OGC® DOCUMENT: 22-004

External identifier of this OGC® document: http://www.opengis.net/doc/PER/ogc-osgeo-asf-codesprint2022

Open Geospatial Consortium

JOINT OGC OSGEO ASF CODE SPRINT 2022 SUMMARY ENGINEERING REPORT

ENGINEERING REPORT

PUBLISHED

Submission Date: 2022-04-29
Approval Date: 2022-10-10
Publication Date: 2022-11-10
Editor: Gobe Hobona, Joana Simoes, Angelos Tzotsos, Tom Kralidis, Martin Desruisseaux

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

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 decade, geospatial technologies and data have become more widespread in use and application. A key catalyst for this increased uptake of geospatial technologies is the interoperability achieved through implementation of open geospatial standards. Another important catalyst for this increased uptake is the availability of open source software products that are able to extract, transform, analyze, and disseminate geospatial data.

In February 2021, the Open Geospatial Consortium (OGC), the Apache Software Foundation (ASF), and the Open Source Geospatial Foundation (OSGeo) held their first joint Open Source Software and Open Standards Code Sprint (OGC 21-008). 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 March 8th and 10th, 2022. The second code sprint, named the 2022 Joint OGC OSGeo ASF Code Sprint, served to accelerate the support of open geospatial standards within the developer community.

Part of the motivation for holding the code sprint in 2022 was the growing uptake of location information across the global developer communities. The code sprint brought together developers of Open Standards, Open Source Software and Proprietary Software. The code sprint therefore provided a rare opportunity for developers across these communities to focus on common challenges, within a short space of time, and in a shared collaborative environment.

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

The Open Source Geospatial Foundation (OSGeo) is a not-for-profit organization whose mission is to foster global adoption of open geospatial technology by being an inclusive software foundation devoted to an open philosophy and participatory community driven development. The foundation consists of projects that develop open source software products. The sprint objectives for OSGeo projects were:

- Release new software versions
- Fix open issues

- Develop new features
- Improve documentation, translations
- Develop prototype implementations of OGC standards

The Apache Software Foundation (ASF) is an all-volunteer community comprising 815 individual Members and 8,500 Committers on six continents stewarding more than 227 million lines of code, and overseeing more than 350 Apache projects and their communities. The sprint objectives for ASF projects were:

- Improve support of OGC standards
- Improve visualization capabilities
- Improve documentation
- Improve interoperability with other libraries

The 2022 Joint Code Sprint introduced several changes that were not there during the 2021 Joint Code Sprint. First, a new collaboration platform – Discord- was used. Discord allowed both chat and video communications to be offered from within the same environment. Discord also supported the creation of multiple chat channels, thereby making it possible for separate projects to have their own dedicated chat channels. Second, the code sprint offered Mentor Streams that presented tutorials for developers that were getting started in using various standards or software products. Finally, a dedicated chat channel was created for the event sponsor, Ordnance Survey, on the code sprint's Discord environment thereby making it possible for sprint participants to visit the channel and ask about the sponsor's products.

The code sprint facilitated the development and testing of prototype implementations of OGC standards, including implementations of draft OGC API standards. Further, the code sprint also enabled the participating developers to provide feedback to the editors of OGC standards. Furthermore, the code sprint provided a collaborative environment for OSGeo and ASF developers to fix open issues in products, develop new features, improve documentation, improve interoperability with other libraries/products, and develop prototype implementations of OGC standards. The code sprint therefore met all of its objectives and achieved its goal of accelerating the support of open geospatial standards within the developer community.

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

- Prototypes of catalogues that can be crawled through by an application. Currently there are several searchable catalogues but no catalogues that can be crawled through by an application.
- More specification validation work for OGC API Records.
- More experiments for the Workflows extension of OGC API Processes. This could try out a variety of workflow approaches.
- Experimentation on how a processing server can interact properly with other OGC API implementations that serve data. For example, in this code sprint there was an

implementation of OGC API Processes (ZOO Project) that interacted with an OGC API Features implementation (MapServer).

- Experimentation with OGC's geoparquet candidate standard and Apache Arrow.

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

- To improve examples and documentation related to OGC API Records.
- To advance the development of the Executable Test Suites of OGC API Processes.
- To advance the development of the Executable Test Suites of OGC API Tiles.
- To advance the development of the Executable Test Suites of OGC API Coverages.

II KEYWORDS

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

OGC, OSGeo, hackathon, code sprint, standards, geospatial, API, open source

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
Angelos Tzotsos	Open Source Geospatial Foundation	Editor
Martin Desruisseaux	Geomatys	Editor
Tom Kralidis	Meteorological Service of Canada	Editor
Gérald Fenoy	GeoLabs	Contributor
Rajat Shinde	IIT Bombay	Contributor
Michael Arneson	INT	Contributor
Blasco Brauzzi	Terradue Srl	Contributor
Vicky Vergara	pgRouting (OSGeo)	Contributor
Josh Townsend	bp	Contributor
Jerome Jacovella-St-Louis	Ecere Corporation	Contributor
Massimiliano Cannata	SUPSI	Contributor
Morten Breiner	EIVA	Contributor

NAME	ORGANIZATION	ROLE
Ana Paula Seraphim	University of Cambridge	Contributor
Paloma Abad	National Center of Geographic Information (Spain)	Contributor
Carmen Tawalika	mundialis	Contributor
Clemens Portele	interactive instruments GmbH	Contributor
Haifeng Niu	Department of Land Economy, University of Cambridge	Contributor
Samantha Lavender	Pixalytics Ltd	Contributor
Ashish Kumar	IIT (BHU) Varanasi	Contributor
Carlos Eduardo Mota	CPRM	Contributor
Bruno Kinoshita	Apache Software Foundation	Contributor
Paul van Genuchten	ISRIC World Soil Information	Contributor
Panagiotis Vretanos	CubeWerx Inc.	Contributor
Anika Weinmann	mundialis	Contributor
Eugene Yu	George Mason University	Contributor
Iván Sánchez Ortega	OSGeo charter member	Contributor
Ayodele Michael A	(self)	Contributor
Weston Renoud	QPS BV	Contributor
Luca Delucchi	Fondazione Edmund Mach	Contributor
Francesco Bartoli	Geobeyond	Contributor
Patrick Dion	Ecere Corporation	Contributor
Antonio Cerciello	Byte Road	Contributor
Brian M. Hamlin	OSGeo California Chapter	Contributor

NAME	ORGANIZATION	ROLE
Jack Riley	NOAA	Contributor
James Case	Case Ocean Services LLC	Contributor
Kevin Lalli	Hydrosat	Contributor
Marta Conceição	FMUL	Contributor
Matthias Loeks	BASF Digital Farming GmbH	Contributor
Maxime Collombin	University of Applied Sciences, Western Switzerland, School of Business & Engineering Vaud (HEIG-VD)	Contributor
Mehmet Akif Ortak	IT	Contributor
Tracey Birch	(self)	Contributor

V

ABSTRACT

The subject of this Engineering Report (ER) is a code sprint that was held from the 8th to the 10th of March 2022 to advance support of open geospatial standards within the developer community, whilst also advancing the standards themselves. The code sprint was hosted by the Open Geospatial Consortium (OGC), the Apache Software Foundation (ASF), and Open Source Geospatial Foundation (OSGeo). The code sprint was sponsored by Ordnance Survey (OS), and held as a completely virtual event.

1 SCOPE

1 SCOPE

This Engineering Report (ER) summarizes the main achievements of the Joint OGC OSGeo ASF Code Sprint, conducted between March 8th and 10th, 2022. Sponsored by Ordnance Survey (OS), the code sprint was hosted by the OGC, ASF, and OSGeo with the goal of accelerating the support of open geospatial standards within the developer community.

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

W3C: HTML5, W3C Recommendation, https://www.w3.org/TR/html5/

Schema.org: https://schema.org/docs/schemas.html

Clemens Portele, Panagiotis (Peter) A. Vretanos, Charles Heazel: OGC 17-069r3, *OGC API — Features — Part 1: Core*. Open Geospatial Consortium (2019). https://docs.ogc.org/is/17-069r3/17-069r3.html.

Heazel, C.: OGC API — Common — Part 1: Core (Draft). OGC 19-072, Open Geospatial Consortium, http://docs.ogc.org/DRAFTS/19-072.html

Heazel, C.: OGC API — Common — Part 2: Geospatial Data (Draft). OGC 20-024, Open Geospatial Consortium, http://docs.ogc.org/DRAFTS/20-024.html

OGC: OGC 07-011, *Topic 6 — Schema for coverage geometry and functions*. Open Geospatial Consortium (2007). https://portal.ogc.org/files/?artifact id=19820.

