Joint OGC OSGeo ASF Code Sprint 2021 Summary Engineering Report

Publication Date: YYYY-MM-DD

Approval Date: YYYY-MM-DD

Submission Date: YYYY-MM-DD

Reference number of this document: OGC XX-XXX

Reference URL for this document: http://www.opengis.net/doc/PER/<initiative_short_name>-ID

Category: OGC Public Engineering Report

Editor: Gobe Hobona, Angelos Tzotsos, Tom Kralidis and Martin Desruisseaux

Title: Joint OGC OSGeo ASF Code Sprint 2021 Summary Engineering Report

---

OGC Public Engineering Report

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.

Table of Contents

Chapter 1. Subject

The subject of this Engineering Report (ER) is a code sprint that was held from 17 to 19 February 2021 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 event was sponsored by Ordnance Survey (OS) and GeoCat BV, and held as a completely virtual event.

Chapter 2. Executive Summary

This Engineering Report (ER) summarizes the main achievements of the Joint OGC OSGeo ASF Code Sprint, conducted between February 17th and 19th, 2021. The sprint served to accelerate the support of open geospatial standards within the developer community.

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

The Open Geospatial Consortium (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 the draft specifications that evolve to become standards. The sprint objectives for the SWGs were:

- Develop prototype implementations of OGC standards, including implementations of draft OGC APIs 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 813 individual Members and 8,000 Committers on six continents stewarding more than 200 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 (GeoSPARQL, Filters, …)
- Improve visualisation capabilities (map, …)
- Improve documentation (web site, …)
- Improve interoperability with other libraries (GeoAPI)

The code sprint facilitated the development and testing of prototype implementations of OGC standards, including implementations of draft OGC APIs 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:

- Themes, trees, nesting, and other strategies need to be explored. This is needed because data providers often have thousands of datasets that they need to manage or publish.
- There is a need for more experimentation on styles and coverages. Including how styles can be used to render/filter coverages.
- Tiled coverages and their support through OGC API - Coverages and OGC API - Tiles integration should be explored further.
- More experimentation is needed on workflows in relation to the OGC API - Processes - Part 3 : Workflows.
- Further development of the DGGS API, including work on client applications and visualization.
- There is a need to advance OGC APIs and other OGC standards to enable the cataloguing and discovery of models (e.g. AI/Machine Learning models). We need to explore how to sufficiently describe the models.
- The implications of OpenAPI 3.1, JSON Schema and Webhooks need to be examined and path towards transition identified.
- Some integration of the MapML format concept with the OGC API offerings, for example into the HTML representation of features, to enhance the spatial indexing of HTML.
- Enhancement of the Link Relations Register.

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

- Associations between records and other elements in catalogues

- Landing page of landing pages
- Searchable collections in OGC APIs (including the collections of collections)
- Where appropriate, clarification that GeoJSON is the default JSON encoding for OGC API - Features and OGC API - Records

2.1. Document contributor contact points

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

Contacts

Name	Organization	Role
Gobe Hobona	Open Geospatial Consortium	Editor
Angelos Tzotsos	Open Source Geospatial Foundation	Editor
Tom Kralidis	Meteorological Service of Canada	Editor
Martin Desruisseaux	Geomatys	Editor
Clemens Portele	interactive instruments GmbH	Contributor
Paul van Genuchten	GeoCat BV	Contributor
Mark Thomas	Apache Software Foundation	Contributor
Sean Arms	UCAR	Contributor
Jamie Goodyear	Apache Software Foundation	Contributor
Hisham Massih	Esri	Contributor
Matt Pavlovich	ASF / HYTE Technologies, Inc.	Contributor
Charles Heazel	Heazeltech LLC	Contributor
Jeffrey Yutzler	Image Matters	Contributor
Edward Lewis	British Geological Survey	Contributor
Bruno Kinoshita	Apache Software Foundation	Contributor
Yugandhar Thippireddy	accenture	Contributor
Mahmoud SAKR	Université Libre de Bruxelles	Contributor

Name	Organization	Role
Rajveer Shekhawat	Manipal University Jaipur	Contributor
Shivashis Padhi	Individual	Contributor
Pongsakorn Udombua	i-bitz company limited.	Contributor
Patrick Dion	Ecere Corporation	Contributor
Sattawat Arab	i-bitz	Contributor
Anon Bianglae	i-bitz	Contributor
Pandu Wicaksono	Badan Pusat Statistik	Contributor
Francis Charette Migneault	Computer Research Institute of Montréal (CRIM)	Contributor
Massimo Di Stefano	Met.no	Contributor
Marco Neumann	KONA	Contributor
Núria Julià	UAB-CREAF	Contributor
Michel Gabriël	GeoCat	Contributor
Vaclav Petras	NC State University	Contributor
Anna Petrasova	NC State University	Contributor
Jeff McKenna	GatewayGeo	Contributor
Matthew Purss	Pangaea Innovations Pty. Ltd.	Contributor
Joan Maso	UAB-CREAF	Contributor
Panagiotis Vretanos	CubeWerx Inc.	Contributor
Ahmad Ayubi	Natural Resources Canada	Contributor
Florian Hoedt	Thünen-Institute	Contributor
Peter Rushforth	Natural Resources Canada	Contributor
Brian Hamlin	Open Source Geospatial Foundation	Contributor
Jody Garnett	GeoCat	Contributor
luca delucchi	Fondazione Edmund Mach	Contributor
Carsten Ehbrecht	DKRZ	Contributor
Chris Little	Met Office	Contributor
Bo Lu	Natural Resources Canada	Contributor

Name	Organization	Role
James Munroe	Elemental Earth Data Ltd.	Contributor
Sander Schaminee	GeoCat BV	Contributor
Paul Hershberg	NOAA	Contributor
Steve Olson	NOAA/NWS	Contributor
Nattapat Phumphan	i-bitz company limited	Contributor
Nutthapol Jansuri	I-bitz	Contributor
Prasong Patheepphoemphong	i-bitz company limited	Contributor
Davince Koyo	Individual	Contributor
Rajat Shinde	Indian Institute of Technology Bombay	Contributor
Andrea Aime	GeoSolutions	Contributor
Shane Mill	NOAA	Contributor
Michael Rushin	George Mason University	Contributor
Nazih Fino	Global Nomad GIS Services	Contributor
Francois Prunayre	titellus	Contributor
Ingrid Santana	UFMG	Contributor
Irene Muema	8teq Technologies	Contributor
Julia Wakaba	8teq	Contributor
Joseph Kariuki	AthenaSl	Contributor
Luke Hodgaon	TPG	Contributor
Steven McDaniel	Hexagon Geospatial	Contributor
Richard Mitanchey	Cerema	Contributor
Pankaj Kumar	https://geoknight.medium.com/	Contributor
Just van den Broecke	Just Objects B.V.	Contributor
Steve Ochieng	Individual	Contributor
Alexander Kmoch	University of Tartu	Contributor
Qianqian Zhang	China Agricultural University	Contributor
Benard Odhimabo	8-teq	Contributor
Alex Mandel	Development Seed	Contributor
Ashish Kumar	IIT (BHU) Varanasi	Contributor

Name	Organization	Role
Martha Vergara	Open Source Geospatial Foundation	Contributor
Gérald Fenoy	GeoLabs	Contributor
Jerome St-Louis	Ecere Corporation	Contributor
Richie Carmichael	Esri	Contributor
Dave McLaughlin	Penn State University	Contributor
Kathleen Schaefer	UC Davis	Contributor
Vicky Vergara	georepublic/OSgeo/pgRouting	Contributor
Srini Kadamati	Preset	Contributor
Ignacio "Nacho" Correas	Skymantics	Contributor
BENAHMED DAHO Ali	TransformaTek	Contributor
Oscar Diaz	Geosolutions Consulting	Contributor
Francesco Bartoli	Geobeyond Srl	Contributor
Astrid Emde	Open Source Geospatial Foundation	Contributor

2.2. Foreword

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.

Chapter 3. References

The following normative documents are referenced in this document.

- OGC: OGC 17-069r3, OGC API - Features - Part 1: Core 1.0 Standard [http://docs.ogc.org/is/17-069r3/17-069r3.html]
- OGC: OGC 21-000,OGC API - Routes - Part 1: Core 1.0 (Draft) [http://docs.ogc.org/DRAFTS/21-000.html]
- OGC: OGC 15-078r6, OGC SensorThings API Part 1: Sensing 1.0 Standard [http://docs.opengeospatial.org/is/15-078r6/15-078r6.html]
- OGC: OGC 09-083r4, GeoAPI 3.0 Implementation Standard with corrigendum 3.0.1 [https://portal.ogc.org/files/?artifact_id=71648]
- OGC: OGC 19-087, OGC API - Coverages - Part 1: Core 1.0 (Draft) [http://docs.ogc.org/DRAFTS/19-087.html]
- OGC: OGC 20-058, OGC API - Maps - Part 1: Core 1.0 (Draft) [http://docs.ogc.org/DRAFTS/20-058.html]
- OGC: OGC 20-057, OGC API - Tiles - Part 1: Core 1.0 (Draft) [http://docs.ogc.org/DRAFTS/20-057.html]
- OGC: OGC 20-009, OGC API - Processes - Part 1: Core 1.0 (Draft) [http://docs.ogc.org/DRAFTS/18-062.html]
- OGC: OGC 20-009, OGC API - Styles - Part 1: Core 1.0 (Draft) [http://docs.ogc.org/DRAFTS/20-009.html]
- OGC: OGC 19-072, OGC API - Common - Part 1: Core 1.0 (Draft) [http://docs.ogc.org/DRAFTS/19-072.html]
- OGC: OGC 20-004, OGC API - Records - Part 1: Core 1.0 (Draft) [http://docs.ogc.org/DRAFTS/20-004.html]
- OGC: OGC 19-086, OGC API - Environmental Data Retrieval 1.0 (Draft) [http://docs.ogc.org/DRAFTS/19-086.html]
- OpenAPI Specification [https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.1.md]
- Map Markup Language (MapML) [https://maps4html.org/MapML/spec/]

Chapter 4. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- *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).

- *coordinate reference system*

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

- *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)

- *Web API*

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

4.1. Abbreviated terms

NOTE: The abbreviated terms clause gives a list of the abbreviated terms and the symbols necessary for understanding this document. All symbols should be listed in alphabetical order. Some more frequently used abbreviated terms are provided below as examples.

- **API** Application Programming Interface
- **ASF** Apache Software Foundation
- **CRS** Coordinate Reference System
- **OGC** Open Geospatial Consortium
- **OSGeo** Open Source Geospatial Foundation

Chapter 5. Overview

Section 6 introduces the code sprint and lists the affiliations of the registered participants.

Section 7 presents the high level architecture of the code sprint, and describes the software products and OGC standards that were deployed in the code sprint.

Section 8 presents a summary of the results of the code sprints.

Section 9 discusses some of the issues raised and explored during the code sprint.

Section 10 summarises the main conclusions and makes recommendations for future work.

Chapter 6. Introduction

This Engineering Report (ER) summarizes the main achievements of the Joint OGC OSGeo ASF Code Sprint, conducted between February 17th and 19th, 2021. Sponsored by Ordnance Survey (OS) and GeoCat BV, 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 [1].

