July 2021 OGC API Code Sprint Summary Engineering Report

Publication Date: YYYY-MM-DD

Approval Date: YYYY-MM-DD

Submission Date: YYYY-MM-DD

Reference number of this document: OGC NN-NNN

Reference URL for this document: http://www.opengis.net/doc/PER/202107APISprintER-ID

Category: OGC Public Engineering Report

Editor: Gobe Hobona, Joana Simoes

Title: July 2021 OGC API Code Sprint 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 virtual code sprint that was held from July 21st to July 23rd , 2021 to advance the development of the OGC API - Processes draft standard, OGC API - Records draft standard, and the OGC API – Coverages draft standard. 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).

Chapter 2. Executive Summary

This Engineering Report (ER) summarizes the main achievements of the July 2021 OGC API Virtual Code Sprint, conducted from July 21st to July 23rd. The goal of the code sprint was to progress the development of the draft OGC API standards for Records, Coverages and Processes. The sprint served to accelerate development of candidate standards of the OGC APIs.

Specifically, the objectives of the code sprint were to:

- Develop prototype implementations of OGC API – Processes
- Develop prototype implementations of OGC API – Records
- Develop prototype implementations of OGC API – Coverages
- Test the prototype implementations
- Provide feedback to the Editor about what worked and what did not work
- Provide feedback about the specification document, especially what is missing from the document

Part of the motivation for holding the sprint was:

- APIs are a popular, effective method for rapid software development
- There is an increasing need for interoperability between Web APIs
- The growing uptake of location within and outside of geospatial developer communities

The draft OGC API - Processes specification defines an interface that enables the execution of geospatial computing processes and the retrieval of metadata describing their purpose and functionality. Typically, these processes execute well-defined algorithms that ingest vector and/or coverage data to produce new datasets.

The draft OGC API - Records specification defines an interface that enables 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 and accessed.

The draft OGC API - Coverages specification defines an interface that enables access to coverages that are modeled according to the Coverage Implementation Schema (CIS) 1.1. 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 code sprint facilitated the development and testing of prototype implementations of the aforementioned OGC API candidate standards. The code sprint therefore successfully met all of its objectives and achieved its goal of progressing the development of the standards.

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	OGC	Editor
Joana Simoes	OGC	Editor
Full Name	Organisation Name	Contributor
Full Name	Organisation Name	Contributor
Full Name	Organisation Name	Contributor
Full Name	Organisation Name	Contributor
Full Name	Organisation Name	Contributor
Full Name	Organisation Name	Contributor
Full Name	Organisation Name	Contributor
Full Name	Organisation Name	Contributor
Full Name	Organisation Name	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 18-062, OGC API - Processes - Part 1: Core candidate standard [http://docs.ogc.org/DRAFTS/18-062.html]
- OGC: OGC 20-004, OGC API - Records - Part 1: Core candidate standard [http://docs.ogc.org/DRAFTS/20-004.html]
- OGC: OGC 19-087, OGC API - Coverages - Part 1: Core candidate standard [http://docs.ogc.org/DRAFTS/19-087.html]
- IETF: RFC-7946 The GeoJSON Format (2016)

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.

- *coordinate reference system*

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

- *coverage*

  feature that acts as a function to return values from its range for any direct position within its spatiotemporal domain, as defined in OGC Abstract Topic 6

- *record*

  atomic unit of information of a catalogue that is used to provide information about a particular resource

- *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]

  NOTE | See Best Practice 24: Use Web Standards as the foundation of APIs (W3C Data on the Web Best Practices) for more detail.

4.1. Abbreviated terms

- **API** Application Programming Interface
- **CORS** Cross-Origin Resource Sharing
- **OGC** Open Geospatial Consortium

Chapter 5. Introduction

This Engineering Report (ER) summarizes the main achievements of the July 2021 OGC API Virtual Code Sprint, conducted from July 21st to July 23rd. The sprint had been organized to advance the development of draft standards of OGC APIs for Records, Coverages and Processes. Sprint participants prototyped implementations of the draft standards, validating the requirements and providing feedback so that the draft standards could be improved.

An OGC 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. OGC 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 as a proof of concept for other OGC Innovation Program initiatives, or to support OGC Standards Program activities.

Chapter 6. High-Level Architecture

6.1. High Level Overview

The focus of the sprint was on support of the development of the draft OGC API - Records [https://ogcapi.ogc.org/records], OGC API - Processes [https://ogcapi.ogc.org/processes] and OGC API - Coverages [https://ogcapi.ogc.org/coverages] standards. 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.