John R. Herring: OGC 17-087r13, *Topic 1 — Features and geometry – Part 1: Feature models* . Open Geospatial Consortium (2020). https://docs.ogc.org/as/17-087r13/17-087r13.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. **Web API**

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

3.5. Abbreviated terms

API	Application Programming Interface
ASF	Apache Software Foundation
CIS	Coverage Implementation Schema
CRS	Coordinate Reference System
DGGS	Discrete Global Grid Systems
EDR	Environmental Data Retrieval
GIS	Geographic Information System
GRASS	Geographic Resources Analysis Support System
MOU	Memorandum of Understanding
OGC	Open Geospatial Consortium
OSGeo	Open Source Geospatial Foundation
OWS	OGC Web Services
REM	Route Exchange Model
REST	Representational State Transfer
SDI	Spatial Data Infrastructure
TMS	Tile Matrix Set
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WMTS	Web Map Tile Service
WPS	Web Processing Service

4

HIGH-LEVEL ARCHITECTURE

4 HIGH-LEVEL ARCHITECTURE

The focus of the sprint was on the support of implementations of open geospatial standards across various open source software projects. Implementations of approved and candidate OGC Standards were deployed in participants' own infrastructure in order to build a solution with the architecture shown below in Figure 1.

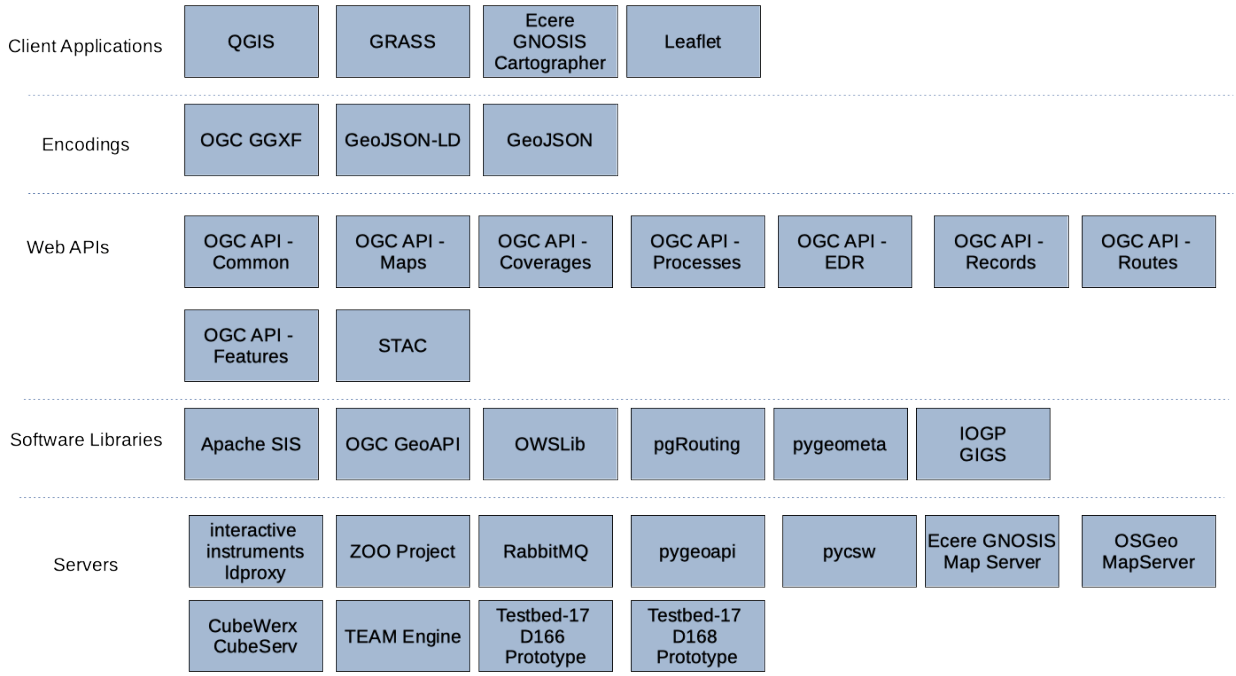

Figure 1 — High Level Overview of the Sprint Architecture

As illustrated the sprint architecture was designed with the view of enabling client applications to connect to different servers that implement open geospatial standards such as the suite of OGC API standards. The architecture also included several different software libraries that support open geospatial standards and enable the extraction, transformation and loading of geospatial data. The rest of this section describes the software deployed and standards implemented during the code sprint.

4.1. Approved OGC Standards

This section describes the approved OGC standards implemented during the code sprint.

4.1.1. OGC API — Features

The OGC API — Features standard offers the capability to create, modify, and query spatial data on the Web and specifies requirements and recommendations for APIs that want to follow a standard way of sharing feature data. A 'feature' is an abstraction of real-world phenomena or entity (OGC 17-087r13). The standard specifies discovery and query operations, with additional operations planned for future extensions of the standard. Discovery operations enable client applications to interrogate the API to determine its capabilities and retrieve information about the dataset that is accessible through the API. In contrast, Query operations enable client applications to retrieve feature data from the underlying storage based upon selection criteria defined by the client. The standard can be considered the successor to the widely implemented Web Feature Service (WFS) standard.

The OGC API — Features standard comprises of multiple parts, which are listed below:

- Part 1: Core is an approved standard that describes the mandatory capabilities that every implementing service has to support and is restricted to read-access to spatial data that is referenced to the World Geodetic System 1984 (WGS 84) Coordinate Reference System (CRS).
- Part 2: CRS by Reference is an approved standard that enables the use of different CRS, in addition to the WGS 84.
- Part 3: Filtering is a candidate standard that defines the behavior of a server that supports enhanced filtering capabilities expressed using the Common Query Language (CQL2).

Additional capabilities that address specific needs will be specified in additional parts. Envisaged future capabilities include, for example, support for creating and modifying data, more complex data models, and richer queries.

4.1.2. OGC API — Environmental Data Retrieval

An Environmental Data Retrieval (EDR) API provides a family of lightweight interfaces to access environmental data resources. Each resource addressed by an EDR API maps to a defined query pattern. The OGC API – Environmental Data Retrieval standard identifies resources, captures compliance classes, and specifies requirements that are applicable to environmental data retrieval. The standard addresses two fundamental operations; discovery and query of environmental data resources. Discovery operations allow the API to be interrogated to determine its capabilities and retrieve information (metadata) about this distribution of a resource. This includes the API definition of the server as well as metadata about the environmental data resources provided by the server. Query operations allow environmental data resources to be retrieved from the underlying data store based upon simple selection criteria, defined by this standard and selected by the client.

4.1.3. OGC API — Processes

The OGC API — Processes standard enables the execution of computing processes and the retrieval of metadata describing their purpose and functionality. Typically, these processes combine raster, vector, and/or coverage data with well-defined algorithms to produce new raster, vector, and/or coverage information. The standard can be considered the successor to the widely implemented Web Processing Service (WPS) standard.

OGC API – Processes is comprised of multiple parts, which are listed below:

- Part 1: Core is an approved standard that specifies an interface that enables the execution of computing processes and the retrieval of metadata describing their purpose and functionality.
- Part 2: Deploy, Replace, Undeploy is a candidate standard that specifies an interface for deployment, replacement, and undeployment of processes from a server.
- Part 3: Workflows is a candidate standard specifies an interface that provides the ability to chain nested processes, refer to external processes and collections accessible via OGC API standards, and trigger execution of processes through implementations of OGC API standards.

4.1.4. OGC GeoAPI

The GeoAPI Implementation Standard defines the normalized use of the GeoAPI library. The GeoAPI library contains a series of interfaces and classes in the Java programming language defined in several packages which interpret into Java the data model and Unified Modeling Language (UML) types that are specified in ISO and OGC standards documents. The library includes extensive Javadoc code documentation which complements the implementation of the ISO/OGC specifications by explaining particularities of the GeoAPI library: interpretations made of the specifications where there was room for choice, constraints due to the library's use of Java, or standard patterns of behavior expected by the library, notably in its handling of return types during exceptional situations.

In this code sprint, the GeoAPI implementers focused on support for the Geospatial Integrity of Geoscience Software (GIGS) specification.

4.2. Draft OGC Specifications

This section describes the draft OGC specifications implemented during the code sprint.

4.2.1. OGC API — Common

The draft OGC API — Common standard specifies the set of common practices and shared requirements that have emerged from the development of Resource Oriented Architectures and Web APIs within the OGC. The specification serves as a common foundation upon which all OGC APIs will be built. Consistent with the architecture of the Web, this specification uses a resource architecture that conforms to principles of Representational State Transfer (REST). The draft OGC API – Common specification establishes a common pattern that leverages the OpenAPI specification for describing APIs.

4.2.2. OGC API — Coverages

The draft OGC API — Coverages standard defines a Web API for accessing coverages that are modeled according to the Coverage Implementation Schema (CIS) 1.1. A coverage is a feature that acts as a function to return values from its range for any direct position within its spatial, temporal or spatiotemporal domain (OGC 07-011). Coverages are represented by some binary or ASCII serialization, specified by some data (en-coding) format. Arguably the most popular type of coverage is that of a gridded coverage. Gridded coverages have a grid as their domain set describing the direct positions in multi-dimensional coordinate space, depending on the type of grid. Satellite imagery is typically modeled as a gridded coverage, for example. OGC API — Coverages can be considered the future successor to the widely implemented Web Coverage Service (WCS) standard.