6.1. Participants

The code sprint had 87 registered participants. Software developers and solutions architects from the following organizations registered to participate in the code sprint:

- 8teq Technologies
- accenture
- HYTE Technologies, Inc.
- Apache Software Foundation
- AthenaSl
- Badan Pusat Statistik
- British Geological Survey
- Cerema
- China Agricultural University
- Computer Research Institute of Montréal (CRIM)
- CubeWerx Inc.
- Development Seed
- DKRZ
- Ecere Corporation
- Elemental Earth Data Ltd.
- Esri
- Fondazione Edmund Mach
- GatewayGeo
- Geobeyond Srl

- GeoCat BV
- GeoLabs
- Geomatys
- georepublic/OSgeo/pgRouting
- George Mason University
- GeoSolutions
- Global Nomad GIS Services
- Heazeltech LLC
- Hexagon Geospatial
- geoknight
- i-bitz company limited
- IIT (BHU) Varanasi
- Image Matters
- Indian Institute of Technology Bombay
- interactive instruments GmbH
- Just Objects B.V.
- KONA
- Manipal University Jaipur
- Met Office
- Met.no
- Meteorological Service of Canada
- Natural Resources Canada
- NC State University
- NOAA NWS
- Open Geospatial Consortium
- Open Source Geospatial Foundation
- Pangaea Innovations Pty. Ltd.
- Penn State University
- Preset
- Skymantics

- Thünen-Institute
- titellus
- TPG
- TransformaTek
- UAB-CREAF
- UC Davis
- UFMG
- Université Libre de Bruxelles
- University of Tartu

Chapter 7. High Level Architecture

The focus of the sprint was on support of the development of the open geospatial standard across various open source software projects. Implementations of these draft standards were deployed in participants' own infrastructure in order to build a solution with the architecture shown below in Figure 1.

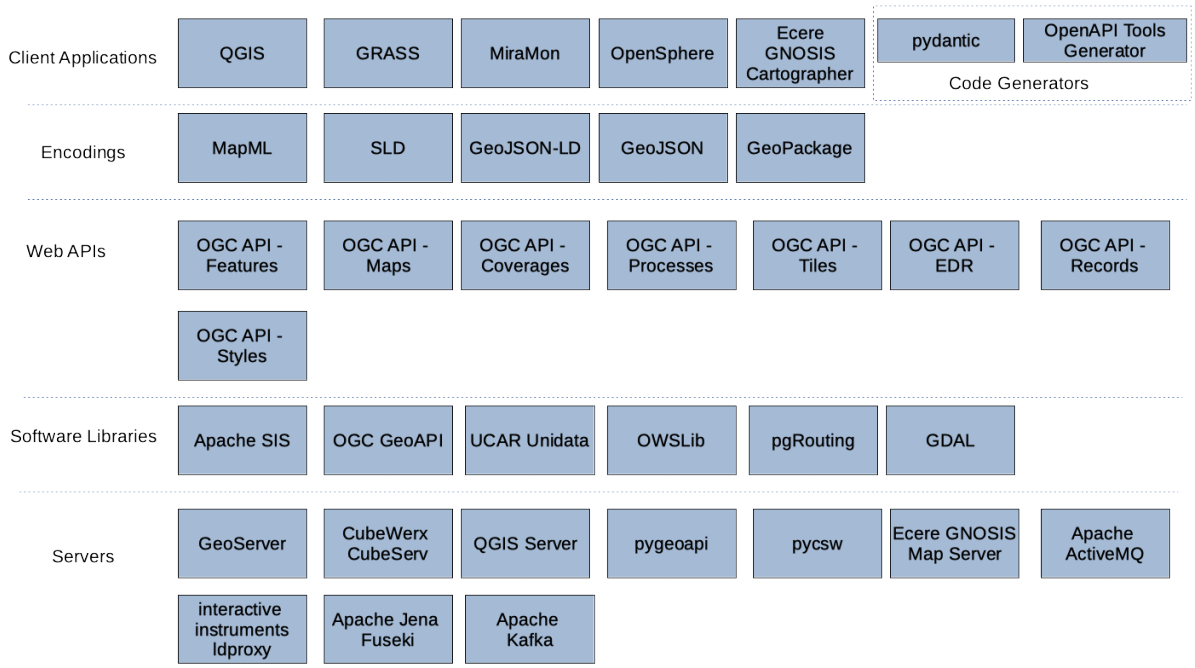


Figure 1. High Level Overview of the Sprint Architecture

As illustrated the sprint architecture was designed with the view of allowing 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.

7.1. Approved OGC Standards

7.1.1. 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. The specification is a multi-part document. Part 1, labelled the Core, describes the mandatory capabilities that every implementing service has to support and is restricted to read-access to spatial data. Part 2 enables the use of different Coordinate Reference Systems, in addition to the World Geodetic System 1984 (WGS 84). 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.

7.1.2. OGC Discrete Global Grid Systems (DGGS)

This standard specifies the core requirements and extension mechanisms for Discrete Global Grid Systems (DGGS). A DGGS is a spatial reference system that uses a hierarchical tessellation of cells to partition and address the globe. DGGS are characterized by the properties of their cell structure, geo-encoding, quantization strategy and associated mathematical functions. The OGC DGGS Abstract Specification supports the specification of standardized DGGS infrastructures that enable the integrated analysis of very large, multi-source, multi-resolution, multi-dimensional, distributed geospatial data. A prototype DGGS API [https://docs.ogc.org/per/20-039r2.html] was recently built in the OGC Testbed-16 initiative.

7.1.3. 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 language defined in several packages which interpret into Java the data model and UML types of the ISO and OGC standards documents. The library includes extensive Javadoc code documentation which complement the injunctions 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.

7.1.4. OGC GeoPackage

GeoPackage is an open, platform-independent, portable, self-describing, compact format for transferring geospatial information. The GeoPackage Encoding Standard governs the rules and requirements for storing content in an SQLite database. The content may include geospatial and non-geospatial content. The standard defines the schema for a GeoPackage, including table definitions, integrity assertions, format limitations, and content constraints.

7.2. Draft OGC Standards

7.2.1. OGC API - Common

The draft OGC API - Common standard documents 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 specification establishes a common pattern that is based on OpenAPI.

7.2.2. OGC API - Coverages

The OGC API - Coverages specification defines a Web API for accessing coverages that are modeled according to the Coverage Implementation Schema (CIS) 1.1 [http://docs.opengeospatial.org/is/09-146r6/09-146r6.html]. Coverages are represented by some binary or

ASCII serialization, specified by some data (encoding) 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. The OGC API - Coverages specification builds on the Web Coverage Service (WCS) standard.

7.2.3. 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. This draft specification identifies resources, captures compliance classes, and specifies requirements which are applicable to OGC Environmental Data Retrieval API's. The draft specification addresses two fundamental operations; discovery and query. 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.

7.2.4. 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. The draft standard can be considered the successor to the widely implemented WMS standard.

7.2.5. OGC API - Processes

The draft 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.

7.2.6. OGC API - Records

OGC API - Records provides discovery and access to metadata about geospatial data and services. OGC API standards define modular API building blocks to spatially enable Web APIs in a consistent way. OpenAPI is used to define the reusable API building blocks.

7.2.7. OGC API - Routes

OGC API - Routes describes the requirements for interoperable web-based route computation and specifies a number of alternative approaches to fulfill these requirements. One of the approaches is based on the current draft of the draft OGC API - Processes - Part 1: Core specification while the other comprises a specialized API although also based on the draft OGC

API - Common - Part 1: Core specification. Both approaches facilitate a common Route Exchange Model (REM) that is based on GeoJSON.

7.2.8. OGC API - Styles

OGC API - Styles describes the interface and exchange of styling parameters and instructions. The construction of symbology components of styles is addressed in OGC Symbol Encoding Conceptual Model: Core Part and multiple OGC and other style encoding standards.

7.2.9. OGC API - Tiles

OGC API - Tiles references the OGC Two Dimensional Tile Matrix Set (TMS) standard. The TMS standard defines the rules and requirements for a tile matrix set as a way to index space based on a set of regular grids defining a domain (tile matrix) for a limited list of scales in a Coordinate Reference System (CRS).

7.3. OSGeo Projects

7.3.1. OSGeo GeoNetwork

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

7.3.2. OSGeo GeoNode

GeoNode is a web-based application and platform for developing geospatial information systems (GIS) and for deploying spatial data infrastructures (SDI).

7.3.3. OSGeo GeoServer

GeoServer is a Java-based software server that allows users to view and edit geospatial data. Using open standards by the Open Geospatial Consortium (OGC), GeoServer allows for great flexibility in map creation and data sharing.

7.3.4. OSGeo GRASS GIS

The Geographic Resources Analysis Support System (GRASS) is an open source Geographic Information System (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 [2].

7.3.5. OSGeo Mapbender

Mapbender is a web based geoportal framework to publish, register, view, navigate, monitor and grant secure access to spatial data infrastructure services.

7.3.6. OSGeo MapServer

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

7.3.7. OSGeo OSGeoLive

OSGeoLive is a self-contained bootable DVD, USB thumb drive or Virtual Machine based on Lubuntu, that allows users to try a wide variety of open source geospatial software without installing anything.

7.3.8. OSGeo PostGIS

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

7.3.9. OSGeo Proj

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

7.3.10. OSGeo pycsw

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

7.3.11. OSGeo QGIS

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

7.4. OSGeo Community Projects

7.4.1. OSGeo GeoHealthCheck

GeoHealthCheck is a Python application to support monitoring OGC Web Services uptime and availability.

7.4.2. MapML extension for GeoServer

Map Markup Language (MapML) is a text-based format which allows map authors to encode map information as hypertext documents exchanged over the Uniform Interface of the Web.

7.4.3. OSGeo GeoTrellis

GeoTrellis is an open source, geographic data processing library implemented in Scala and designed to work with large geospatial raster data sets.

7.4.4. OSGeo mapproxy

MapProxy is an open source proxy for geospatial data. It caches, accelerates and transforms data from existing map services and serves different desktop or web GIS client applications.

7.4.5. OSGeo MobilityDB

MobilityDB is a database management system for moving object geospatial trajectories, such as GPS traces.

7.4.6. OSGeo OWSLib

OWSLib is a Python package for client programming with Open Geospatial Consortium (OGC) web service (hence OWS) interface standards, and their related content models.

7.4.7. OSGeo pdal

PDAL is a C++ BSD library for translating and manipulating point cloud data.

7.4.8. OSGeo pgRouting

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

7.4.9. OSGeo pygeoapi

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

7.4.10. OSGeo Stetl

Stetl, Streaming ETL, is an open source (GNU GPL) toolkit for the transformation (ETL) of geospatial data. Stetl is based on existing ETL tools like GDAL/OGR and XSLT.

7.4.11. OSGeo ZOO-Project

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

7.5. ASF Projects

7.5.1. Apache Activemq

Apache ActiveMQ™ is an open source, multi-protocol, Java-based messaging server. It supports industry standard protocols so users get the benefits of client choices across a broad range of languages and platforms.