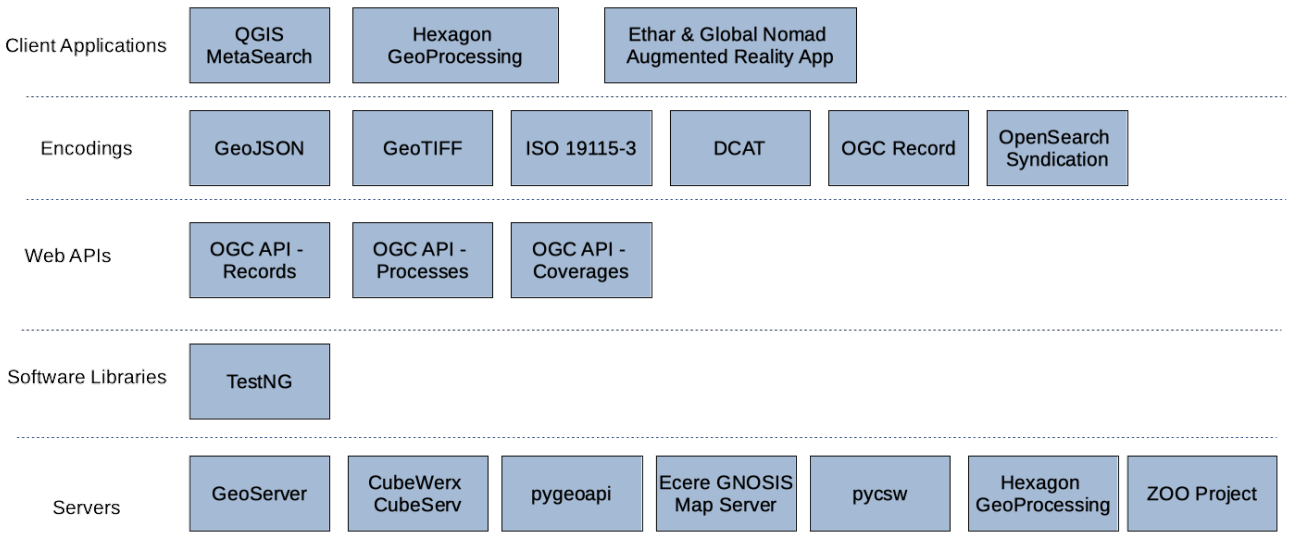

Figure 1. High level overview of the architecture implemented during the sprint

As illustrated, the sprint architecture was designed with the view of enabling client applications to connect to different servers that implement OGC APIs. The servers were provisioned with metadata, implementations of algorithms for geospatial analysis, vector feature data, and coverages such as satellite imagery.

6.2. Candidate Standards

6.2.1. OGC API - Processes

The draft OGC API - Processes specification defines an interface that enables the execution of geospatial computing processes and the retrieval of metadata describing their purpose and functionality. Typically, these processes execute implementations of well-defined algorithms that ingest vector and/or coverage data to produce new datasets or analytical products.

6.2.2. OGC API - Records

The draft OGC API - Records specification defines an interface that enables 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 and accessed.

6.2.3. OGC API - Coverages

The draft OGC API - Coverages specification defines an interface that enables access to coverages that are modeled according to the Coverage Implementation Schema (CIS) 1.1. 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.

Chapter 7. Results

Multiple organizations provided servers, API implementations, and capabilities during the event. The rest of this section describes each of the implementations.

7.1. Augmented Reality Application

Developers from Ethar Inc and Global Nomad took part in the code sprint, extending an Augmented Reality (AR) application that they are building for Testbed-17. The AR application, which runs on a smartphone, allows a coverage such as one representing a Digital Elevevation Model (DEM), to be visualized in a three-dimensional AR view as shown in Figure 2. This facilitates 'tabletop' visualization so that a group of people could potentially stand around a table and see geographic representation superimposed onto that table, with the DEM viewed either through smart glasses or a smartphone. The AR application can ingest coverages encoded in GeoTIFF format, therefore the application could be served by an implementation of OGC API - Coverages.