4.2.3. OGC API — Maps

The draft OGC API — Maps standard describes an API that presents maps portraying data that has been rendered according to a style. The maps served by implementations of the draft OGC API — Maps standard are retrieved as images of any size, generated on-the-fly, and with the styling determined by the client application. OGC API — Maps can be considered the future successor to the widely implemented Web Map Service (WMS) standard.

4.2.4. OGC API — Records

The draft OGC API — Records standard provides discovery and access to metadata records about resources such as features, coverages, tiles / maps, models, assets, services or widgets. The draft specification enables the discovery of geospatial resources by standardizing the way collections of descriptive information about the resources (metadata) are exposed. The draft specification 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.

4.2.5. OGC API — Routes

The draft OGC API — Routes standard specifies the behavior of a Web API that allows applications to request routes in a manner independent of the underlying routing data set, routing engine or algorithm. The draft standard defines modular API building blocks for computing new routes, specifying commonly used routing parameters, and for managing routes. Implementations of OGC API — Routes represent routes as a feature collection encoded in GeoJSON according to the OGC Route Exchange Model (REM) candidate standard.

4.2.6. OGC API — Tiles

The draft OGC API - Tiles standard specifies the behavior of Web APIs that provide access to tiles of one or more geospatial data resources (collections) or from the whole dataset that the Web API offers. This draft standard defines how to discover which resources offered by the Web API can be retrieved as tiles, get metadata about the available tile sets (including according to which tile matrix set each tile set is partitioned in and the limits of that tile set within a common potentially global tile matrix set) and how to request a tile.

4.2.7. OGC Gridded Geodetic Data Exchange Format (GGXF)

The purpose of the Gridded Geodetic Data Exchange Format (GGXF) project team is to design a file structure and computer storage mechanism for the efficient exchange of regularly gridded geodetic data.

4.3. OSGeo Projects

This section describes software products, from OSGeo Projects, that were deployed during the code sprint.

4.3.1. OSGeo GRASS GIS

The Geographic Resources Analysis Support System (GRASS) is an open source GIS providing raster, vector and geospatial processing capabilities. It can be used either as a stand-alone application or as backend for other software packages such as QGIS and R or in the cloud [1].

4.3.2. OSGeo MapServer

MapServer is an open source platform for publishing spatial data and interactive mapping applications to the web.

In this code sprint, the participants implemented an extension to MapServer that enabled MapServer to offer an interface conforming to OGC API — Features.

4.3.3. OSGeo PostGIS

PostGIS provides spatial objects for the PostgreSQL database, allowing storage and query of information about location and mapping.

4.3.4. OSGeo Proj

PROJ is a generic coordinate transformation software library that transforms geospatial coordinates from one coordinate reference system (CRS) to another.

4.3.5. OSGeo pycsw

pycsw is a server-side python implementation of the OGC Catalogue Services for the Web (CSW) standard.

4.3.6. OSGeo QGIS

QGIS is a free and open-source cross-platform desktop GIS that supports viewing, editing, and analysis of geospatial data.

4.4. OSGeo Community Projects

This section describes software products, from OSGeo Community Projects, that were deployed during the code sprint.

4.4.1. 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.

4.4.2. OSGeo OWSLib

OWSLib is a Python package for client programming with OGC Web Service (OWS) standards, and their related content models. OWSLib also supports some OGC API Standards.

4.4.3. OSGeo pgRouting

pgRouting extends the PostGIS / PostgreSQL geospatial database to provide geospatial routing functionality.

4.4.4. OSGeo pygeoapi

pygeoapi is a Python server implementation of the OGC API suite of standards.

4.4.5. OSGeo pygeometa

pygeometa is a Python package to generate metadata for geospatial datasets.

4.4.6. TEAM Engine

The Test, Evaluation, And Measurement (TEAM) Engine is a testing facility that executes test suites developed using the TestNG framework or the OGC Compliance Test Language (CTL). It is typically used to verify specification compliance and is the official test harness of the OGC Compliance Testing Program (CITE).

4.4.7. OSGeo ZOO-Project

ZOO-Project is a Web Processing Service (WPS) implementation written in C. It is an open source platform which implements the WPS 1.0.0 and WPS 2.0.0 OGC Standards.

4.5. ASF Projects

This section describes software products, from ASF Projects, that were deployed during the code sprint.

4.5.1. Apache SIS

Apache Spatial Information System (SIS) is a free and open source software library for developing geospatial applications. The library is an implementation of OGC GeoAPI 3.0.1 interfaces and can be used for desktop or server applications. Services provided by Apache SIS include metadata, coordinate transformations, filtering and grid coverages. The library is implemented using the Java programming language.

In this code sprint, Apache SIS was used for testing the Geospatial Integrity of Geoscience Software (GIGS) tests runner.

4.6. Other open source products

4.6.1. 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).

4.6.2. RabbitMQ

RabbitMQ is an open source message broker that supports multiple messaging protocols. RabbitMQ is lightweight and deployable on premises and cloud computing environments. RabbitMQ can support distributed and federated enterprises to meet scalability and availability requirements.

In this code sprint, RabbitMQ was reviewed by ZOO Project developers as an option for supporting pubsub (publish/subscribe) needs.

4.6.3. Testbed-17 D166 Prototype

The aim of the Testbed-17 D166 prototype was to support experimentation on lowering the entry barrier for implementing OGC Web APIs [2]. The prototype uses JavaScript to implement offer interfaces conforming to OGC API Features and OGC API EDR standards. The base system is provisioned by a PostgreSQL database and supported by an instance of the GDAL library.

4.6.4. Testbed-17 D168 Prototype

The aim of the Testbed-17 D168 prototype was to help with optimizing documentation and scripts to support data providers implementing OGC API Standards in Cloud environments [2]. The prototype is designed to be a conda-deployable python environment.

4.7. Proprietary products

4.7.1. 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.

4.7.2. GNOSIS Map Server

The GNOSIS Map Server is written in the eC programming language and supports multiple OGC API specifications. GNOSIS Map Server supports multiple encodings including GNOSIS Map Tiles (which can contain either vector data, gridded coverages, imagery, point clouds or 3D meshes), Mapbox Vector Tiles, GeoJSON, GeoECON, GML and MapML. An experimental

server is available online at https://maps.ecere.com/ogcapi and has been used in multiple OGC Innovation Program initiatives.

5

RESULTS

5 RESULTS

The code sprint included multiple software libraries, OWS implementations, OGC API implementations and different client applications. In addition to supporting OWS and OGC API standards, various ASF and OSGeo software products involved in the code sprint also supported a variety of OGC encoding standards. This section presents some of the results from the code sprint.

5.1. Leaflet

One of the contributors of Leaflet implemented support for OGC API — Maps in the code sprint. Support for OGC API — Maps in Leaflet was implemented as a plug-in, enabling anyone to integrate the JavaScript files into their JavaScript application.

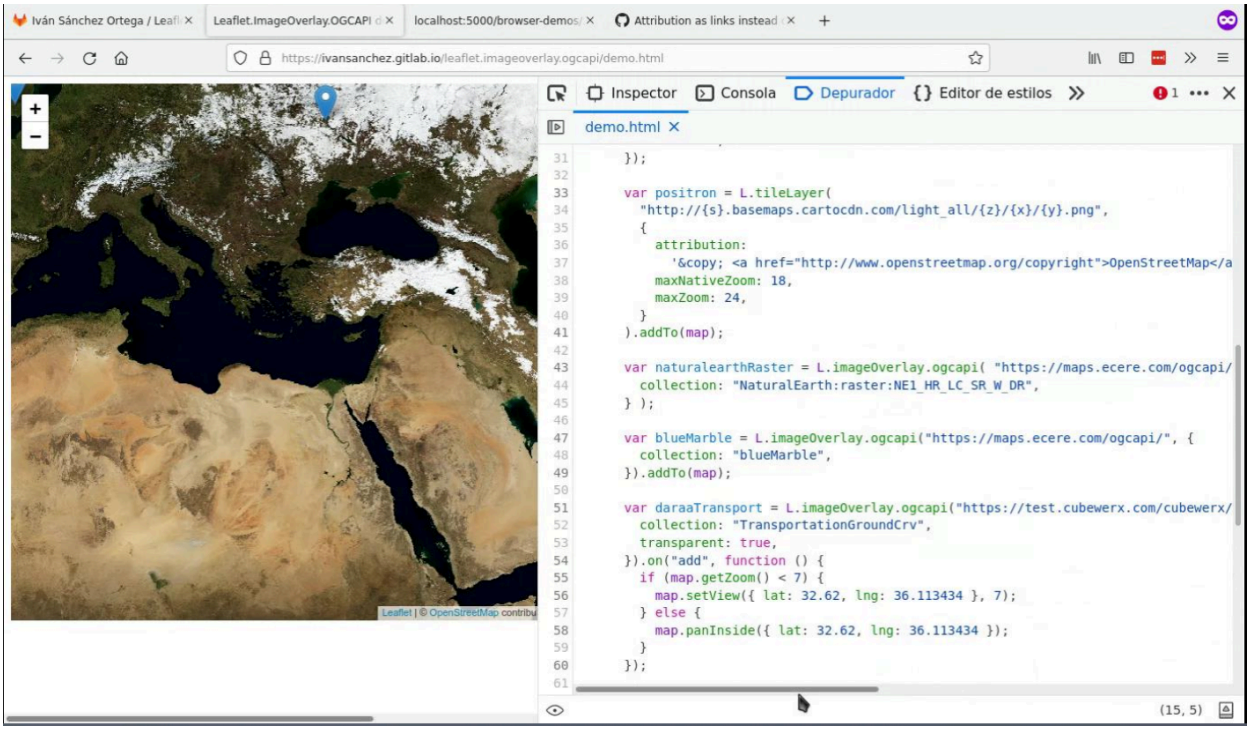

Figure 2 — Screenshot of the leaflet demo

5.2. QGIS MetaSearch Plugin

Developers from OSGeo and the Meteorological Service of Canada (MSC) worked on an enhancement to the QGIS MetaSearch plugin to improve support for OGC API — Records.

