Clause License.

OWSLib supports numerous OGC standards, including increasing support for the OGC API suite of standards. The official documentation provides an overview of all supported standards.

4.3. Proprietary products

This section describes proprietary software products that were deployed during the code sprint.

4.3.1. MariaDB CubeWerx CubeServ

The CubeWerx server (“cubeserv”) is implemented in C and currently implements the following OGC specifications:

- All conformance classes and recommendations of the OGC API — Features — Part 1: Core standard.
- Multiple conformance classes and recommendations of the draft OGC API — Records — Part 1: Core standard.
- Multiple conformance classes and recommendations of the draft OGC API — Coverages — Part 1: Core standard.
- Multiple conformance classes and recommendations of the OGC API — Processes — Part 1: Core standard.
- Multiple versions of the Web Map Service (WMS), Web Processing Service (WPS), Web Map Tile Service (WMTS) and Web Feature Service (WFS) standards.
- A number of other “un-adopted” OGC web services including the Testbed-12 Web Integration Service, OWS-7 Engineering Report — GeoSynchronization Service, and the Web Object Service prototype.

The cubeserv executable supports a wide variety of back ends including Oracle, MariaDB, SHAPE files, etc. It also supports a wide array of service-dependent output formats (e.g. GML, GeoJSON, Mapbox Vector Tiles, MapMP, etc.) and coordinate reference systems.

5

RESULTS

5 RESULTS

The code sprint included multiple software products and implementations of OGC and ISO Standards. This section presents some of the results from the code sprint.

5.1. Leaflet

One of the contributors of Leaflet implemented support for JSON-FG in the code sprint. Support for JSON-FG in Leaflet was implemented as a plug-in, enabling anyone to integrate the JSON-FG files into a leaflet application.

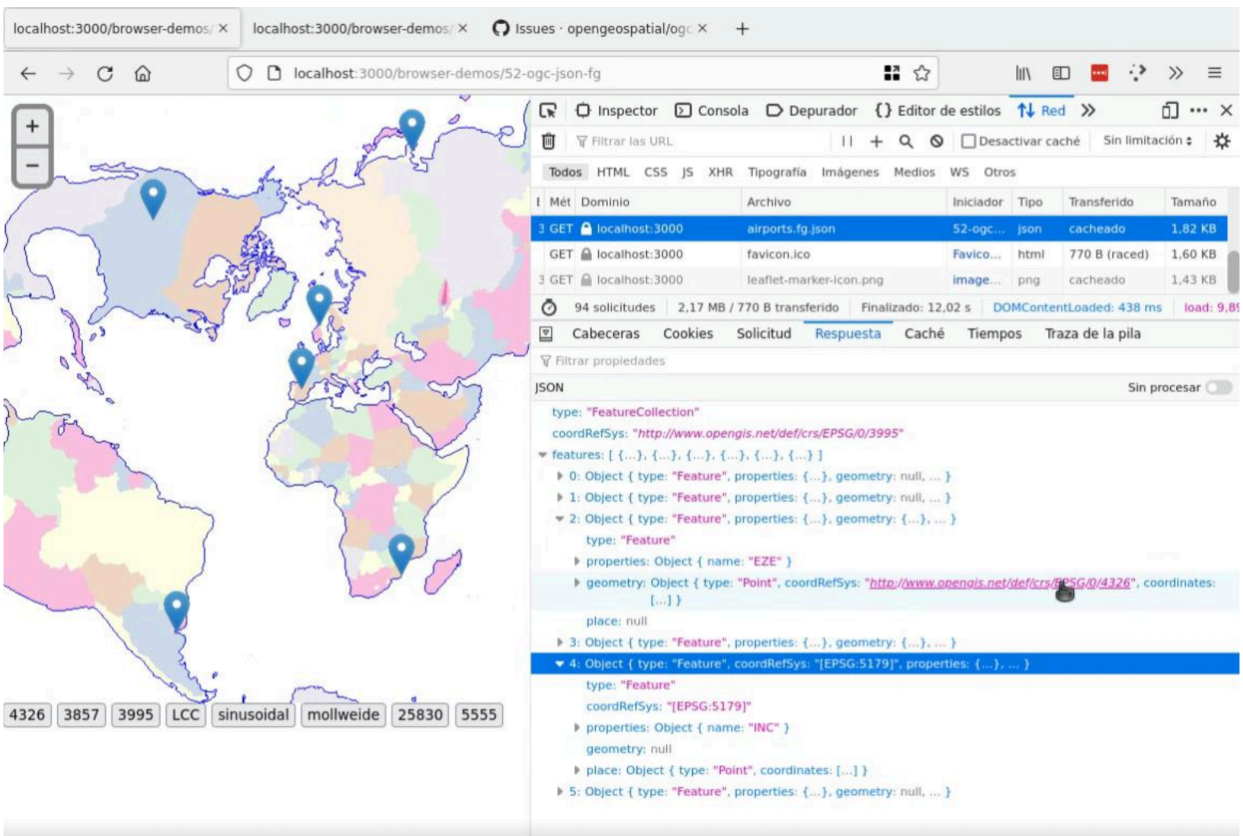

Figure 2 — Screenshot of the leaflet demo

5.2. MariaDB CubeWerx CubeSERV

The participants from CubeWerx worked on their implementation of OGC API – Records and the interoperability with STAC. A screenshot from their prototype implementation is shown in the figure below.