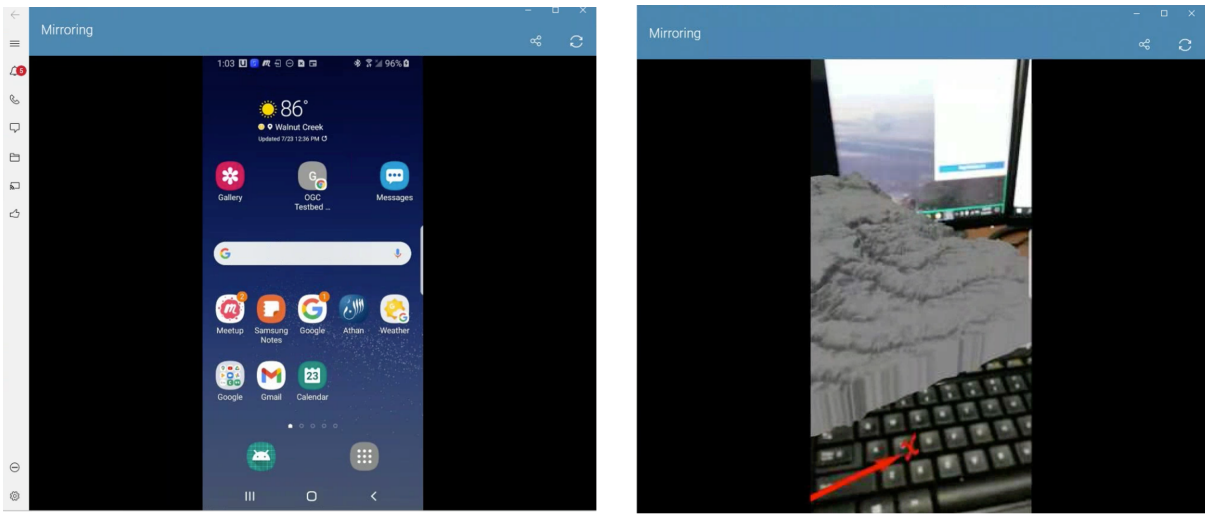

Figure 2. Screenshots of a smartphone mirrored (a) at standby (b) with the augmented reality application running

7.2. CubeWerx CubeSERV

The CubeWerx server ("CubeSERV") 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. During this code sprint, CubeWerx helped fine-tune and mature several aspects of the OGC API specifications in focus for the sprint, and refined CubeSERV accordingly. CubeSERV supports multiple OGC API specifications.

The CubeSERV implementation of OGC API - Processes was configured to demonstrate a Ship

Detection capability, identifying ships from RADARSAT imagery. This was achieved by executing the analytical process on multiple nodes running in different Public Cloud regions (i.e. one in North America and another in Europe). Figure 3 shows a view of the different nodes, running in different Cloud regions and used to detect the ships. The different nodes communicated through interfaces that conform to OGC API - Processes.