MetaSearch is a core plugin that enables QGIS to act as a client application for interacting with metadata catalogue services.

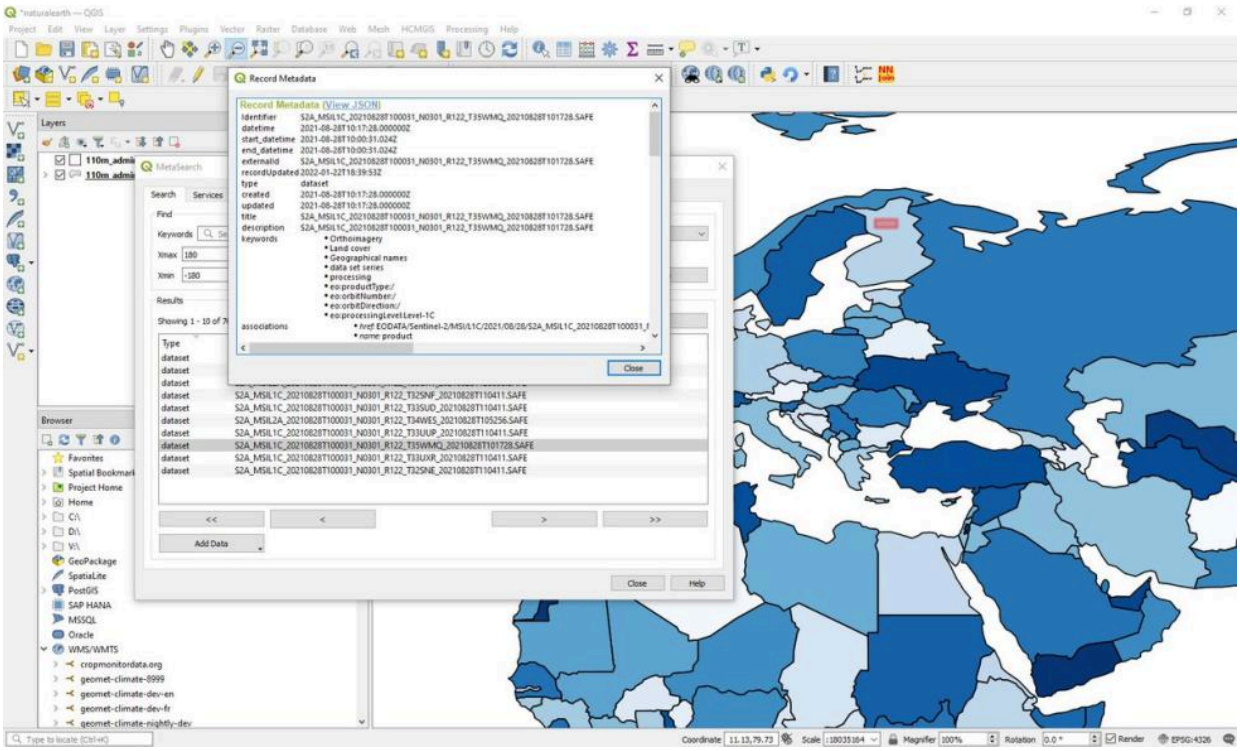

Figure 3 — Screenshot of the demo of the MetaSearch plugin for QGIS

5.3. Extension of the Testbed-17 Dataset testing (D166) prototype

For the code sprint, the maintainers of the Testbed-17 Dataset testing (D166) prototype experimented with use of ArcGIS and arcpy to offer an implementation of OGC API EDR. During the experimentation, the sprint participants identified an alternative way of defining the bbox query parameter in the YAML-encoded schema files provided with the OGC API EDR Standard. This would enable an OpenAPI documentation tool such as SwaggerUI to display a field for the query parameter.

Currently the bbox query parameter is defined as shown below:

```
schema:
  oneOf:
  - maxItems: 4
    minItems: 4
    type: object
  - maxItems: 6
    minItems: 6
    type: object
```

```
      type: array
```

Figure 4

This results in the SwaggerUI presentation looking as shown in Figure 5 below.

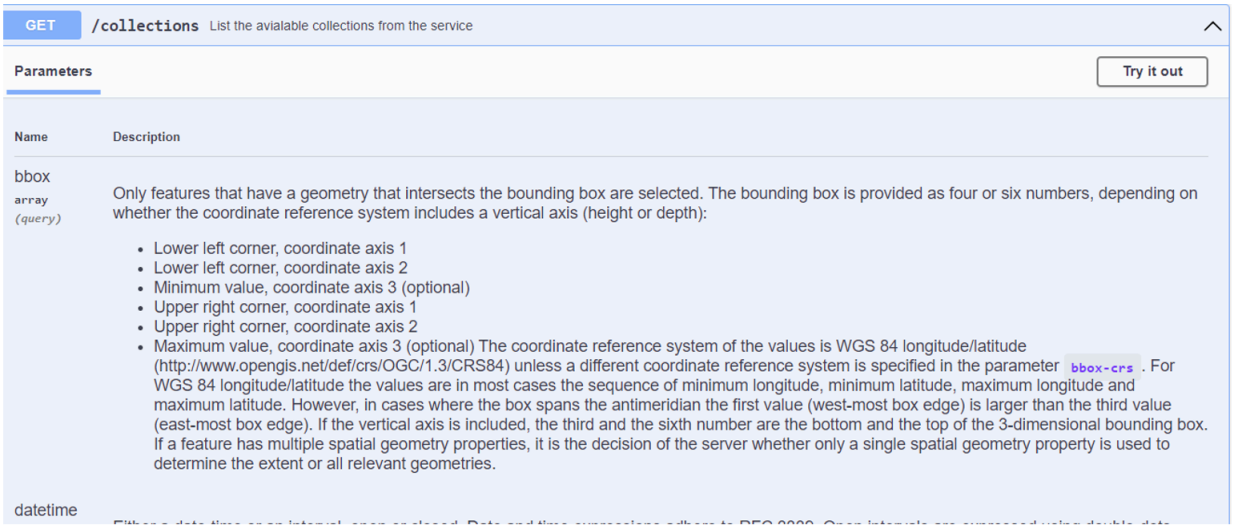

Figure 5 — Current display of the bbox query parameter on SwaggerUI

The sprint participants suggested changing the definition of the bbox query parameter to the one shown below:

```
      schema:
        oneOf:
        - items:
            maxItems: 4
            minItems: 4
            type: object
          type: array
        - items:
            maxItems: 6
            minItems: 6
            type: object
          type: array
```

Figure 6

The change results in the SwaggerUI presentation looking as shown in Figure 7 below.