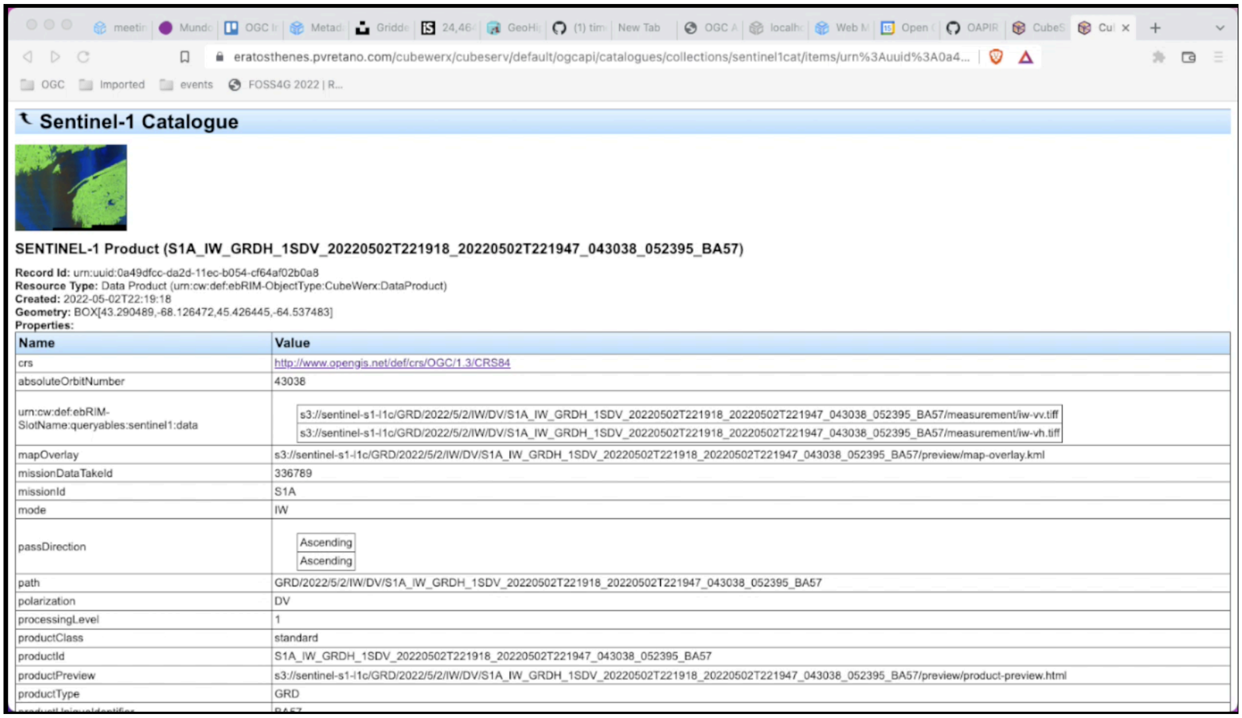

Figure 3 — Screenshot of the CubeWerx CubeSERV demo

5.3. ldproxy

The participants from interactive instruments worked on their implementation of OGC API – Features to prototype support for JSON-FG. A screenshot from their prototype implementation is shown in the figure below.

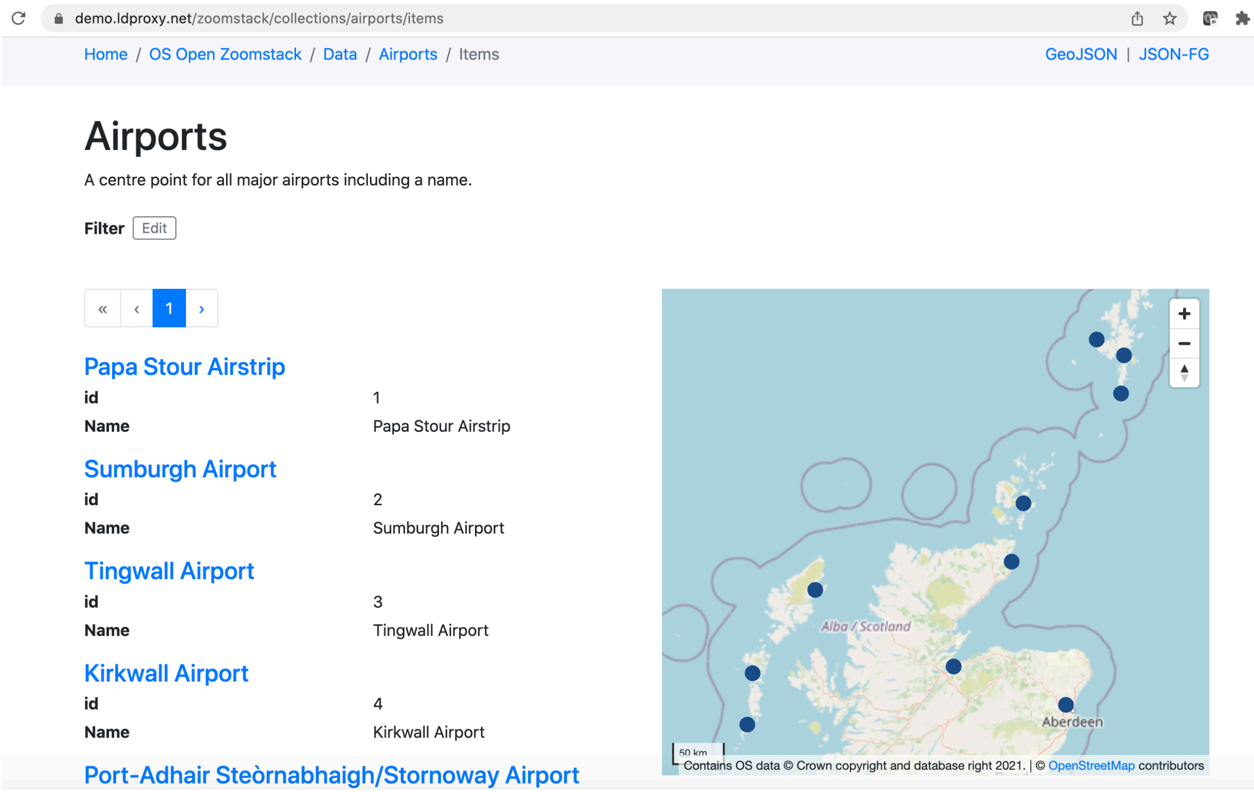

Figure 4 — Screenshot of the CubeWerx CubeSERV demo

5.4. GeoNetwork

GeoNetwork team improved the following topics during the Metadata Code Sprint

5.4.1. ISO 19115

Following discussions with participants, some would like to benefit from the possibility in ISO 19115-3 to describe the dataset's feature catalogue using the following method:

- As a citation in the dataset record (was also available in ISO 19139)
- As an embedded feature catalogue
- As a standalone feature catalogue

GeoNetwork was supporting previous version of ISO 19110 as standalone record and it can be more relevant now to use ISO 19115-3 (which contains the latest version of ISO 19110) for standalone (or embedded) feature catalogue. With that change, users can decide how to relate dataset description and their corresponding feature catalogue using one of the 3 approaches above.

Work in progress here https://github.com/geonetwork/core-geonetwork/pull/6545

5.4.2. OGC API — Records implementation

Output formats improved:

- GeoJSON for /items and /items/uuid
- DCAT for /items (not only for /items/uuid). Improved DCAT conversion.

Added a conformance section.