7.5.2. Apache CXF

Apache CXF™ is an open source services framework. CXF helps you build and develop services using frontend programming APIs, like JAX-WS and JAX-RS.

7.5.3. Apache Jena

A free and open source Java framework for building Semantic Web and Linked Data applications.

7.5.4. Apache Karaf

Apache Karaf is a small OSGi based runtime which provides a lightweight container onto which various components and applications can be deployed.

7.5.5. Apache Superset

Apache Superset is an open-source software cloud-native application for data exploration and data visualization able to handle data at petabyte scale.

7.5.6. Apache Tomcat

Apache Tomcat is an open-source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and WebSocket technologies.

7.6. Other open source

7.6.1. 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, written in Java (Source Code [https://github.com/interactive-instruments/ldproxy]) and is typically deployed using docker (DockerHub [https://hub.docker.com/r/iide/ldproxy/]). 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 other draft extensions (including Part 3 and Part 4). ldproxy also has draft implementations for additional resource types (Tiles, Styles).

7.7. Proprietary products

7.7.1. CubeWerx CubeServ

The CubeWerx server ("cubeserv") is implemented in C and currently implements the following OGC web services:

- 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 specification.
- Multiple conformance classes and recommendations of the draft OGC API - Coverages - Part 1: Core specification.
- Multiple conformance classes and recommendations of the draft OGC API - Processes - Part 1: Core specification.
- 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, Web Object Service Implementation Specification.

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.

7.7.2. CREAF MiraMon Map Server

The MiraMon Map Server is a CGI application encoded in C language that is part of the MiraMon Geographic Information System (GIS) & Remote Sensing (RS) suite. Currently the server implements WMS, WMTS and partially implements WFS and WCS. It also partially implements the OGC Sensor Observation Service (SOS) standard. In order to perform efficiently, it requires a process preparing the data to be offered. It can interoperate with other vendors clients but combined with the MiraMon Map Client it offers more functionality, including functionality recently developed for the Catalan Data Cube.

The software originally started 10 years ago as a WMS server in support of the Catalan Administration and CREAF data services.

It includes prototype support for the draft OGC API - Maps and OGC API - Tiles specifications.

7.7.3. 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 http://maps.ecere.com/geoapi and has been used in multiple OGC Innovation Program initiatives.

7.7.4. NOAA NWS EDR API

The National Weather Service deployed an instance of the OGC API - Environmental Data Retrieval candidate standard. The implementation offers the following types of data products: MEteorological Terminal Aviation Routine Weather Report (METAR), Terminal Aerodrome Forecast (TAF), Global Forecast System(GFS), North American Mesoscale model (NAM), and National Digital Forecast Database (NDFD).

Chapter 8. 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.

Figure 2 shows a screenshot of OpenSphere displaying a DGGS that had been uploaded as a GeoJSON-encoded feature collection. On the screenshot, one of the grid cells is highlighted after being clicked on.

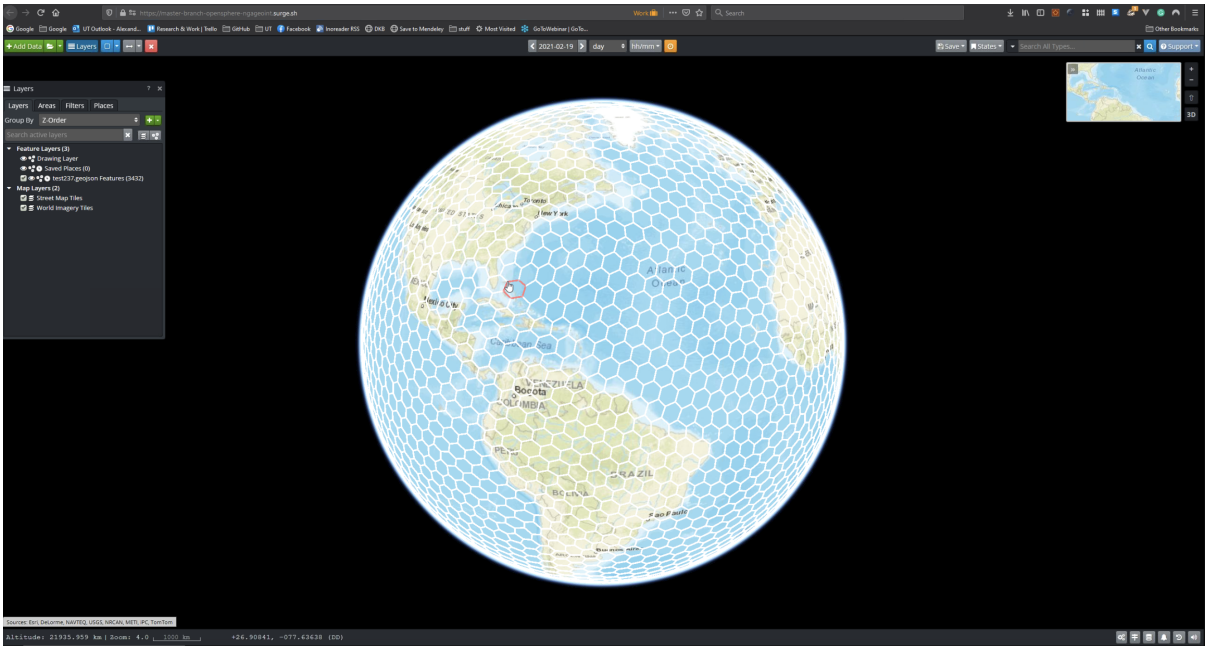


Figure 2. A screenshot of OpenSphere displaying a DGGS

Figure 3 shows a screenshot of the UAB-CREAF MiraMon application accessing resources through OGC API - Tiles and OGC API- Maps interfaces. The client application is shown retrieving content from Ecere and CubeWerx servers.

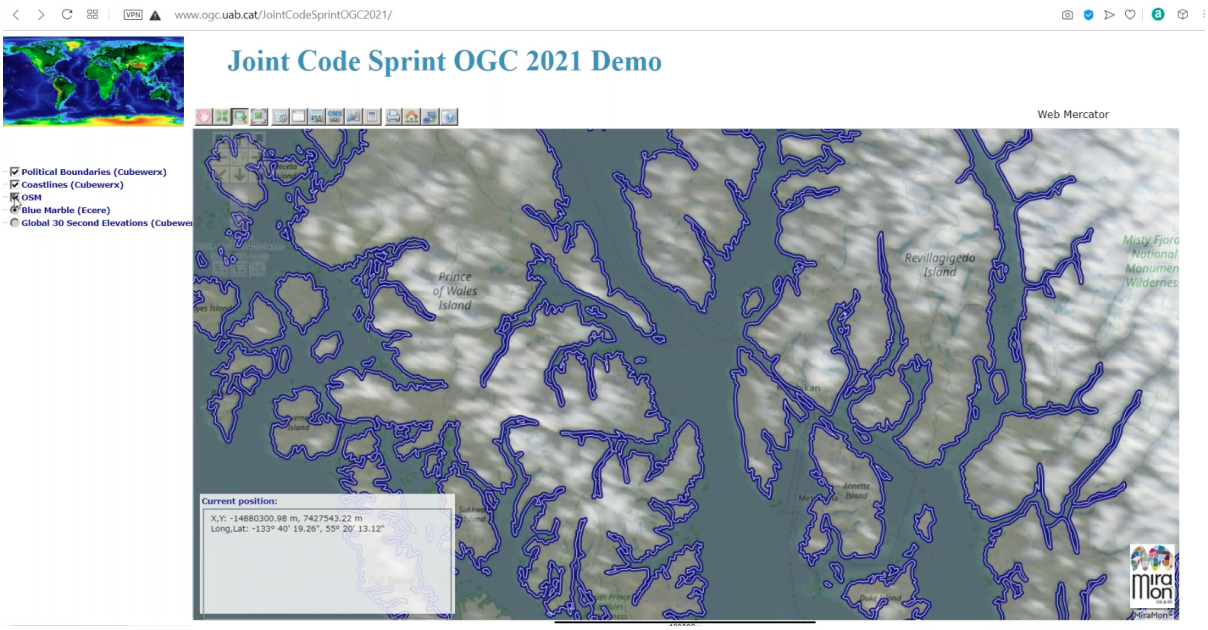


Figure 3. A screenshot of the UAB-CREAF MiraMon application accessing resources through OGC API - Tiles and OGC API- Maps interfaces

Figure 4 shows a screenshot of the collections offered by a GeoServer implementation of OGC API - Styles. The screenshot shows a series of styles belonging to one of the collections. This organization of styles makes it possible to have different styles for different contexts (e.g. light, day, outdoor and so forth).

Figure 4. A screenshot of the collections offered by a GeoServer implementation of OGC API - Styles

Figure 5 shows two screenshots, one above the other. The upper screenshot shows the Swagger user interface (UI) of a prototype implementation of OGC API - Processes by Hexagon Geospatial. The lower screenshot shows a series of in-progress and completed jobs as monitored by the implementation.

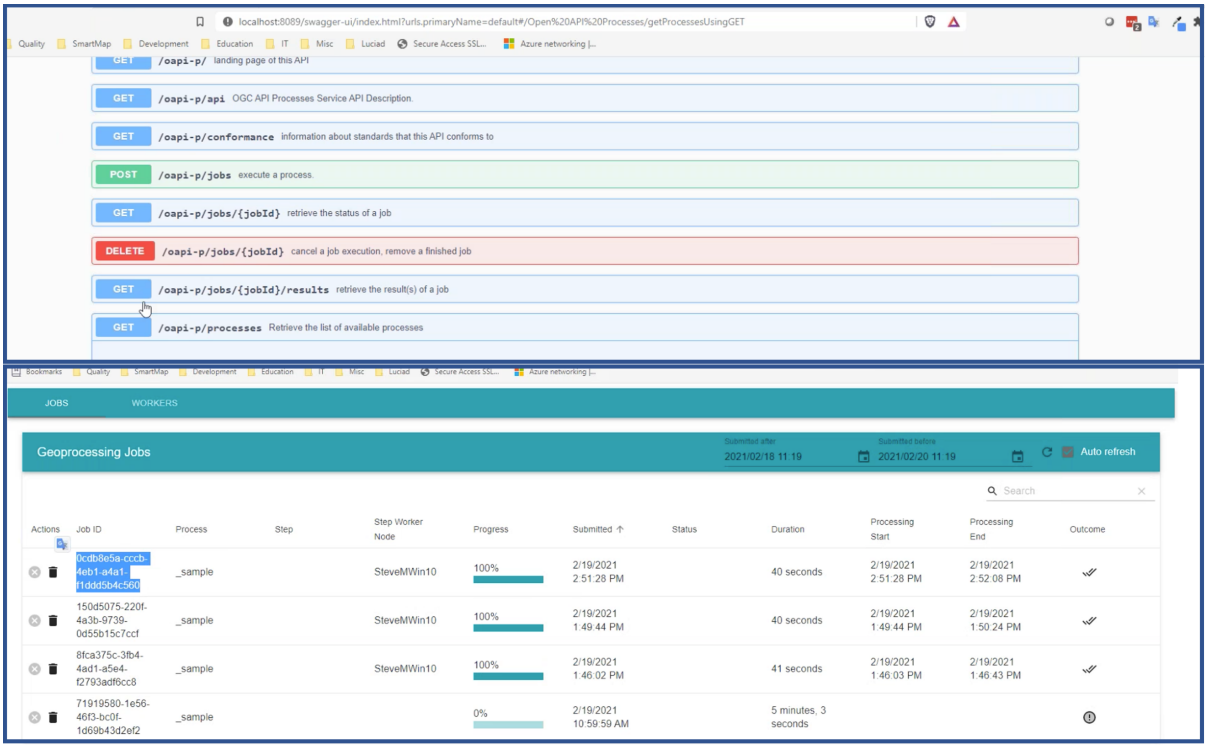


Figure 5. A screenshot of the Swagger UI of a prototype Hexagon implementation of OGC API - Processes

Figure 6 shows a screenshot of pygeoapi displaying part of the Uber H3 [https://eng.uber.com/h3/] Hexagonal Hierarchical Geospatial Indexing System. H3 supports hierarchical tessellation of regular polygons at increasingly fine resolutions up to an areal size of square meters [3].