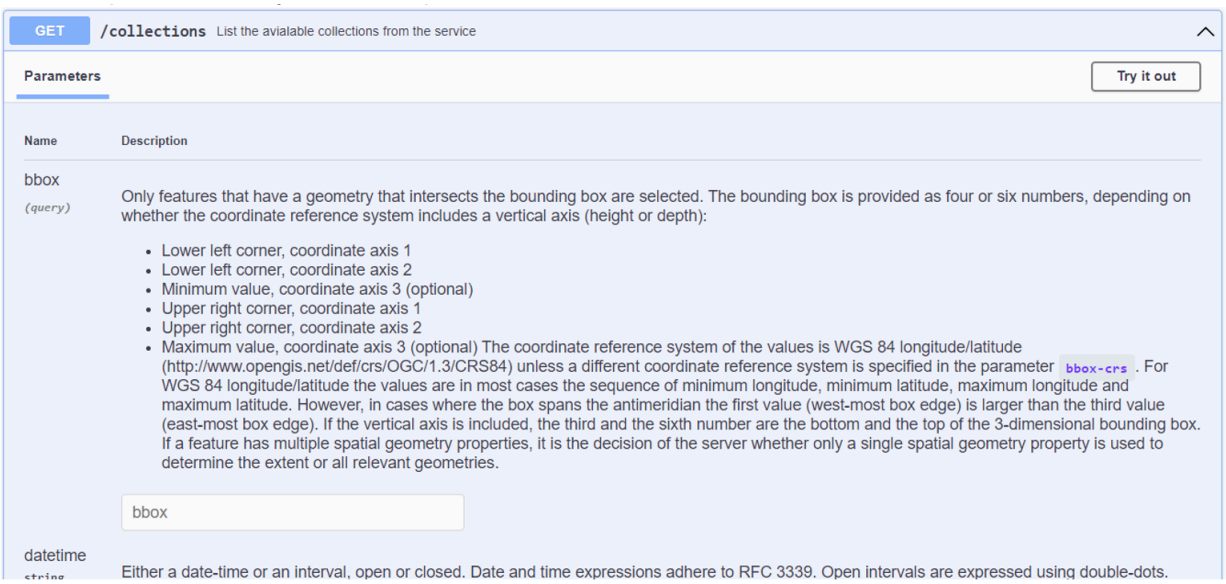

Figure 7 — Display of the bbox query parameter on SwaggerUI after the modification

5.4. Extension of the Testbed-17 Dataset testing (D168) prototype

For this code sprint, the prototype's support for OGC API — Records was reviewed.

The OSGeo pygeometa package is used to create the metadata for both the crawlable catalogue Record and individual object Records. The current implementation uses a fork of pygeometa with a new schema for the catalogue versus existing schema for the objects. It was tested whether the existing pygeometa schema could be adjusted to also satisfy the catalogue Record creation.

The maintainers of the prototype deployed catalogues with multiple objects, stored in an AWS S3 bucket, to test various options and support the discussion.

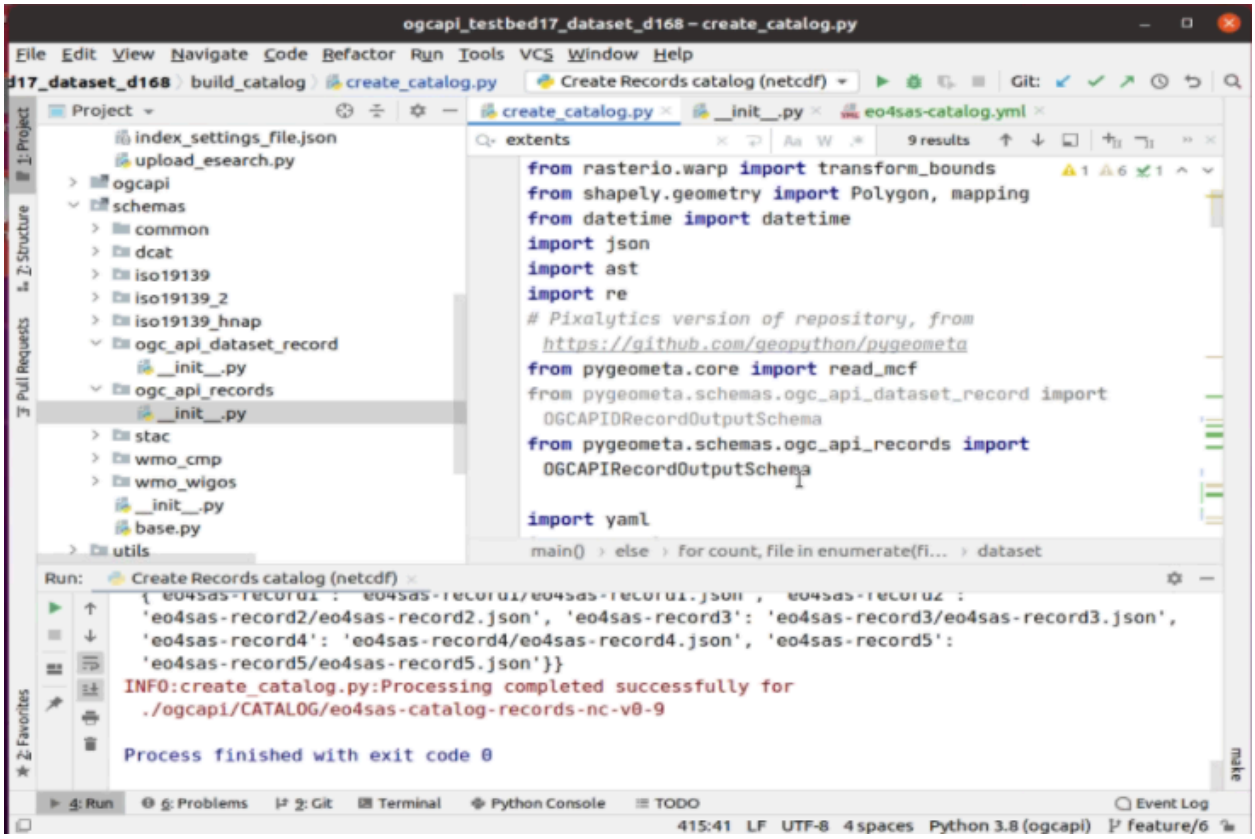

Figure 8 — Screenshot of part of a script from the Testbed-17 Dataset testing (D168) prototype

5.5. 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.

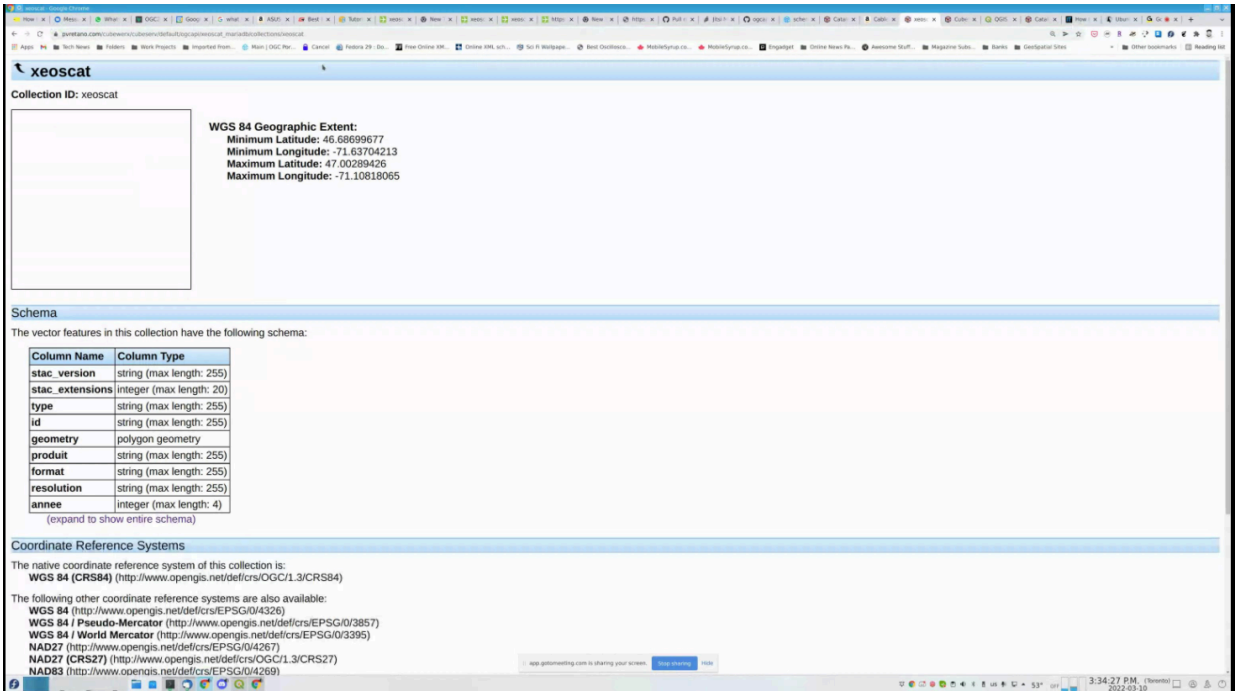

Figure 9 — Screenshot of the CubeWerx CubeSERV demo

5.6. Ecere GNOSIS Cartographer

Developers from Ecere supported the code sprint through provision of client-side and server-side implementations of various OGC API candidate and approved standards. Ecere demonstrated visualization of data accessed via OGC APIs through GNOSIS Cartographer.