See https://github.com/geonetwork/geonetwork-microservices/pull/59

Elasticsearch by default return only 10K records. Add support for larger catalogue using track total hits (see https://github.com/geonetwork/geonetwork-microservices/pull/58).

QGIS client interaction was tested and requires a conformance section and GeoJSON output. Work to be continued.

5.5. GeM+

GeM+ is a desktop application developed in C for managing and editing geospatial metadata. The participants from CREAF have worked on the implementation of an OGC API Record client integrated into GeM+. The implemented prototype of GeM+ is able to read metadata in XML format according to ISO 19115 (ISO 19139) and ISO 19115-2 by connecting to OGC API Records. The following figures show several screenshots of the implementation in GeM+.

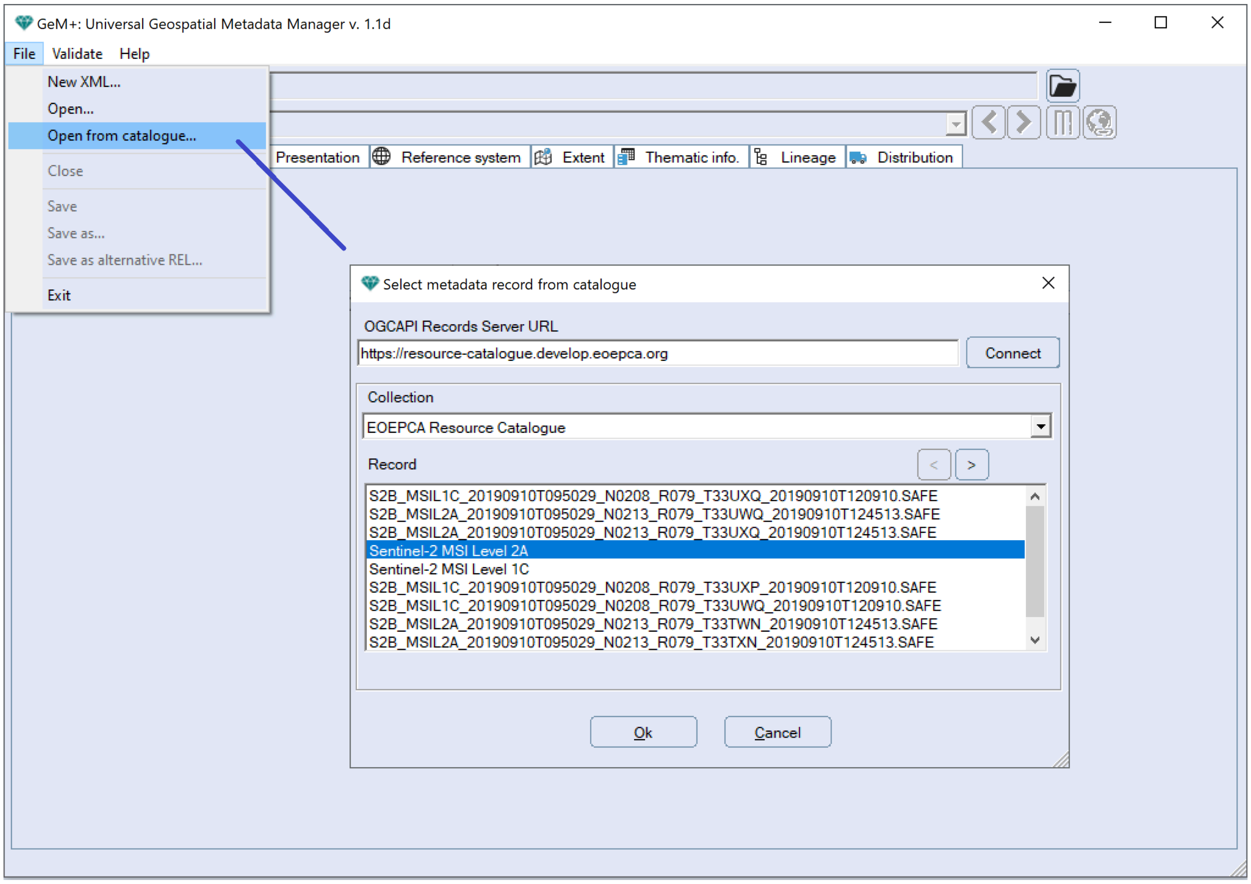
GeM+: Universal Geospatial Metadata Manager v. 1.1d
File
Validate
Help
New XML...
Open...
Open from catalogue...
Close
Save
Save as...
Save as alternative REL...
Exit
Presentation
Reference system
Extent
Thematic info.
Lineage
Distribution
Select metadata record from catalogue
OGCAPI Records Server URL
https://resource-catalogue.develop.eoepca.org
Connect
Collection
EOEPCA Resource Catalogue
Record
S2B_MSIL1C_20190910T095029_N0208_R079_T33UXQ_20190910T120910.SAFE
S2B_MSIL2A_20190910T095029_N0213_R079_T33UWQ_20190910T124513.SAFE
S2B_MSIL2A_20190910T095029_N0213_R079_T33UXQ_20190910T124513.SAFE
Sentinel-2 MSI Level 2A
Sentinel-2 MSI Level 1C
S2B_MSIL1C_20190910T095029_N0208_R079_T33UXP_20190910T120910.SAFE
S2B_MSIL1C_20190910T095029_N0208_R079_T33UWQ_20190910T120910.SAFE
S2B_MSIL2A_20190910T095029_N0213_R079_T33TWN_20190910T124513.SAFE
S2B_MSIL2A_20190910T095029_N0213_R079_T33TXN_20190910T124513.SAFE
Ok
Cancel