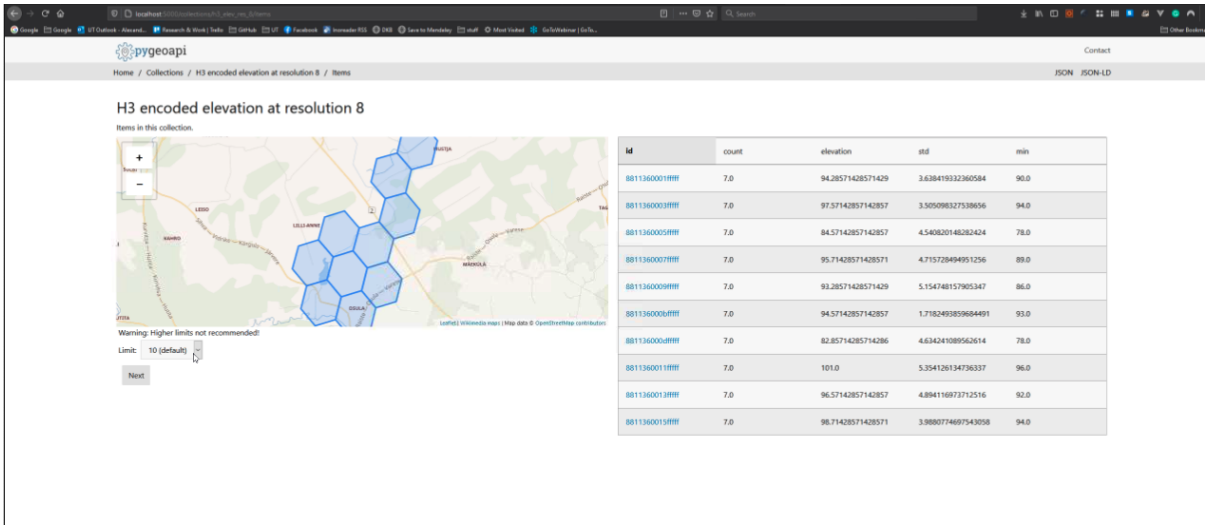


Figure 6. A screenshot of pygeoapi displaying a DGGS

Figure 7 shows a screenshot of a map created from OS Open Zoomstack [https://www.ordnancesurvey.co.uk/business-government/products/open-zoomstack] using a GeoServer instance that supports OGC API - Maps. OS Open Zoomstack offers comprehensive basemap of Great Britain showing coverage from national level right down to street detail.

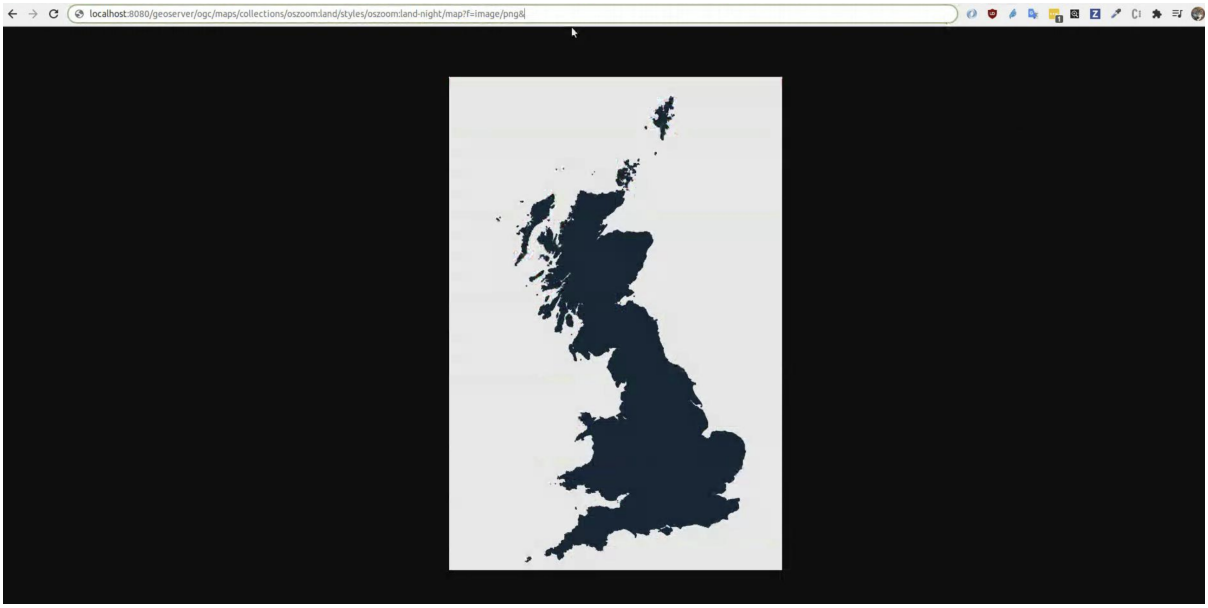


Figure 7. A screenshot of a map created from OS Open Zoomstack using a GeoServer and OGC API - Maps (Contains OS data © Crown Copyright and database right 2020)

Figure 8 shows a screenshot of the NetBeans IDE running GeoAPI, Apache SIS and the UCAR netCDF library. The use of these three libraries demonstrated support for both the OGC GeoAPI [https://www.ogc.org/standards/geoapi] standard and the OGC netCDF [https://www.ogc.org/standards/netcdf] standard.

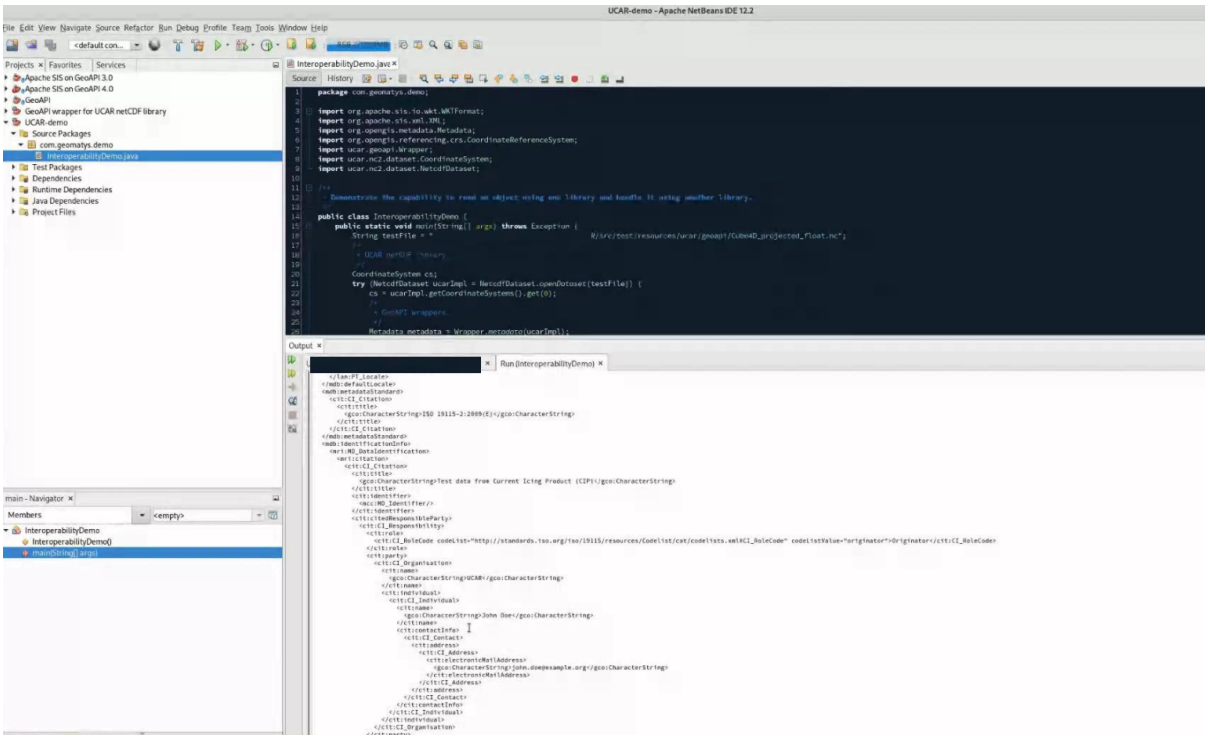


Figure 8. A screenshot of the NetBeans IDE running GeoAPI, Apache SIS and the UCAR netCDF library

Figure 9 shows a screenshot of pygeoapi displaying a sample metadata record from the Dutch National GeoRegister. The pygeoapi project formally completed OGC API - Records support in the software, and updated the pygeoapi demo [https://demo.pygeoapi.io/master/collections/dutch-metadata] for community testing and feedback.

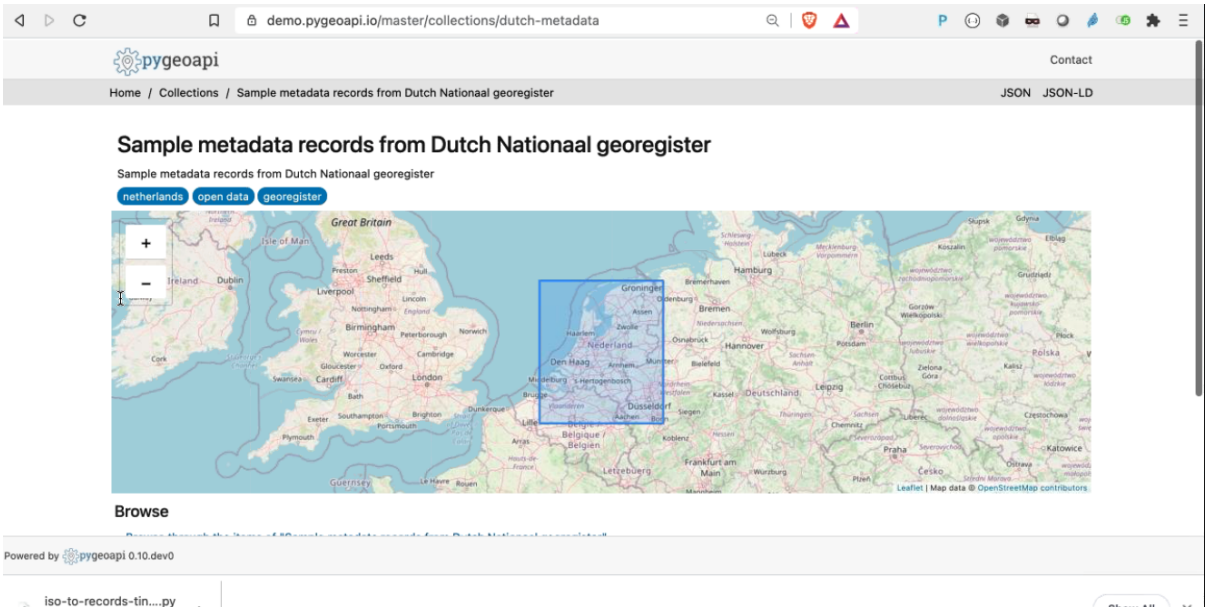


Figure 9. A screenshot of pygeoapi displaying a sample metadata record

Figure 10 shows a screenshot of GeoNetwork and the download buttons (right-hand side of the screen) for different supported formats. As shown on the screenshot the formats included HTML, XML, JSON, RSS and JSON-LD structured according to the schema.org specification.