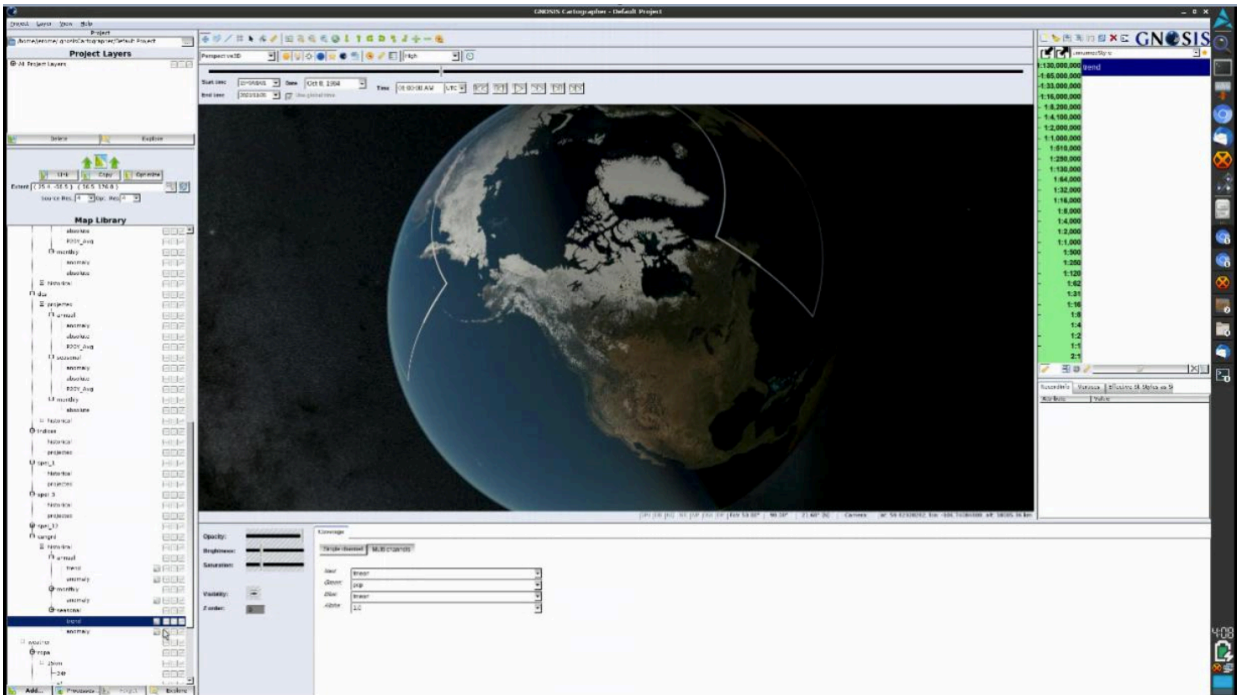

Figure 10 — Screenshot of the Ecere GNOSIS Cartographer demo

Ecere also provided an instance of the GNOSIS Maps Server, which implements multiple OGC API candidate and approved standards. Other sprint participants were then able to interact with GNOSIS Maps Server to help build OGC API support in their client applications, for example leaflet.

5.7. ZOO Project

The maintainers of ZOO Project worked on an extension of ZOO Project that enabled third party software to interoperate with an instance of ZOO Project. The third party software products integrated with ZOO Project during the code sprint included RabbitMQ and OSGeo MapServer.

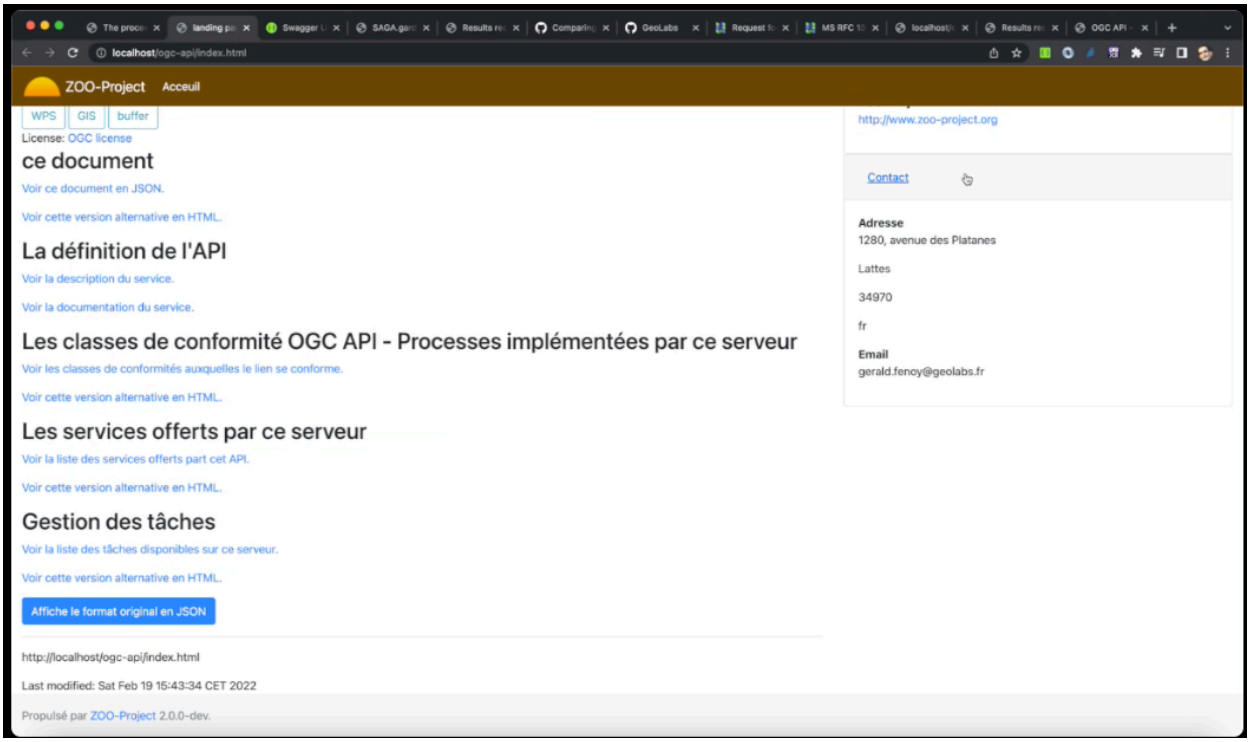

Figure 11 — Screenshot of the landing page of an instance ZOO Project

5.8. Integration with OSGeo MapServer

The integration with OSGeo MapServer made it possible for the results of a process that had been executed through the OGC API — Processes interface of ZOO Project to be automatically configured for display in an instance of OSGeo MapServer.

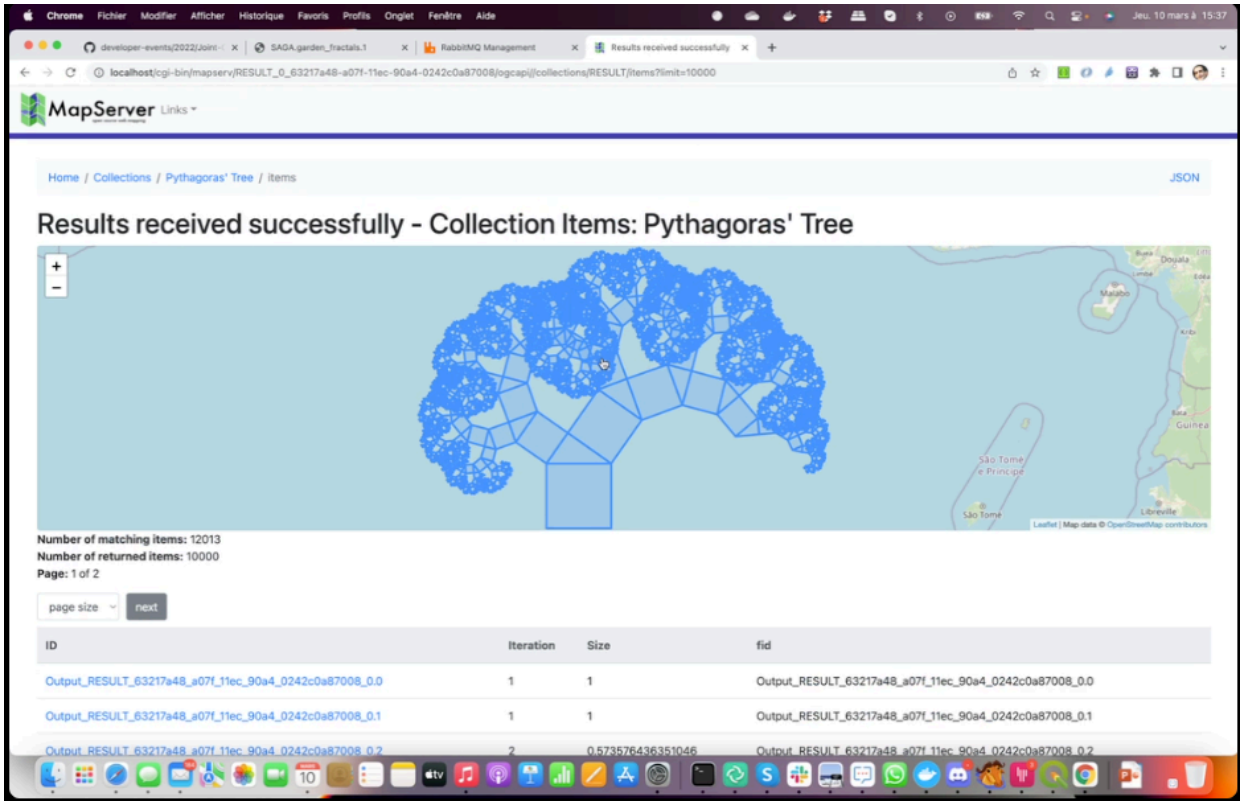

Figure 12 — Screenshot of an instance of MapServer interoperating with an instance of ZOO Project

5.9. Integration with RabbitMQ

The integration with RabbitMQ made it possible to use the RabbitMQ product for invoking a process that is available through the OGC API — Processes interface of ZOO Project.