Figure 5

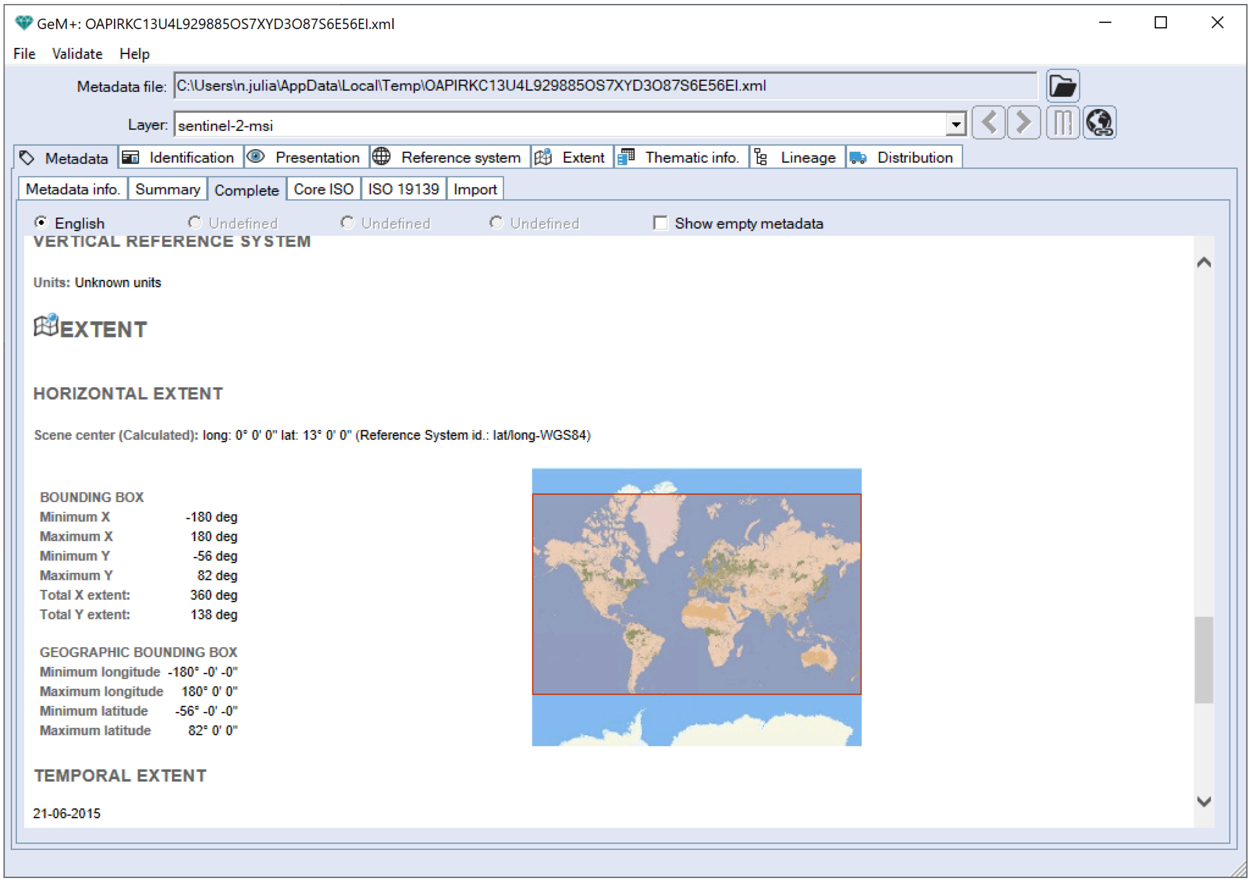

Figure 6

5.6. pygeoapi

The pygeoapi project implemented capability for metadata transactions following the OGC API — Records — Part 4: Create, Replace, Update and Delete draft specification. Support for the "Requirements Class "Create/Replace/Delete" requirements class was implemented, reviewed and approved by the pygeoapi development team. As a result, pygeoapi implementations are now able to easily define "editable" resources for metadata (OGC API — Records) as well as data (OGC API — Features) thanks to the building block approach of OGC APIs.

The official pygeoapi documentation provides more information on how to enable the new functionality. A screenshot of the associated OpenAPI/Swagger interface from the implementation is shown in the figure below.

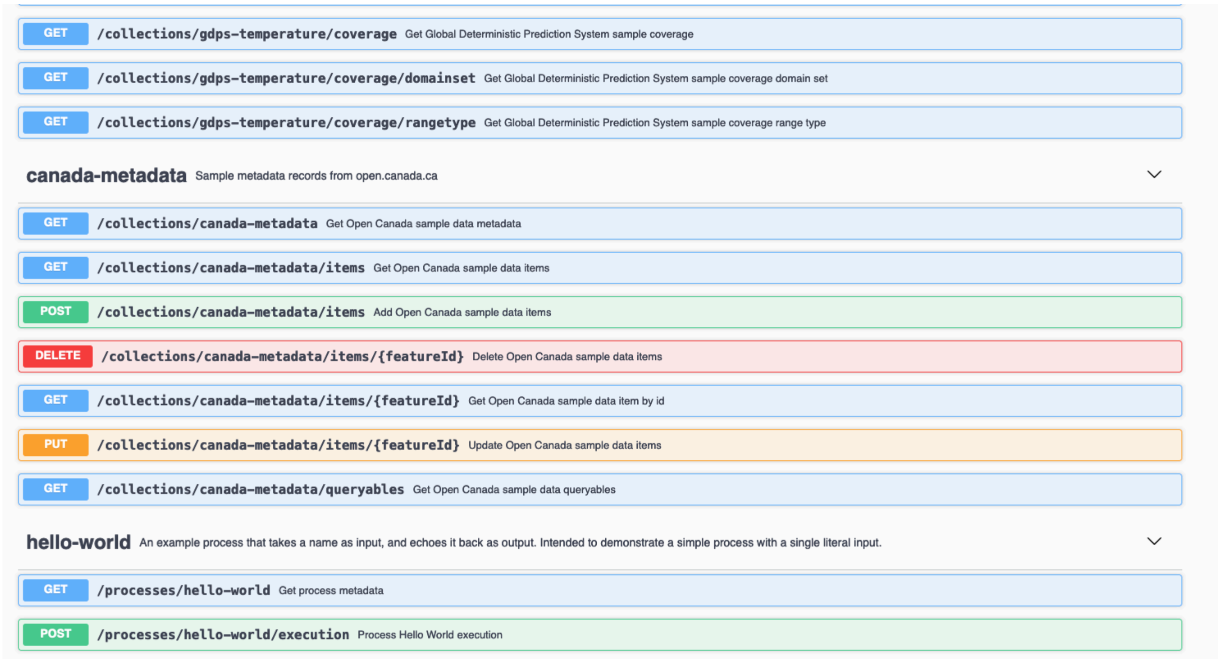

Figure 7 — Screenshot of the pygeoapi transactions demo

pygeoapi also added support for STAC-based link relations (`rel=root`) in support of interoperability with STAC tooling.

5.7. pycsw

5.7.1. Transactions support

The pycsw project implemented capability for metadata transactions following the OGC API — Records — Part 4: Create, Replace, Update and Delete draft specification. Support for the "Requirements Class "Create/Replace/Delete" requirements class was implemented, in review and expected to be included in the main codebase. The functionality leverages pycsw's existing underlying transactional support made available by CSW-T. As a result, pycsw implementations are now able to provide OGC API based transactional support for metadata management via OGC API — Records.