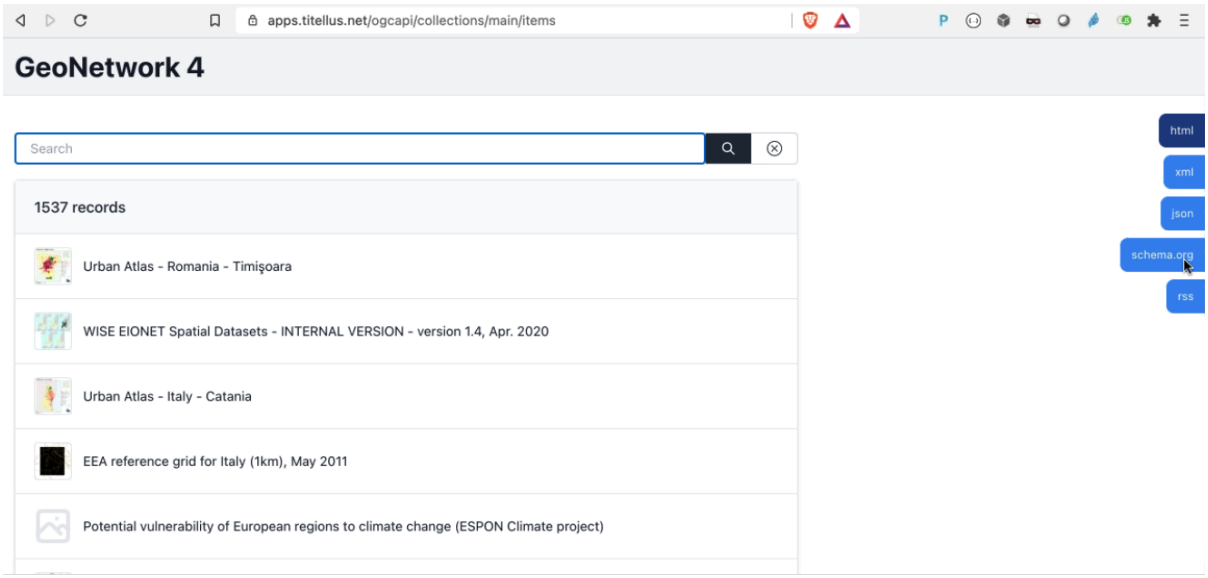


Figure 10. A screenshot of the GeoNetwork user interface

Figure 11 shows a screenshot of the MapML viewer built for GeoServer. The screenshot shows two separate layers in the same view, one showing part of the United States and the other showing Canada. The screenshot also shows a pop-up window triggered by a mouse click and revealing attributes about the clicked feature.

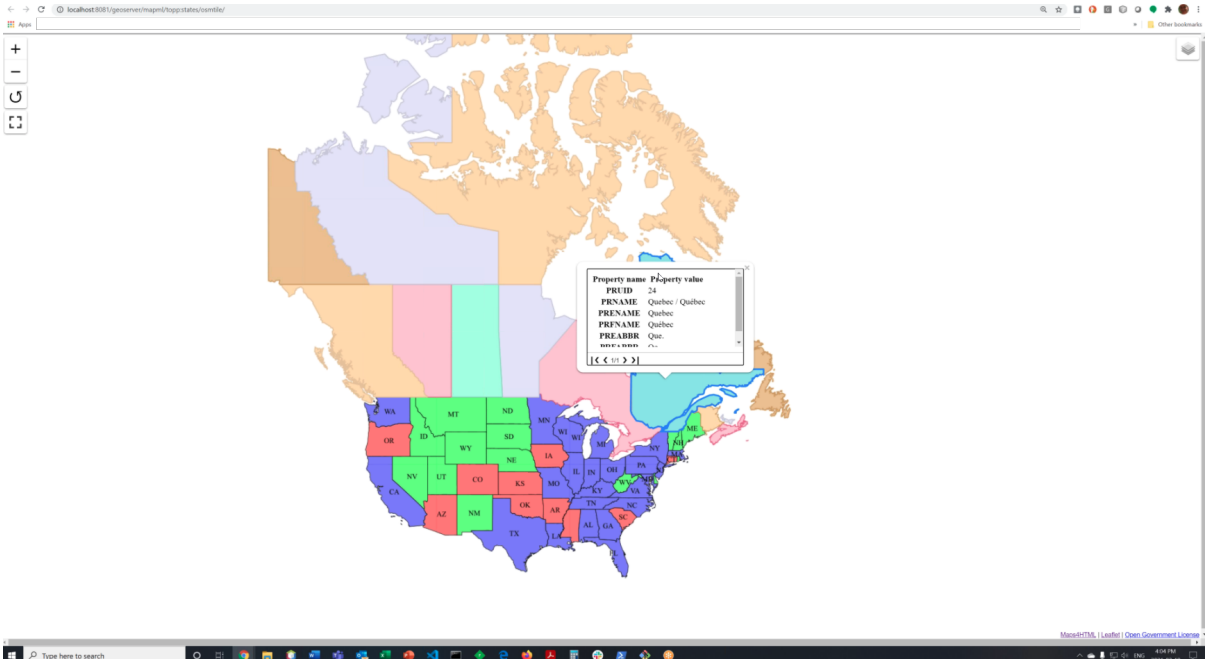


Figure 11. A screenshot of the MapML viewer in GeoServer

Figure 12 shows an additional screenshot of the MapML viewer, with a grid placed above the map layers.

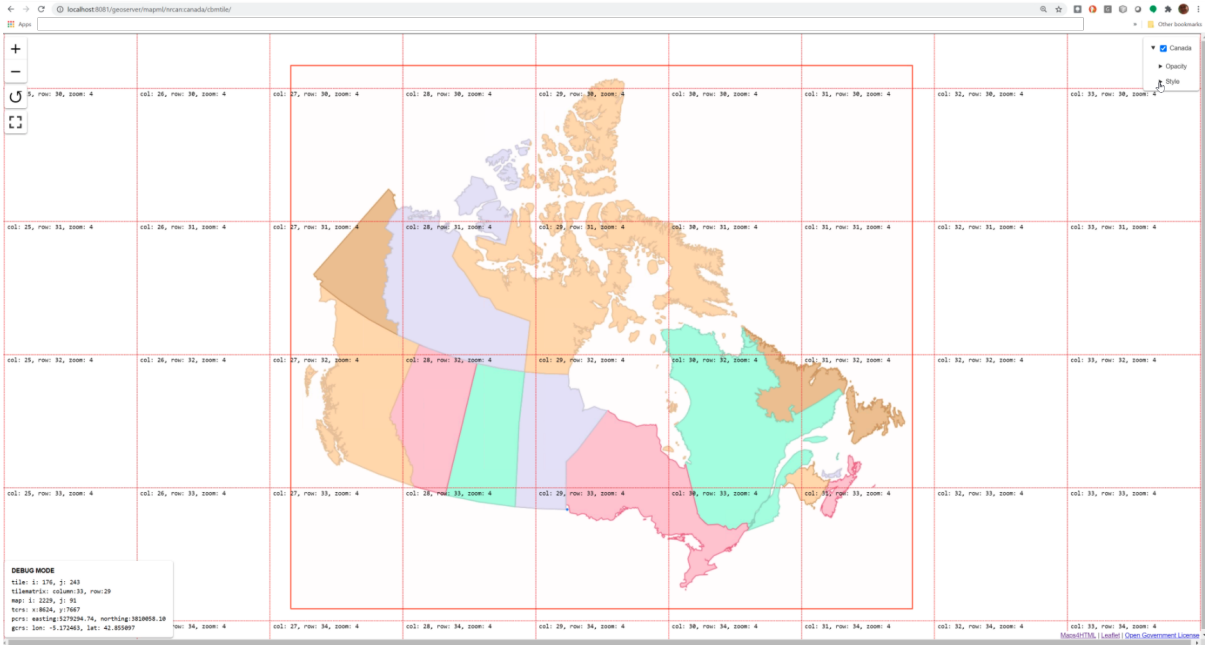


Figure 12. A gridded screenshot of the MapML viewer built for GeoServer

Figure 13 shows a screenshot of the landing page of an ldproxy instance that publishes data from the National Mapping Agency of the Federal Republic of Germany. The screenshot demonstrates the content negotiation capabilities supported by OGC APIs that enable a client application such as a web browser to request a resource in HTML and a different client application such as a developer utility (e.g. postman [https://www.postman.com]) to request the same resource in JSON.

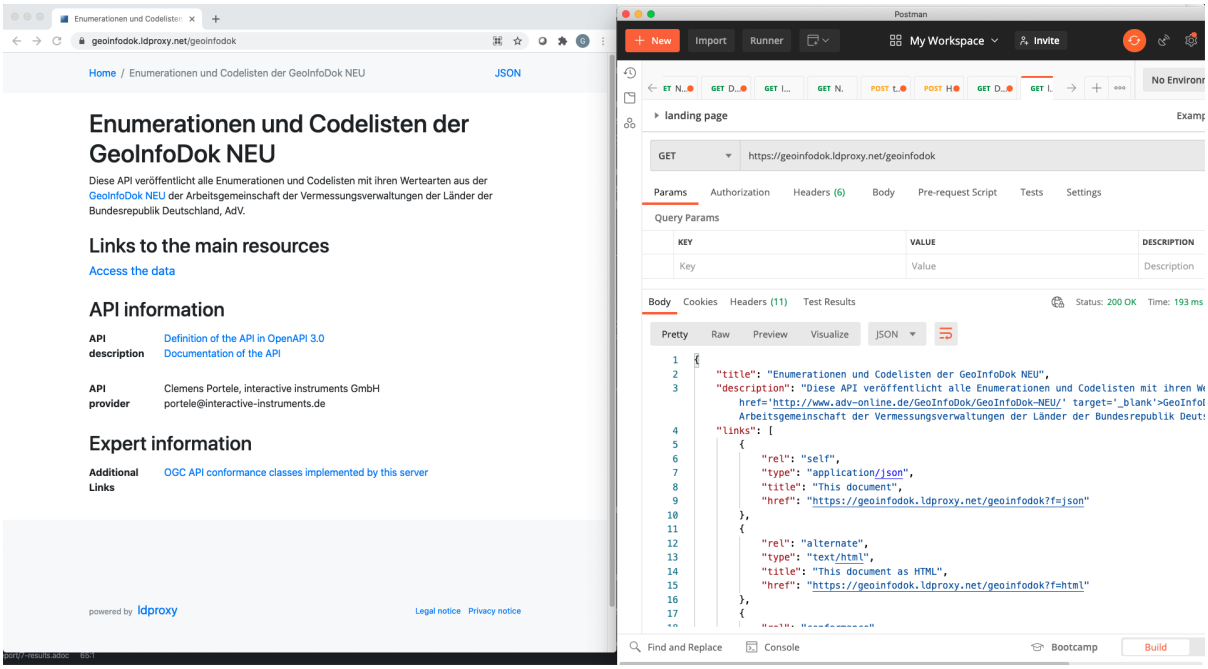


Figure 13. A screenshot of the landing page of an ldproxy instance accessed using a web browser (left) and postman (right)

Figure 14 shows a screenshot of the CubeWerx Ship Detection processes running on Sentinel data in the Amazon Web Services Cloud. Available input datasets are listed on the left-hand side of the figure, whereas in-progress and completed jobs are listed on the right-hand side of the figure.