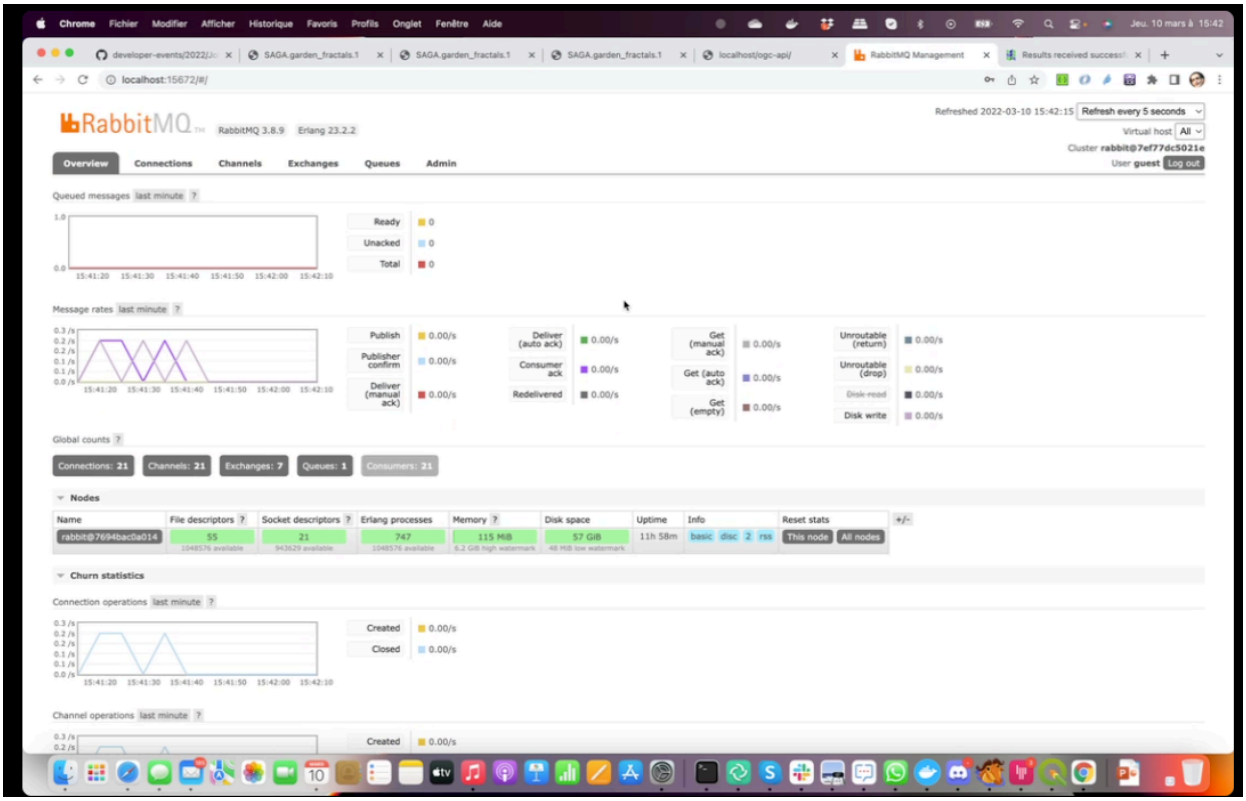

Figure 13 — Screenshot of the interface of RabbitMQ

5.10. Geospatial Integrity of Geoscience Software (GIGS)

The Geospatial Integrity of Geoscience Software (GIGS) tests are the products of an initiative of the International Association of Oil & Gas Producers (IOGP) to provide tests for coordinate operations. GIGS datasets are available as CSV files on the IOGP-GIGS GitHub repository. The aim of the GIGS activity during this code sprint was to convert the CSV files into implementation-neutral executable tests. The Apache SIS project has been used for testing the tests, but those tests do not contain any dependency to SIS. Implementation-neutral tests can be realized by a web service, and efforts in this direction are already underway at IOGP. But implementation-neutral tests (or other services) can also be done through programming interfaces.

At the time of writing, the only programming language for which an OGC Standard API exists is the Java language (note: it does not imply that implementations must be in Java; an implementation can be in C/C++ with wrappers such as PROJ-JNI). A Python binding of GeoAPI is in development but not yet submitted to the OGC GeoAPI SWG for approval.

The executable tests are implemented as JUnit 5 tests and can be executed by any implementation of GeoAPI 3.0.1 Java interfaces. A future version may allow execution of any implementation of Python interfaces if those interfaces (after completion) are approved as an OGC standard. The tests can be either integrated in the test suite of a project (e.g. using

Maven, but other build tools work as well), or be executed by a Graphical User Interface (GUI) shown below. When using the GUI, the user selects the JAR file of the library to test. The implementation entry points are automatically discovered using the standard `java.util.ServiceLoader` mechanism of the Java platform. Optional metadata such as implementation name and version are fetched from the standard `MANIFEST.MF` file bundled in the specified JAR file.

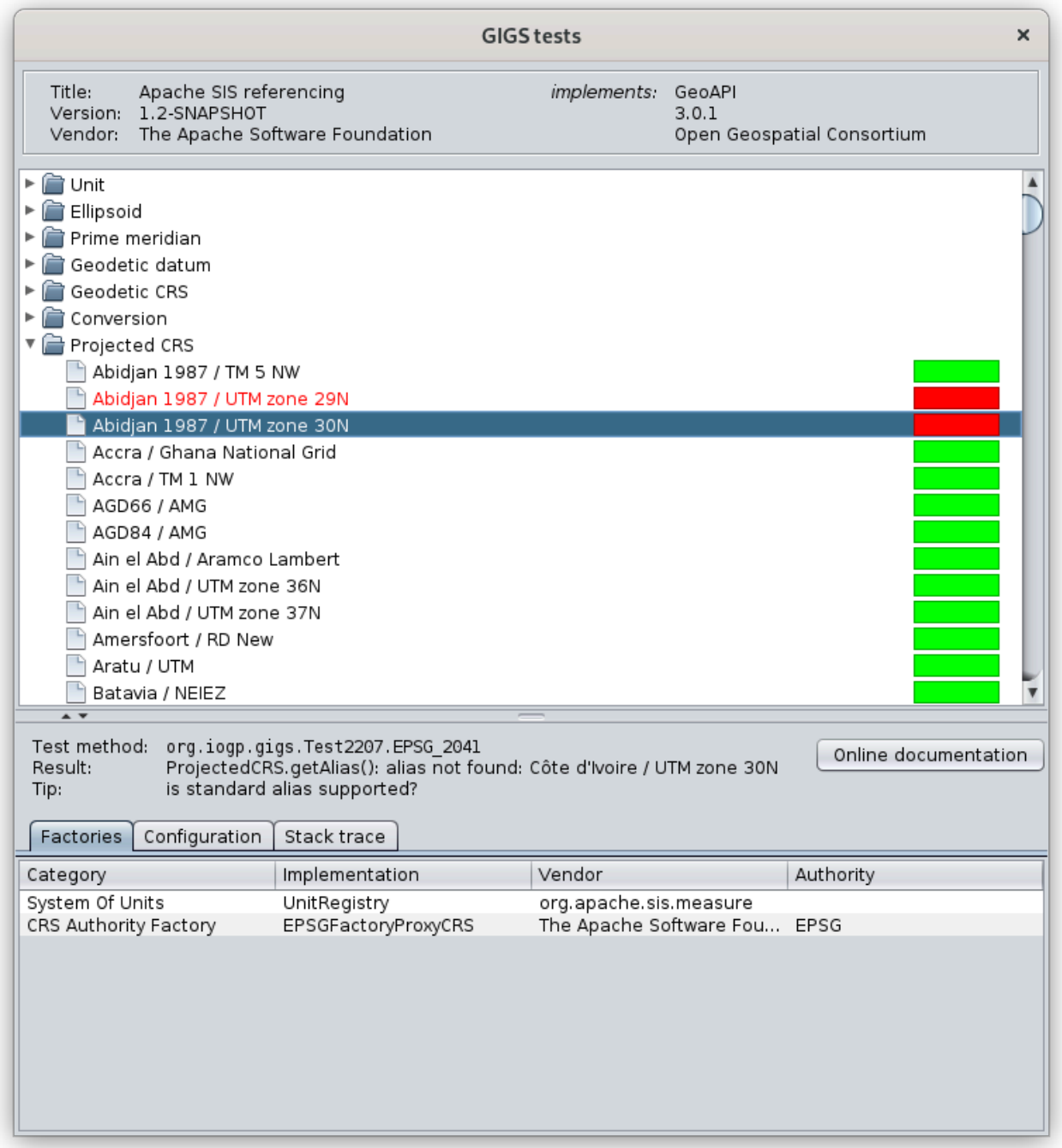

Figure 14 — Screenshot of GIGS test execution on Apache SIS library

Tests are configurable. Configuration options include:

- Whether names of objects (datum, CRS…) in the implementation are the names defined by EPSG.
- Whether the list of aliases for each object includes at least the aliases defined by EPSG.
- Whether dependencies (e.g. the datum in a CRS) also use the EPSG codes, names and aliases.
- Whether the implementation can still create objects that have been deprecated by EPSG.
- Whether the implementation has information about which version of a coordinate operation is used.
- And more…

Configuration options can be modified on a case-by-case basis in a project (not yet through the GUI). In the above screenshot, where a test was failing because a “Côte d’Ivoire” alias was missing, it would be possible to disable the alias check only for that particular CRS. When a test has some disabled checks, the green bar indicating a successful test becomes shorter.