A screenshot of the associated OpenAPI/Swagger interface from the implementation is shown in the figure below.

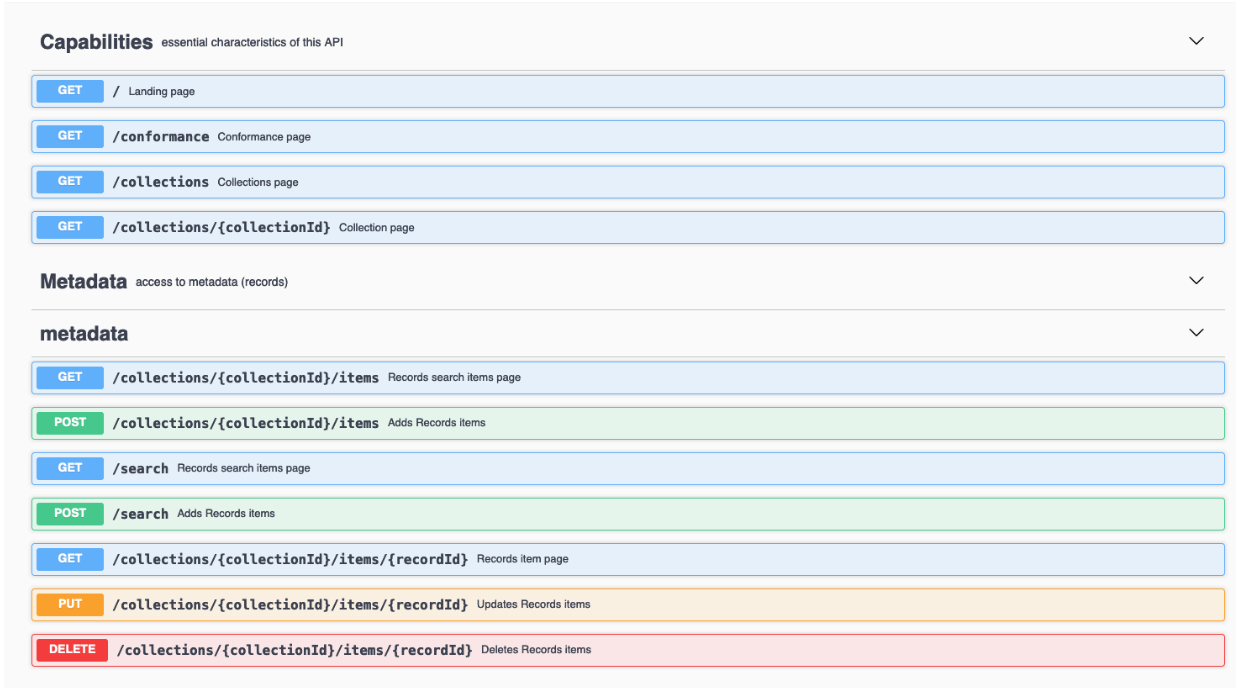

Figure 8 — Screenshot of the pycsw transactions demo

5.7.2. OGC Testbed-18 instance of pycsw

An instance of pycsw, implemented for Testbed-18, was included in the code sprint to support a demonstration of asynchronous catalogues that implement OGC API — Records.

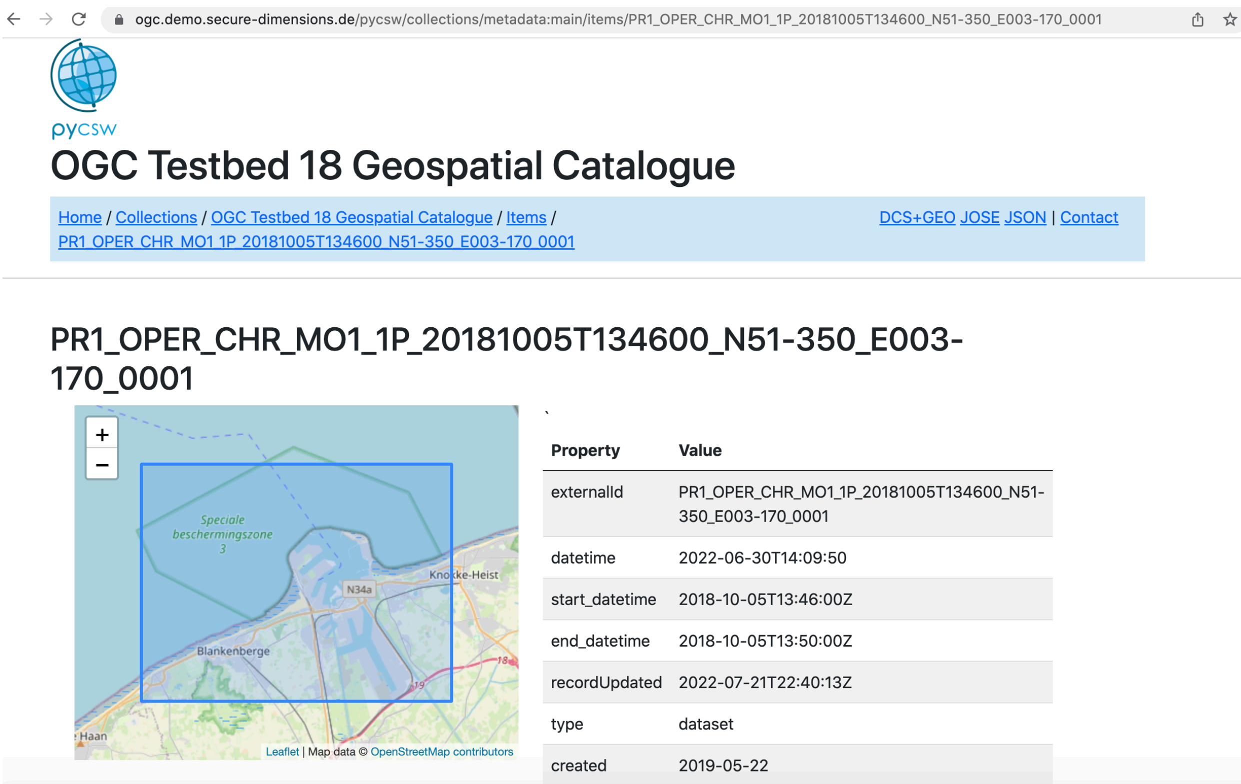

Figure 9 — Screenshot of the instance of pycsw

5.8. owslib

The OWSLib project implemented capability for client transactions following the OGC API — Records — Part 4: Create, Replace, Update and Delete draft specification. Support for the “Requirements Class “Create/Replace/Delete” requirements class was implemented, reviewed and approved by the OWSLib development team. As a result, Python clients can now use OWSLib for simple and Pythonic workflow for resource transactions for metadata (OGC API — Reocrds) as well as data (OGC API — Features).