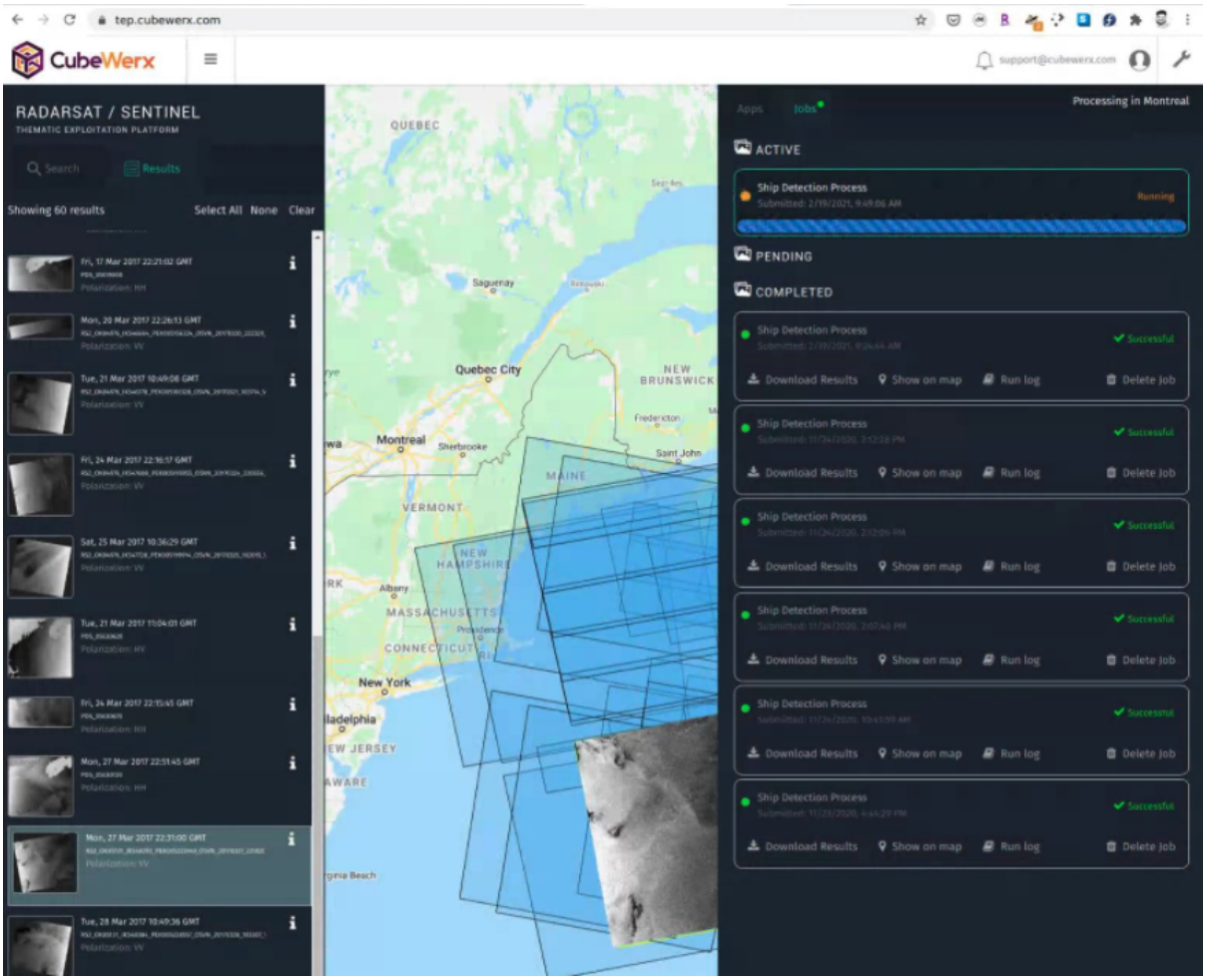


Figure 14. A screenshot of the CubeWerx processes management tool

Figure 15 presents example output from the CubeWerx Ship Detection processes. The positions of detected ships are shown by the red markers.

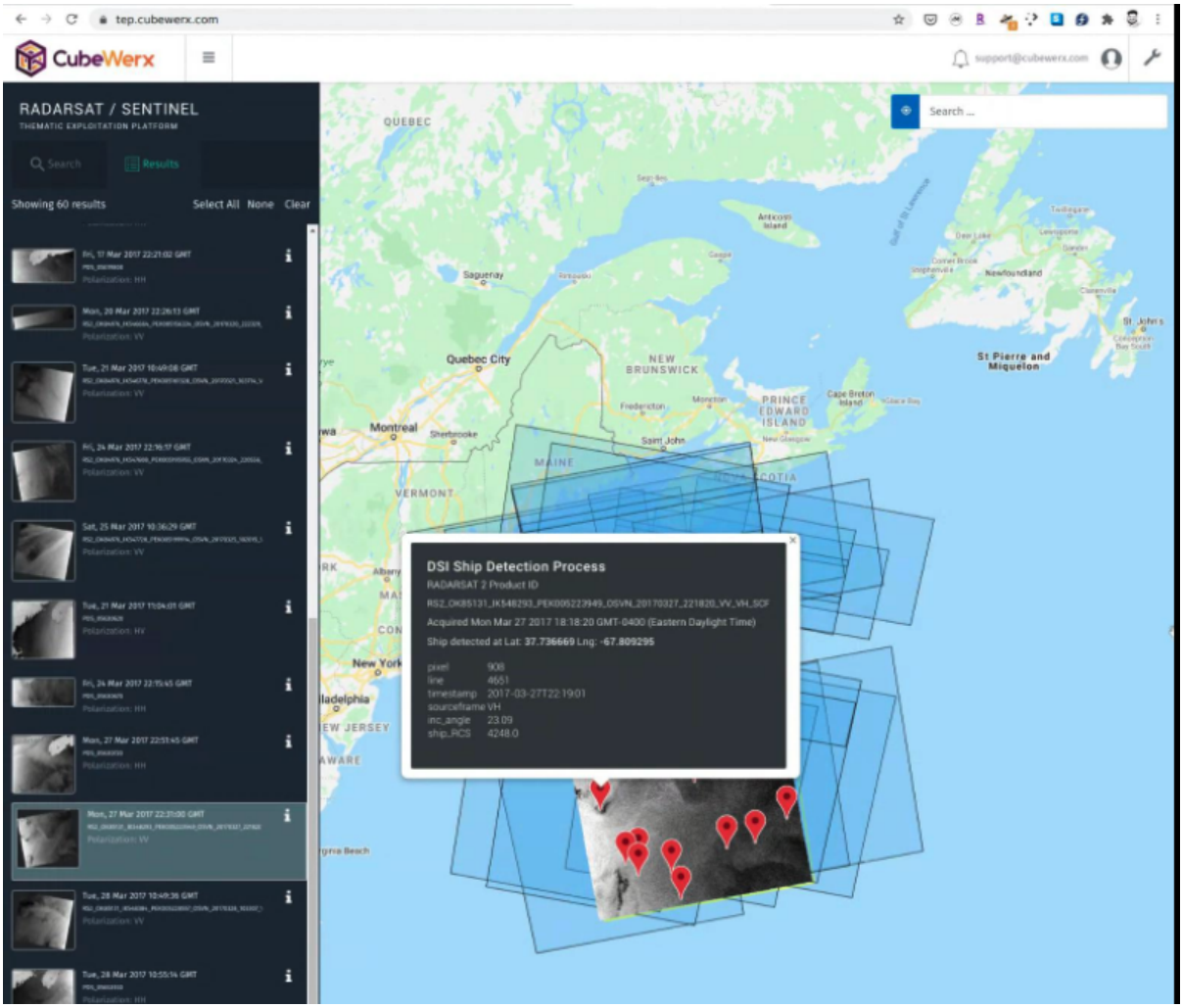


Figure 15. Example output from the CubeWerx Ship Detection processes

Figure 16 shows a screenshot of an xarray supported pygeoapi displaying a coverage. The coverage is accessed through an OGC API - Coverages interface and has been styled for portrayal purposes. The demonstration showed how OGC API - Tiles could be implemented alongside OGC API - Coverages to enable access to tiled coverage data.