GIGS tests are divided into series:

- Series 2200 tests the capability of an implementation to build various kinds of objects (datum, CRS, operation…) from EPSG codes, and checks that the object matches EPSG definitions.
- Series 3200 tests the capability of an implementation to build various kinds of objects from custom parameters (not necessarily matching an object from the EPSG database).
- Series 5100 tests various coordinate operations (including map projections) and checks that the implementation produces the expected values.
- Series 5200 tests various coordinate transformations (including datum shifts) and checks that the implementation produces the expected values.
- Series 5300, 5400 and 5500 tests operations with seismic and Wells data.
- Series 7000 checks that some deprecated objects are marked as such.

In this code sprint, we completed all Series 2200 tests. Series 3200 tests are available from GIGS version 1, but have not yet been ported to version 2. A little bit of time has been invested in making the GUI test runner more usable.

6

DISCUSSION

6 DISCUSSION

6.1. OGC API — Records

The participants prototyped implementations of a static catalogue. The idea of a crawlable catalogue is that the catalogue can be created using static files in accessible locations and then to have records linked together. It is therefore more of a manifest or a register. This approach is useful for publishing metadata records on Cloud storage services such as Amazon S3. There was a discussion around having fixed various files in a crawable catalogue. It was agreed that it would be better for the Standards to only make a recommendation, and then allow the implementer to decide on what names to give to the files.

There was discussion about schemas, including around the nesting of the `properties` keys i.e. one provided by GeoJSON and the one provided by OGC API — Records. There was also a discussion around some of the structures of STAC, for example the STAC Catalogue. It was agreed that the STAC Catalogue structure is a useful structure that could inform the approach taken for OGC API — Records.

The participants discussed the challenge of identifying a profile or extension of GeoJSON through a media type, when the media type is not registered with IANA. This topic came about because of work undertaken during the code sprint to implement an Executable Test Suite for the EO Dataset Metadata GeoJSON (-LD) Standard (OGC 17-003r2). If a conformance class for the EO Data Metadata standard is specified in a future extension of OGC API — Records, then a way to identify the profile of GeoJSON requested by the client from the server would need to be specified.

6.2. OGC API — Routes

There was discussion around the Route Exchange Model (REM). One of the observations made by sprint participants was that attaching the units of a value to a property name (e.g. `length_m`) could potentially lead to ambiguity (e.g. whether the `_m` stands for meters or miles).

Another observation was that assigning a value of `Segment` to a feature with a `Point` geometry could lead to confusion amongst implementers that use Graph databases to calculate routes. This is because in Graph theory Nodes are typically represented as points, and Edges are typically represented as lines.

There was a suggestion that a proposal should be submitted to the Google Summer of Code (GSOC) program for a project that implements OGC API — Records and/or the Route Exchange Model in the open source pgRouting extension of PostgreSQL/PostGIS.

6.3. OGC API — Environmental Data Retrieval

Results of the extension of the Testbed-17 Dataset testing (D166) prototype led to a discussion with the editors of the OGC API — Environmental Data Retrieval standard and a Pull Request was created for updating the API definition files of the standard. It was noted that the modification might also be relevant to OGC API — Common.

6.4. OGC API — Maps and OGC API — Common

One of the sprint participants, a contributor to Leaflet, observed that the implementation of OGC API — Maps that they were interacting with did not offer much attribution.

A summary of the issue recorded by the participants is below. The issue related to attribution on both the list of collections, and the metadata of a collection:

- Clarity is needed on whether attribution belongs to OGC API — Common — Part 1 (Core) or Part 2 (Geospatial Data). Currently it appears there can be an attribution field in either/both the list of collections (Part 1) or in the metadata of a specific collection (Part 2). Therefore, clarity is needed on what the behavior of a client should be — shall a client display attribution about the service, or only about the collections being displayed/used? Allowing attribution only in Part 2, or clarifying how the attribution in Part 1 would be displayed would solve this concern.
- Allowing servers to use any kind of HTML markup in the attribution field might be problematic due to three specific problems: <script>s, tracking pixels, and parsing. In a best-case scenario, an OGC API server might push some invisible tracking pixels to clients (an image linking to an invisible 1×1 pixel). While good-intentioned, this opens a rabbit hole of cookies and privacy concerns. Another concern is clients with no means of parsing HTML and building a document model about it. Using a specific format of markup will force clients to include a parser for that markup language. A similar concern would arise if another markup language would be used, e.g. markdown. The worst-case scenario would be an infosec attacker gaining a hold of an OGC API server, and changing the attribution field to include a <script> reference. That would be an attack vector to inject malware in any clients. Therefore, there appears not to be any scenario where allowing <script> as part of attribution might be useful.

6.5. Editorial

The sprint participants also identified a number of editorial issues in draft specifications and recorded the issues for Standards editors to address. Examples include:

- References to example files of the draft OGC API — Records — Part 1: Core
- Use of the correct case in link relations in a draft future extension of OGC API — Routes
- Documentation in yaml files of the draft OGC API — Maps — Part Core

6.6. Summary of Lessons Identified

The following are the immediate lessons identified by the sprint participants:

- There is concern about the format for presenting attribution of collections and collection metadata. There is a need for some compromise on the attribution format.
- The code sprints provide an opportunity for feedback from more developers within and outside of the OGC, ASF, and OSGeo communities.
- There is a need to always have a bbox response in the HTTP header, to help with determining the extent of the data returned.
- OGC API Standards offer an improved developer experience compared to the OGC Web Services suite of standards.
- The examples in OGC API Standards are absolutely key to helping developers to implement the standards.
- There is a frequently asked question of: What is the relationship between OGC API Records and STAC? There is a need for clear communications on what the relationship is and which standard to use where. There is a need to delineate these from both a communications and technical standpoint.
- There is a need to make OGC API Standards stand out.
- There is some similarity between STAC and some of the OGC API Standards. STAC also uses OGC API Features. The idea is to specify OGC API Records such that STAC can be a profile of OGC API Records.
- With some resources, it is not sufficient to record a link to a resource. For example, with a WMS there would be a need to parse the capabilities.

Whereas some of the lessons listed above might have implications for standards, others may have implications for developer guide documentation and similar resources. This highlights why conducting code sprints that involve open standards and open source software communities is critical for the uptake of standards.

7 CONCLUSIONS

7 CONCLUSIONS

The code sprint facilitated the development and testing of prototype implementations of OGC Standards, including implementations of draft OGC API Standards. Further, the code sprint also enabled the participating developers to provide feedback to the editors of OGC Standards. Furthermore, the code sprint provided a collaborative environment for OSGeo and ASF developers to fix open issues in products, develop new features, improve documentation, improve interoperability with other libraries/products, and develop prototype implementations of OGC standards. The code sprint therefore met all of its objectives and achieved its goal of accelerating the support of open geospatial standards within the developer community.

7.1. Future Work

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

- Prototypes of catalogues that can be crawled through by an application. Currently there are several searchable catalogues but not catalogues that can be crawled through by an application.
- More specification validation work for OGC API Records.
- More experiments for the Workflows extension of OGC API Processes. This could try out the various workflow approaches. https://github.com/opengeospatial/ideas/issues/115
- Experimentation on how a processing server can interact properly with other OGC API implementations that serve data. For example, in this code sprint there was an implementation of OGC API Processes (ZOO Project) that interacted with an OGC API Features implementation (MapServer).
- Experimentation with OGC's geoparquet specification and Apache Arrow.

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

- To improve examples and documentation related to OGC API Records.
- To advance the development of the Executable Test Suites of OGC API Processes.
- To advance the development of the Executable Test Suites of OGC API Tiles.
- To advance the development of the Executable Test Suites of OGC API Coverages.

A

ANNEX A (INFORMATIVE) REVISION HISTORY

A ANNEX A (INFORMATIVE) REVISION HISTORY

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

2. Aleksandar Balaban: 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