A screenshot of a sample Python/OWSLib workflow of the new functionality is shown in the figure below.

```python
import json

from owslib.ogcapi.records import Records

record_data = 'sample-record.json'

url = 'http://localhost:8000'
collection_id = 'metadata:main'

r = Records(url)

cat = r.collection(collection_id)

with open(record_data) as fh:
    data = json.load(fh)

identifier = data['id']

r.collection_item_delete(collection_id, identifier)

# insert metadata
r.collection_item_create(collection_id, data)

# update metadata
r.collection_item_update(collection_id, identifier, data)

# delete metadata
r.collection_item_delete(collection_id, identifier)
```

Figure 10 — Screenshot of the OWSLib transactions demo

5.9. Geopython stack

The geopython suite of tooling was integrated and demonstrated as part of a mentor stream to demonstrate end to end metadata lifecycle management.

- pygeometa: create / manage metadata from simple YAML configurations, exporting to numerous metadata formats
- OWSLib: publish metadata via OGC API — Features — Part 4: Create, Replace, Update and Delete to an OGC API — Records server
- pygeoapi: serve published metadata via OGC API — Records
- pycsw: serve published metadata via OGC API — Records

- QGIS: use QGIS' MetaSearch capability to discover resources via OGC API — Records

A screenshot of the mentor session is shown in the figure below.

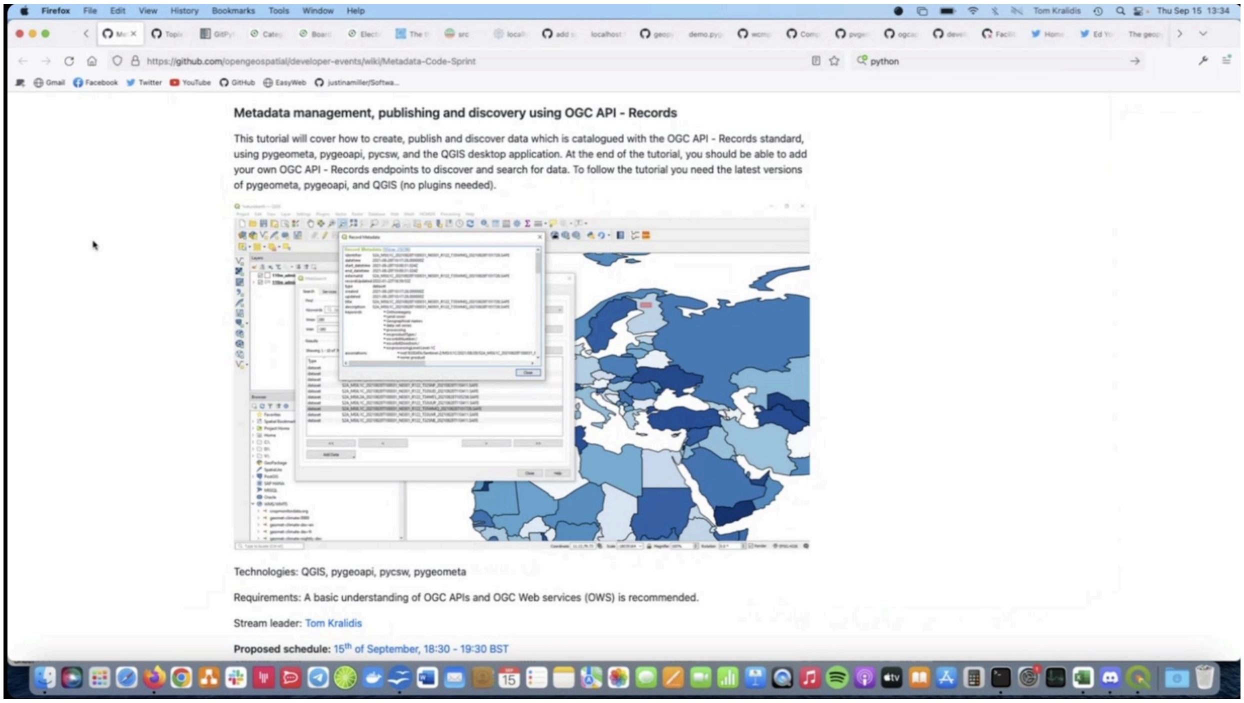

Figure 11 — Screenshot of metadata lifecycle management mentor stream using pygeometa, OWSLib, pygeoapi/pycsw, and QGIS

5.10. ISO 19115 activity by OpenWork

5.10.1. Background

OpenWork participated in the code sprint, supported by both OGC and the Australia and New Zealand Intergovernmental Committee on Survey and Mapping (ICSM). Byron Cochrane, a metadata expert from OpenWork, participated on behalf of OpenWork with the primary role in the code sprint of sharing technical expertise on ISO 19115 metadata. This expertise included participation in standard development, implementation, development of tools and guidance. A significant part of this expertise has been gained through involvement in the ICSM Metadata Working Group (MDWG) and chairing of the OGC Metadata and Catalogue Domain Working Group, as well as participation in the OGC API Records Standards Working Group.

One of the challenges of assisting and advising organisations in creating, managing and using spatial metadata is that due to the apparent proliferation of solutions, it has become more challenging to advise which metadata solutions are best suited to a particular organisation's needs. This is partly because the role of metadata and catalogues is multifaceted and varied across domains. Currently, unfortunately, some of the communities providing the tools or solutions do not offer such guidance. A framework by which such guidance could be created and shared was a primary goal of OpenWork's participation.

Other sprint goals included to:

- Examine the possibility of creating a JSON encoding of ISO 19115-1
- Create Crosswalks between ISO 19115-1, STAC and OGC API Records

5.10.2. Day 1

The focus of Day One of the code sprint was for participants to better understand the primary components: OGC API — Records, STAC, ISO 19115, and JSON-FG — their histories, capabilities, use and future development plans. OpenWork led an hour-long presentation and discussion on ISO 19115 and related standards.