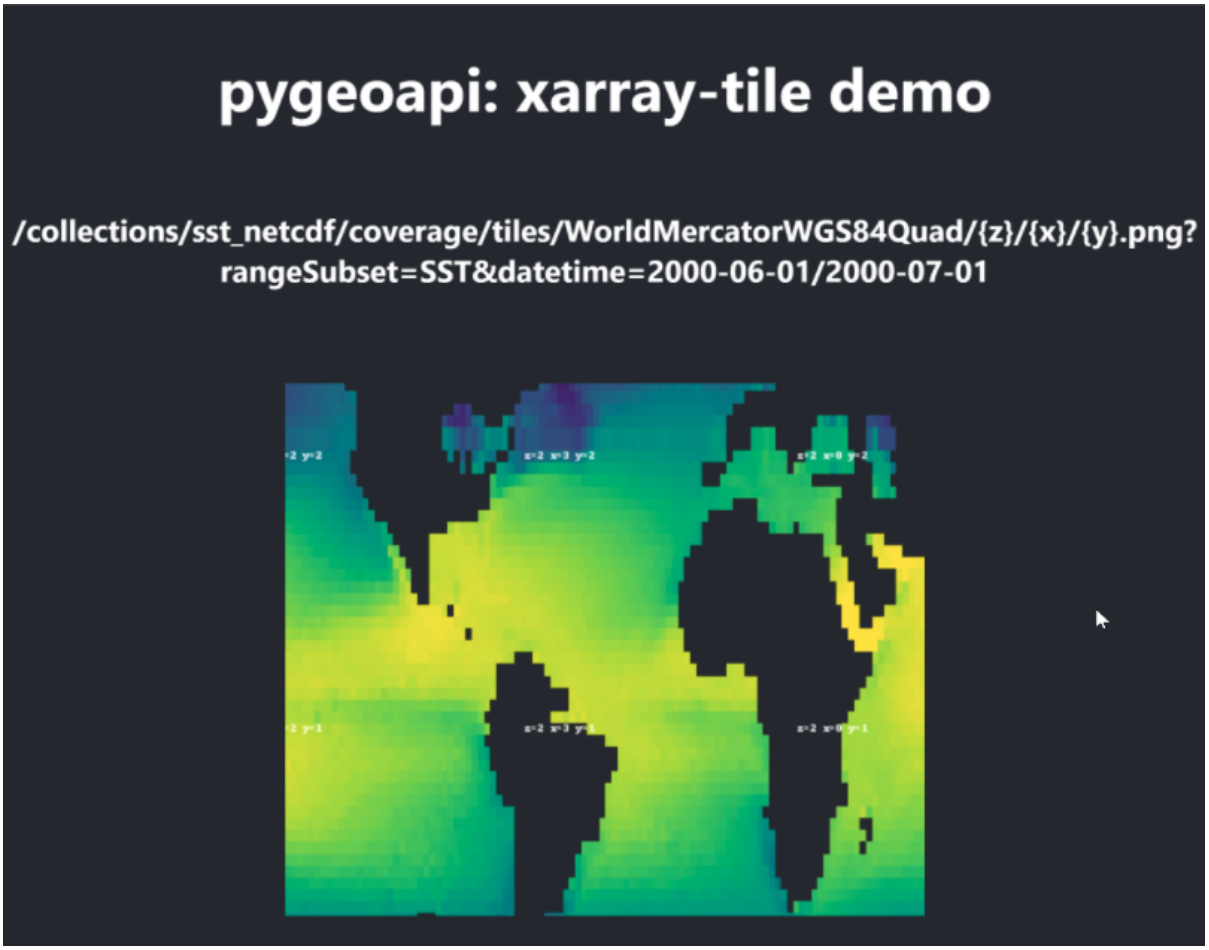


Figure 16. A screenshot of an x-array supported pygeoapi displaying a coverage

Chapter 9. Discussion

9.1. Cross cutting topics

9.1.1. Landing Pages

Amongst the cross cutting issues to be discussed during the code sprint was the issue of discovery of landing pages. Currently OGC APIs offer a landing page concept which serves as the entrypoint for discovery of other resources offered by an API. The sprint participants highlighted the potential benefit of having a "landing page of landing pages" that could enable a client application to discover other landing pages. Participants also observed that an OGC API - Records implementation could provide the discovery functionality needed to discover landing pages of other OGC APIs.

The participants observed the discovery of landing pages to be particularly useful in a microservice-based architecture. The microservices architecture deploys application code into small and manageable containers that can run independently of others. Whereas the microservices approach offers some benefits such as isolation of many types of faults, it can also introduce challenges in memory consumption as each microservice runs in an independent container.

9.1.2. Code generation from OpenAPI definition documents

The sprint participants also discussed the ability of code generation tools to generate code that adequately describes the APIs defined using OpenAPI. Several participants commented that code generation tools seldom describe the business logic required to handle geospatial operations. Therefore, developers often have to manually write code into the stubs created by code generation tools. Further, the participants noted that for APIs that enable the publishing of dynamic resources at runtime, code generation tools often require additional manually written code to enable support of those dynamic resources. In this context, dynamic resource include feature collections that might chance their schema at some point in time. The participants therefore recommended that implementers of code generating tools should make enhancements to address some of those shortcomings, for example support for dynamic resources and enabling semantic awareness to provide context to API components.

9.2. Group specific discussions

9.2.1. OGC API - Records

There was discussion [https://github.com/opengeospatial/joint-ogc-osgeo-asf-sprint-2021/issues/50] on how to represent the language for the record (possibly using an HTTP resource; the language that was requested). There was discussion on what happens if a client requests all the languages available at the same time. There is the potential to end up with verbose arrays. The participants suggested that a better approach could be to leverage hypermedia.

CubeWerx updated their server during the code sprint to follow the draft OGC API - Records specification. The team made progress [https://www.pvretano.com/cubewerx/cubeserv/default/ogcapi/catalogues/collections/sentinel1cat/items] on their implementation of support for OpenSearch. There was also discussion on potential integration with ActiveMQ to support implementation of a GSS.

The pygeoapi team advanced their implementation of OGC API - Records. There were some updates to the Records API schema. They also implemented itemType property to enable identification of the types of items returned by an API. The pygeoapi team also discussed with the GeoNode team about the possibility of using pygeoapi as a backend to GeoNode.

The pycsw team discussed how they are going to implement OGC API - Records, while still supporting versions 2 and 3 of the OGC Catalogue Services for the Web (CSW) standard. CSW is the predecessor of OGC API - Records. CSW supports the ability to publish and search collections of descriptive information (metadata) for data, services, and related information objects.

interactive instruments updated ldproxy during the code sprint to support the Core, HTML, JSON and CQL Filter conformance classes of the current draft of OGC API - Records (pull request [https://github.com/interactive-instruments/ldproxy/pull/370:]). A sample instance [https://geoinfodok.ldproxy.net/geoinfodok] has been set up during the sprint and added as an implementation in the GitHub repository of OGC API - Records [https://github.com/opengeospatial/ogcapi-records/blob/master/implementations.md#ldproxy].

9.2.2. OGC API - Processes

Participants experimented with the implementation of OGC API - Processes through use of the Spring Framework. The Spring Framework is an application framework for the Java platform. Spring has been particularly popular amongst Java developers for enabling the implementation of RESTful APIs and microservices. Process description and process execution were successfully implemented. Some of the considerations include workflows, transactions and security.

9.2.3. GeoAPI

There was a discussion about supporting the library of UCAR Unidata. The participants sought to make UCAR's library an optional item that could be used alongside implementations of the GeoAPI. The UCAR library supports netCDF - a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. A GitHub repository [https://github.com/Unidata/geoapi-netcdf-java] was created to support this initiative.

There was also discussion about the possibility of advancing the Python port of the GeoAPI library. Work has already started on the Python port. Other languages were also consider, for example NodeJS. Volunteers were invited to lead GeoAPI efforts on other programming languages.

9.2.4. MapML

The MapML team discussed with the GeoServer team about the progress for upgrading from a community module to core. There was some work done on feature queries and navigation. The participants encountered some issues relating to Java 8 and Maven, however the issues were successfully resolved. The team planned to continue server improvement.

9.2.5. OGC API - Maps and OGC API - Tiles

Discussed how to transition from one approach of serving tiles to another way. The challenge is to combine a collection with a style. One suggestion is that haying the map before the style on the URL may improve access. The GeoServer team expressed their preference for that sequence (i.e. map/style/).

9.2.6. pygeoapi and OWSLib

There was discussion about enabling pygeoapi to support x-array databases. An xarray database supports the labelling of data to indicate dimensions, coordinates and other attributes.

On the OWSLib side, there was discussion about doing an OGC API client. One of the options considered for adding to OWSLib was pydantic, a python-based framework for data validation. There was also discussion about doing OWS documentation in jupyter using Sphinx.

9.2.7. pgRouting

There was discussion about the potential to implement an OGC API - Routes interface on top of pgRouting. The participants acknowledged that the scope of pgRouting does not currently APIs. However, the participants also acknowledged the potential benefit of ensuring that information provided by an OGC API - Routes implementation could be fully ingested by pgRouting and vice versa. Another OGC standard that was identified as potentially relevant to pgRouting is the OGC IndoorGML standard.

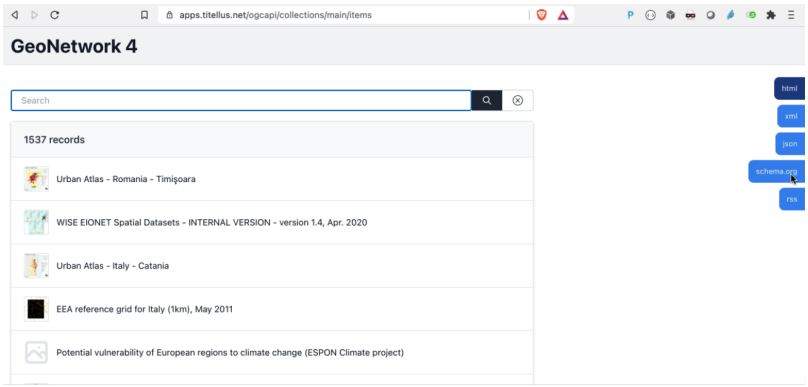
9.2.8. OSGeo GDAL

There was an announcement that an experimental version of a GDAL driver [https://gdal.org/drivers/raster/ogcapi.html] that supports OGC API - Tiles and OGC API - Coverages has recently been implemented. The GDAL driver has been developed to demonstrate work related to the “Modular OGC API Workflows” initiative that is led by Ecere and supported by Natural Resources Canada. Much of the discussion regarding the GDAL driver focused on improving the documentation.

9.2.9. OSGeo GeoNetwork

The GeoNetwork team worked on a refactor of the code, related to OGC API - Records. In previous iterations the YAML files provided by OGC were used to build a Java skeleton. With the refactor, the openapi document is generated from Java code using the SpringFox [https://springfox.github.io/springfox/] library. This allows the developer to add some properties and methods which are not

(yet) in the standard.



For example GeoNetwork provides DCAT, schema.org, Atom outputs of Metadata records in various encodings, such as HTML, JSON, (ISO19139) XML, JSON-LD, RDF/XML and Turtle.

A list of discussions from the sprint is presented below:

- Add the thumbnail/graphicoverview fields to the core record model https://github.com/opengeospatial/ogcapi-records/issues/94
- Start an initiative to create an extension for Facet Statistics https://github.com/opengeospatial/ogcapi-records/issues/52
- Multilingual aspects of metadata https://github.com/opengeospatial/ogcapi-records/issues/91
- Since GeoNetwork does support Atom/Opensearch, verification is needed to confirm whether all requirements of conformance classes are met, and if so, this should be listed on the conformance endpoint
- It was previously not clear that GeoJSON will be a default (json) response format for record queries. This needs to be made clearer.
- The participants tested the json-ld output of one of the OGC API - Records implementations with the Jena ARQ SPARQL client. It worked for a single document, but queries did not continue into related records. Mostly because the json-ld context, at the time, did not list relations to the collection or siblings. In theory it should be possible to find the `dcat:catalogue` from a `dcat:catalogueRecord`, and from the catalogue, other records.

9.2.10. OSGeo GeoServer

openlayers does not have support for OGC API - Maps. So the GeoServer team observed that

OGC APIs expect parameters to be case sensitive. In some cases however, for example with OpenLayers, the parameters can be of different case.