5.10.2.1. Presentation of ISO 19115

OpenWork briefed the sprint participants, highlighting issues around the lack of understanding of the differences between the standard names and versions of ISO 19115. In Australia and New Zealand, ICSM strongly endorsed the ISO 19115-1 metadata standard and deprecated the previous ISO 19115 profiles. This support includes the latest version, ISO 19115-1:2014/Amd2:2020, which provides improved support for capturing information related to coordinate drift due to tectonic movement — a significant issue in Australia where coordinate values can change over one and a half metres in 20 years. However, it is clear that many professionals confuse the versions and do not understand why changes exist. For example, a common confusion is referencing ISO 19115:2003 as ISO 19115-1 (initially likely to contrast with ISO 19115-2:2009 Extensions for imagery and gridded data). Yet ISO 19115-1 was not released until 2014. As a result, today, many software vendors claim support for ISO 19115-1 when they only support the older ISO 19115:2003 standard. Byron Cochrane explained that, for Europe, this issue is aggravated by the fact that the INSPIRE framework of the European Union (EU) has yet to provide a mechanism to support ISO 19115-1 even though work on this standard was well underway when the EU launched INSPIRE. The INSPIRE community must address this situation as it prohibits many in Europe from using the improved standards and limits software developers' willingness to include support for enhanced standards worldwide.

Two significant discussions resulted from this presentation. The first involved the value of namespaces in encoded metadata. The namespace issue is closely related to the desire to create a canonical JSON encoding of ISO 19115-1. One position was that many JSON developers dislike using namespaces and tend to remove them if they are present. A contrasting view posited that namespaces are an elegant approach for reducing ambiguity in data and, when not used, are often eventually replaced by less friendly solutions. A question raised and unanswered was whether a canonical encoding of ISO 19115-1 would be possible.

This first discussion led to the second. The question was, "What would the best approach to a JSON encoding of ISO 19115-1 be?" Would it be best to create a fully compliant canonical version like ISO 19115-3 encoded in JSON? Or would it be better to take a more piecemeal approach and extend OGC API Records where needed until it could provide a fully compliant record? The challenges of directly aligning the GeoJSON format used by OGC API Records with ISO 19115-1 and the desire to improve crosswalks for at least some elements during the code sprint lead OpenWork to take the latter approach. This proved to be the right approach for this sprint's shorter-term purposes and outcomes.

5.10.2.2. Crosswalks

For the remainder of day one, taking the advice and support of participants from MariaDB CubeWerx and MSC, OpenWork undertook an effort to improve the ability to create crosswalks for keywords in ISO 19115-1 to OGC API — Records. OGC API — Records offers two possible locations for Keywords — Keywords and Themes/Concepts. Themes/Concepts were chosen as the natural fit. OGC API — Records follows the pattern of DCAT in this. The "Keywords" class is to hold free text tags. The Theme/Concepts can also contain free text but primarily support keyword concepts sourced from controlled vocabularies. (This is potentially a very **inelegant** solution that confuses developers, thus the simpler class should be considered for removal.)

By the end of the evening, OpenWork had altered the YAML schema to support the ingestion of ISO 19115-1 keyword elements with controlled vocabularies. OpenWork also proposed changing the names of the relevant classes for clarity suggesting "keywords" be "tags" and "themes" be "keywords". In addition, to provide a method to retrieve the original entire ISO 19115-1 record, OpenWork proposed a new element, "source", to link a complete ISO 19115-3 record, should it exist. We rejected this latter recommendation in favour of guidance on using the existing "link" class to provide this functionality.

5.10.3. Day 2

On Day 2, OpenWork focussed on creating a Record document conforming to the OGC API — Records candidate standard to test the guidance that OpenWork had developed the previous evening. Several changes ensued, mainly relating to the ability to provide resolvable URIs to both the concepts and their controlled vocabularies. More work and testing are needed.

Discussions clarified the roles that traditional catalogues, OGC API — Records and STAC serve. These can be summarised as follows:

- STAC is most appropriate for fine-level metadata — particularly cataloguing scenes in a collection of images to allow their discovery
- OGC API — Records focuses on the general discovery of spatial resources of any type. Because of this, it may not be useful or appropriate to capture in OGC API — Records any ISO 19115-1 metadata that exists for purposes other than discovery.
- ISO 19115-1 exists to document and manage a resource fully. Provenance, structural, informative, administrative and other metadata elements exist in this standard catalogue. This resource is most useful after discovery.

5.10.4. Day 3

The focus of OpenWork on Day 3 was to start providing a framework to guide the use of OGC API — Records. This followed the early discussion regarding how much structure should be encoded in the standard. Creating a rigid standard was not seen as desirable. However, a guidance document that could provide authoritative direction would be acceptable. The immediate aim was to support crosswalks from ISO 19115 through this approach.

On Day 3, OpenWork focused on the keyword (themes) class for which Byron Cochrane had created crosswalks earlier in the sprint. Inclusion of additional classes and crosswalked elements could be added over time. A pattern language approach, applied in OpenWork's work with ICSM, was used as a template. Such documentation, while useful, requires a great deal of effort to produce. But it allows the schema to remain simple. However, automated validation is difficult to provide through this approach.

6

DISCUSSION

6 DISCUSSION

6.1. Harmonization between STAC and OGC API Records

Although both STAC and OGC API — Records use GeoJSON for encoding metadata, alignment is necessary in order to improve metadata exchange. STAC has an Asset construct and OGC API — Records uses a Link construct. If OGC API — Records adopts the use of Assets then there is a need to be clear about the difference between Assets and Links.

There is a perception that STAC is focused on Earth Observation (EO). The Record metadata model which is specified in OGC API — Records is supposed to cover more than just EO. The idea with the Records metadata model is that it offers a small set of generic properties that can be used to describe anything. This makes it possible for communities of interest to extend the model to support their particular use case. The way that one uses a Record depends on how the person wants to make the resource discoverable. This impacts how the record is created and how it is linked with other Records.

There is a conflict between some elements where they may have the same name but different meaning (e.g. created and updated dates). The Records metadata model targets a the resource, whereas STAC targets a distribution. This creates a challenge for Records because it would be impossible to give different dates of update for different distributions of the same resource. To align OGC API Records with STAC there would be a need to change 'record-created' and 'record-updated' fields to simply 'created' and 'updated'. Other potential opportunities for alignment include the use a 'roles' element in the links section.

It was also determined that the STAC 'root' link relation types need to be further clarified as OGC link relation types or Compact URIs (CURIEs).

Every Item has a link to one collection. You could create a hierarchy using a collection. One level of collection and then the items are records. In this context, STAC and OGC API — Records are already aligned.

The lessons identified regarding harmonization can be summarized as:

- Modify the create and updated fields by removing them from the Record class add them to the links. This would improve alignment with STAC.
- In some cases it may be necessary to use either STAC or OGC API — Records, depending on the needs of the community of interest.
- Clarify definition and intent of STAC 'root' link relation types in relation to OGC link relation types or Compact URIs (CURIEs)

6.2. Harvesting

It is important to be clear about what harvesting means. It is the complementary operation to Transactions. In Transactions you push a record to the catalogue, that requires the client to take the source and push that to the catalogue. For harvesting, you tell the catalogue “here is the resource” and the catalogue extracts the metadata and takes care of the rest. It is always possible that the catalogue will go to the resource and not know what to do with it. In the CSW standard there was support for harvesting. With harvesting the onus is on the catalogue to determine how to transform the resource and create the record for that resource.

Is is also a question, if you point it to something, should we leave that to the harvestor? The simplest approach is to leave it up to the harvestor once the client has directed the harvestor towards a resource. In the CSW ebrim model there was guidance of how to harvest a GetCapabilities response. As a minimum they would look for all of the fields that are in the record and try to populare those. More guidance would be needed for OGC API Standards, for example how deep to go in navigating the link graph.

In certain instances, for example in the local Catalogue offered by OGC API Records, we cover that with local instance resources. This supports a situation where an entity might have a catalogue but the catalogue might have records about a separate entity.

An example workflow would be to: Submit a harvest request, with a prefer header, then the request would be executed synchronously or asynchronously, then it would notify you that it is initiated a job meaning that the client application can monitor the job and retrieve the results when the job is complete. It could then leverage an approach similar to that demonstrated by Secure Dimensions. To achieve such a workflow, there is a need to define an API for harvesting resources or collections of resources. Providing guidance on how to crawl an OGC API resource tree and to harvest the resources it offers.

The lessons identified regarding harmonization can be summarized as:

- It is important to be clear about what harvesting means.
- Keeping the metadata close to the data is more efficient than copying the metadata to a separate server.
- Ideally harvesting would be of selected bits of metadata instead of the complete metadata record.
- There are different types of harvesting, in some cases there may be some processing needed. One type of approach means harvesting the discovery metadata.
- In some cases, augmented metadata may need to be pushed back to the source.
- We need to be clear about what we mean when we say “close to the data”.

6.3. ISO 19115 metadata and OGC API Records

The lessons identified regarding ISO 19115 metadata and OGC API Records can be summarized as:

- Expressing ISO 19115 metadata in OGC API Records should focus on discovery elements.
- Initial prototyping has been focused on Keywords to Themes
- What is needed is a profile that enables us to work with ISO 19115
- Content negotiation by profile could be useful.
- The incremental approach would be useful.
- It may be necessary to also design a JSON profile of ISO 19139 as well.
- There are various considerations relating to alignment with ISO 19115 e.g. alignment with DCAT
- We need to balance how deeply we want to represent ISO metadata in JSON

The following section presents the conclusions.

6.4. JSON-FG

TBA

6.5. Transactions

The lessons identified regarding transactions can be summarized as:

- Currently, OGC API — Features — Part 4: Create, Replace, Update and Delete defines two requirements classes (“Create/Replace/Delete”, and “Update”). It would be valuable to split the “Create/Replace/Delete” requirements class into 3 separate classes, to allow for finer granularity for resource management.

6.6. Addition of JSON-FG as another encoding for OGC API Records

The lessons identified regarding addition of JSON-FG as another encoding for OGC API Records can be summarized as:

- There is not an identified need for JSON-FG encodings in OGC API Records and STAC. However, it could be identified in the future.
- The canonical time is a top-level element in JSON-FG. This could be useful for OGC API Records.

More feedback from implementers is needed in order to improve understanding of the potential use of JSON-FG in OGC API — Records.

7

CONCLUSIONS

7 CONCLUSIONS

This was the first hybrid code sprint (consisting of both in-person and remote elements) organized by the OGC in more than two years, due to the pandemic. A record number of participants registered to attend the code sprint, exceeding pre-pandemic registration numbers. There were however, more remote participants than those attending in-person. This suggests that there continues to be significant interest in code sprints, and that the online collaboration environment should continue to be used post-pandemic.

The code sprint facilitated the development and testing of prototype implementations of OGC and ISO Standards that relate to geospatial metadata and catalogues. The code sprint also enabled the participating developers to provide feedback to the editors of candidate standards. The code sprint therefore met all of its objectives and achieved its goal of accelerating the support of open geospatial standards that relate to geospatial metadata and catalogues.

7.1. Future Work

The sprint participants made the following recommendations for future innovation work items:

- Initiatives to facilitate implementation of JSON-FG (e.g. 3D, cadastral data, etc)
- Initiatives to facilitate implementation of catalogues
- Prototyping of tools for creating metadata (e.g. the automated STAC metadata crawler demonstrated during the sprint)

The sprint participants also made the following recommendations for things that the SWGs should consider:

- Outreach for JSON-FG
- Code Sprint for designing profiles of JSON-FG for different communities of interest
- Documentation of the different roles of catalogues and API, as well as guidance on when to use them
- Code Sprint on versioning, possibly combining an OGC API Features Part 4 with OGC API Records
- Exploring how to move GeoDCAT forward within OGC

It is envisaged that the SWGs will consider the above-listed recommendations for future work items.

A

ANNEX A (INFORMATIVE) REVISION HISTORY

A

ANNEX A (INFORMATIVE) REVISION HISTORY

DATE	RELEAS	AUTHOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2022-03-11	0.1	G. Hobona	all	initial version
2022-04-29	0.2	G. Hobona	all	Updated with feedback from participants

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Holmes, C.: Static SpatioTemporal Asset Catalogs in Depth, Available at https://medium.com/radiant-earth-insights/static-spatiotemporal-asset-catalogs-in-depth-710530934a84

2. OGC: OGC 21-008: Joint OGC OSGeo ASF Code Sprint 2021 Summary Engineering Report, 2021

3. Balaban, A.: OGC 21-019, OGC Testbed-17: Attracting Developers: Lowering the entry barrier for implementing OGC Web APIs. Open Geospatial Consortium (2022). https://docs.ogc.org/per/21-019.html