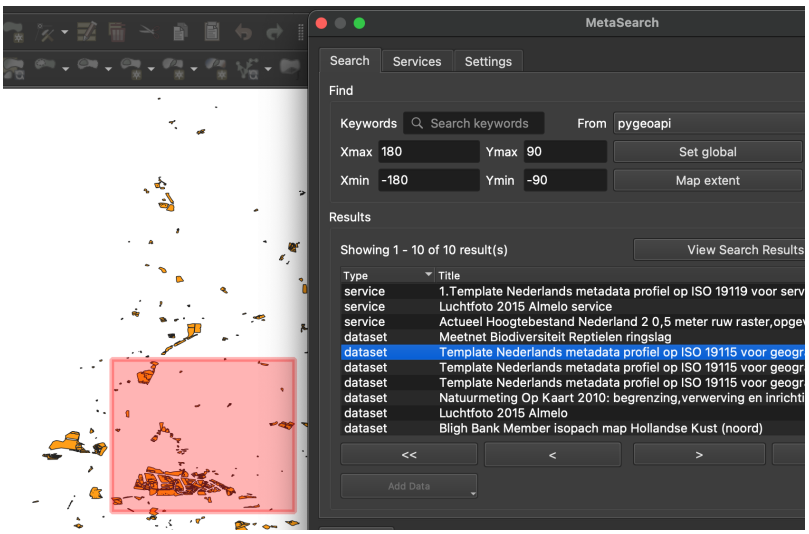
The team observed that there is a need to provide a way of displaying layerGroups. At the moment they are displayed as collections.

9.2.11. OSGeo GRASS GIS

The participants created [https://github.com/OSGeo/grass-addons/pull/439] the alpha version of GRASS GIS import modules for the OGC API - Features Standard and the draft OGC API - Coverages specification.

9.2.12. OSGeo QGIS

QGIS team worked on an OGC API Records client. A pull request has been prepared at https://github.com/qgis/QGIS/pull/41713



Relevant to this development is the discussion at https://github.com/opengeospatial/ogcapi-records/issues/95. QGIS facilitates to add the service layer to the map, when it detects the type of the service. Current CSW implementation has a long algorythm of testing various metadata properties, the service url and probing the endpoint.

The Apache group was interested to use QGIS as a tool for data modification and then loading it into Fuseki. They were looking for tooling that would export the data as RDF from QGIS. They found a solution using GeoServer as middleware, which has a community plugin at https://docs.geoserver.org/latest/en/user/community/json-ld/output.html, which offers json-ld capabilities.

QGIS has OGC API Features support.

9.2.13. Apache ActiveMQ

The participants observed that ActiveMQ is relevant to the concepts discussed in the OGC Publish/Subscribe (PubSub) 1.0 standard and the draft PubSub Whitepaper. OGC PubSub 1.0 is an interface specification that supports the core components and concepts of the Publish/Subscribe message exchange pattern with OGC Web Services. Another observation made was that ActiveMQ is potentially related to the draft GeoSynchronization Service (GSS) specification. The draft GSS specification describes a service that allows data collectors to propose changes to be made to a data provider's features.

9.2.14. Apache Jena

The Apache Jena team discussed an approach with the GeoServer team for enabling the export of a feature collection from QGIS to GeoServer, and then to Apache Jena Fuseki [https://jena.apache.org/documentation/fuseki2/]. Fuseki offers a SPARQL Server interface on top of Apache Jena. The groups concluded that it would be possible to export create a plug-in for QGIS that allows it to post GeoJSON to GeoServer. An additional plug-in could then be implemented to export GeoJSON-LD [https://geojson.org/geojson-ld/] from GeoServer to Fuseki.

The GeoJSON-LD document would be interpreted as RDF [https://www.w3.org/TR/rdf11-concepts/] triples by Fuseki. There was also discussion about how to represent the schema of the feature collection. SHACL [https://www.w3.org/TR/shacl/] was identified as a possible candidate for representing the schema of the feature collection. Jena supports SHACL.

9.3. Lessons Learnt

- There is need to improve discovery of resources (e.g. landing pages) across multiple deployments. The binding aspects also need to be considered, for example potentially using actionable links. We need some form of harmonization on link relation types.
- When the SWGs are defining the APIs they should define the default behavior. OWS needed several parameters to retrieve a resource, however for OGC APIs they should be designed to allow default behavior when params are not present.
- The participants therefore recommended that implementers of code generating tools should make enhancements to address some of those shortcomings, for example support for dynamic resources and enabling semantic awareness to provide context to API components. There is also a fundamental limit in the specifications. This could potentially be addressed through support for templatedRef https://github.com/OAI/OpenAPI-Specification/issues/2453
- There is a need to enable paging of collections. For example, it is possible that some implementations of OGC API - Processes could have hundreds of processes. This needs to be addressed in the OGC API - Common - Part 2: Geospatial Data extension. This is also related to Principle 11 of the Guidelines https://github.com/opengeospatial/OGC-Web-API-Guidelines
- There may also need to be a mechanism to group collections (i.e. a collection of collections). https://github.com/opengeospatial/oapi_common/issues/11

9.3.1. OSGeo and OGC

As part of ongoing discussions regarding renewing the OSGeo/OGC Memorandum of Understanding (MOU), there was discussion between Scott Simmons (OGC Chief Standards Officer) and members of the OSGeo Board of Directors on assessing the current MOU as well as current state of affairs. In addition, there was discussion on opportunities to strengthen the MOU given the OGC's increasing focus on developers and the importance of free and open source software.

The draft MOU continues to be updated and will be tabled for review by both organizations.

Chapter 10. Conclusions

The code sprint facilitated the development and testing of prototype implementations of OGC standards, including implementations of draft OGC APIs 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.

10.1. Future work

The following general recommendations for future innovation work items were made during the sprint:

- Themes, trees, nesting, and other strategies need to be explored. This is needed because data providers often have thousands of datasets that they need to manage or publish.
- There is a need for more experimentation on styles and coverages. Including how styles can be used to render/filter coverages.
- Tiled coverages and their support through OGC API - Coverages and OGC API - Tiles integration should be explored further.
- More experimentation is needed on workflows in relation to the OGC API - Processes - Part 3 : Workflows.
- Further development of the DGGS API, including work on client applications and visualization.
- There is a need to advance OGC APIs and other OGC standards to enable the cataloguing and discovery of models (e.g. AI/Machine Learning models). We need to explore how to sufficiently describe the models.
- The implications of OpenAPI 3.1, JSON Schema and Webhooks need to be examined and path towards transition identified.
- Some integration of the MapML format concept with the OGC API offerings, for example into the HTML representation of features, to enhance the spatial indexing of HTML.
- Enhancement of the Link Relations Register.

Standards Working Groups should consider introducing support for the following:

- Associations between records and other elements in catalogues
- Landing page of landing pages
- Searchable collections in OGC APIs (including the collections of collections)

- Where appropriate, clarification that GeoJSON is the default JSON encoding for OGC API - Features and OGC API - Records

Appendix A: Organization

A.1. Schedule / Agenda

All times on the agenda are in UTC-5/EST (New York time).

 means that the session will be in the main Gotomeeting virtual room.

Time (EST)	What	Lead
Webinar (GotoMeeting) 12th February 2021 at 10:00 See World Clock [https://www.timeanddate.com/worldclock/meetingtime.html?iso=20210212&p1=224&p2=179&p3=16&p4=44&p5=240&p7=136]		
5 minutes	Welcome Remarks	Gobe Hobona (OGC), Angelos Tzotsos (OSGeo), Tom Kralidis (OSGeo) and Martin Desruisseaux (ASF)
5 minutes	Sponsor Remarks	Sponsor representative
10 minutes	Overview of participating OGC standards working groups	Gobe Hobona (OGC)
10 minutes	Overview of participating OSGeo projects	Angelos Tzotsos , Tom Kralidis (OSGeo)
10 minutes	Overview of participating ASF projects	Martin Desruisseaux (ASF)
20 minutes	Questions & Answers	
17th February 2021 See World Clock [https://www.timeanddate.com/worldclock/meetingtime.html?iso=20210217&p1=224&p2=179&p3=16&p4=44&p5=240&p7=136]		
02:00-03:00	Networking	A chance to network and meet other participants before the day starts.
03:00-07:00	Break	
07:00-07:05	Welcome & objectives	Gobe Hobona (OGC), Angelos Tzotsos (OSGeo), Tom Kralidis (OSGeo) and Martin Desruisseaux (ASF)
07:05-07:10	Sprint programme & way of working	
07:10-07:20	Sprint goals for OGC standards working groups	Gobe Hobona (OGC)

Time (EST)	What	Lead
07:20-07:30	Sprint goals for OSGeo projects	Angelos Tzotsos , Tom Kralidis (OSGeo)
07:30-07:40	Sprint goals for ASF projects	Martin Desruisseaux (ASF)
07:40-08:00	Questions and Answers	
08:00-09:00	5-minute pitch by each project or working group	
09:00-13:00	Practical work in break-out rooms	
13:00-13:30	Break	
13:30-16:30	Practical work in break-out rooms	
16:30-17:30	Daily brief back	
18th February 2021 See World Clock [https://www.timeanddate.com/worldclock/meetingtime.html?iso=20210218&p1=224&p2=179&p3=16&p4=44&p5=240&p7=136]		
02:00-03:00	Networking	A chance to network and meet other participants before the day starts.
03:00-07:00	Break	
07:00-09:00	Practical work in break-out rooms	
09:00-10:00	a short stand-up and preliminary demonstration	
10:00-12:30	Practical work in break-out rooms	
12:30-13:00	Issues / concerns	
13:00-13:30	Break	
13:30-16:30	Practical work in break-out rooms	
16:30-17:30	Daily brief back	
19th February 2021 See World Clock [https://www.timeanddate.com/worldclock/meetingtime.html?iso=20210219&p1=224&p2=179&p3=16&p4=44&p5=240&p7=136]		
02:00-03:00	Networking	A chance to network and meet other participants before the day starts.
03:00-07:00	Break	
07:00-09:00	Practical work in break-out rooms	
09:00-10:00	a short stand-up and preliminary demonstration	
10:00-12:30	Practical work in break-out rooms	

Time (EST)	What	Lead
12:30-13:00	Issues / concerns	
13:00-13:30	Break	
13:30-15:30	Practical work in break-out rooms	
15:30-16:30	Demonstration	
16:30-17:30	Wrap-up: immediate lessons, next steps, thanks and goodbyes	Gobe Hobona (OGC), Angelos Tzotsos (OSGeo), Tom Kralidis (OSGeo) and Martin Desruisseaux (ASF)

Appendix B: Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
2021-02-18	G. Hobona	.1	all	initial version

Table 1. Revision History

Appendix C: Bibliography

[1] Simonis, I.: OGC Innovation Program: Policies and Procedures. OGC 05-127r10,Open Geospatial Consortium, https://portal.ogc.org/files/?artifact_id=92756 (2020).

[2] Hobona, G., Jackson, M., Anand, S.: Implementing Geospatial Web Services for Cloud Computing. In: Zhao, P. and Di, L. (eds.) Geospatial Web Services: Advances in Information Interoperability. pp. 287–308. IGI Global (2011).

[3] Bondaruk, B., Roberts, S.A., Robertson, C.: Discrete Global Grid Systems: Operational Capability of the Current State of the Art. In: Fast, V., McKenzieand, G., and Sieber, R. (eds.) Proceedings of the 2019 Spatial Knowledge and Information Canada. CEUR (2019).