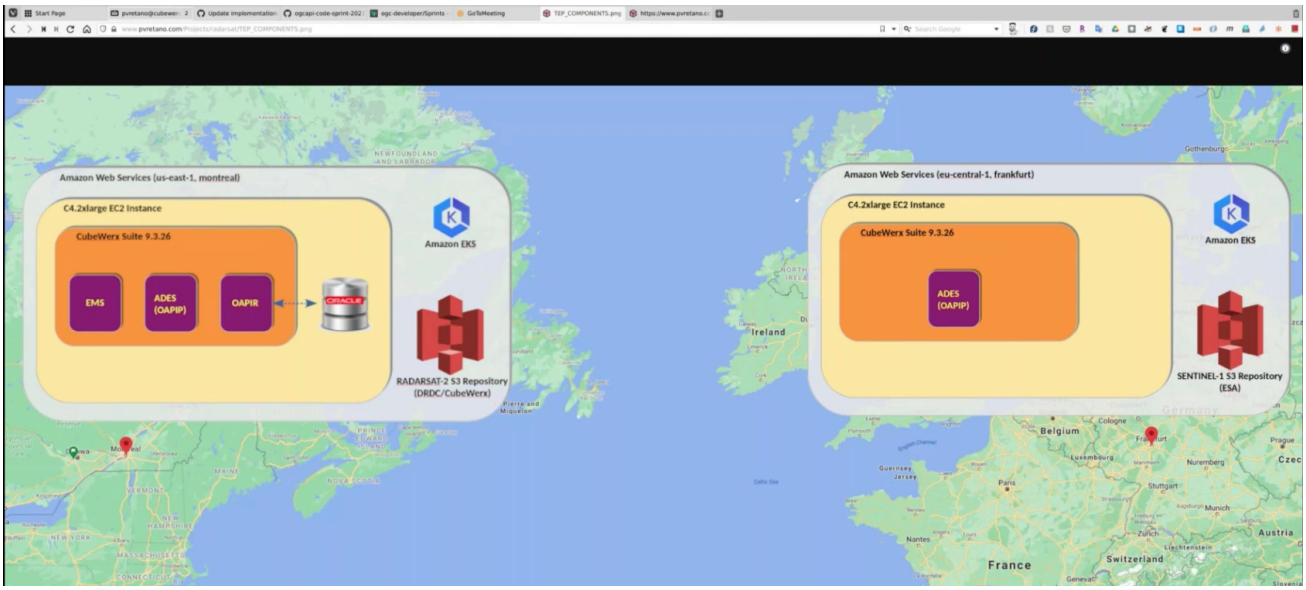

Figure 3. Screenshot from a demonstration of CubeSERV

Once a processing job running in CubeSERV has completed the outputs are made available for display on a web portal. In the case of Ship Detection, the location of ships was displayed as shown in Figure 4.

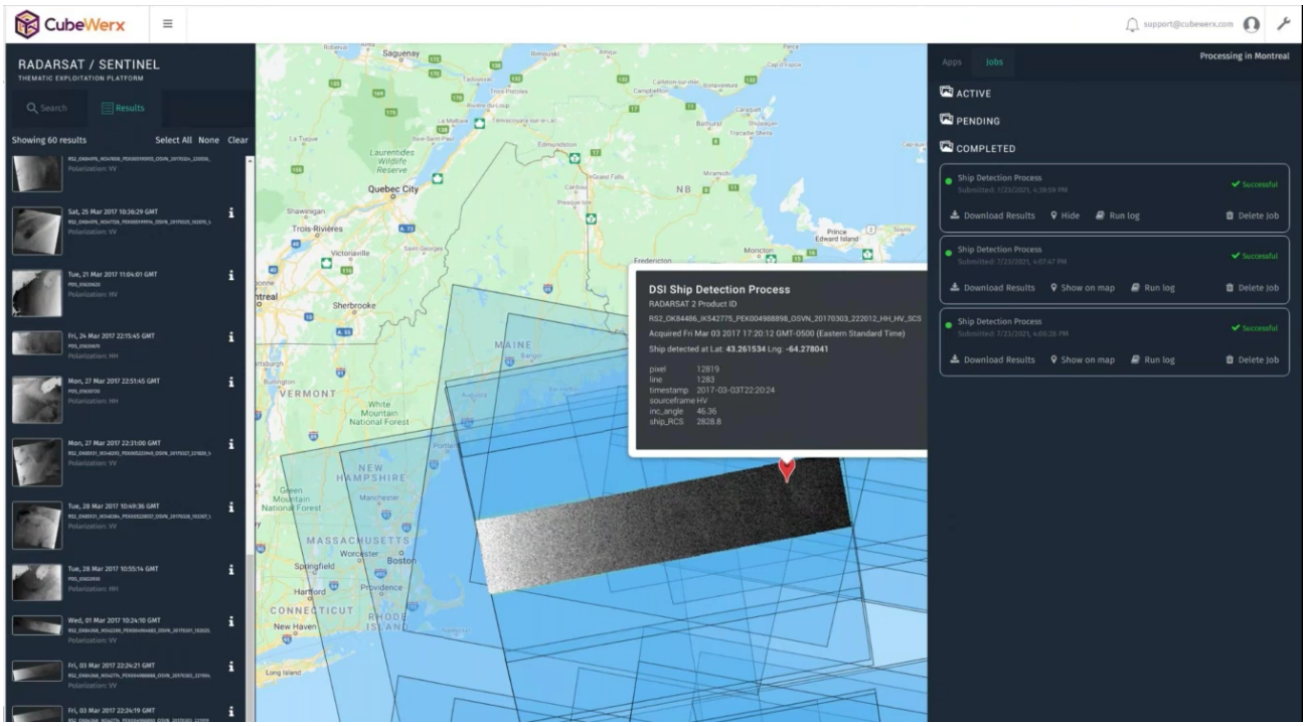

Figure 4. A second screenshot from a demonstration of CubeSERV

The CubeSERV implementation of OGC API - Records was configured to support discussion and validation of the specification, whereas the OGC API - Coverages implementation facilitated

development of compliance tests during the code sprint.

7.3. ZOO Project

Developers from Geolabs deployed an instance of ZOO Project, an open source platform written in C that implements a number of interface standards for geospatial processing, including OGC API - Processes and multiple versions of the WPS standard. The ZOO Project instance deployed for the code sprint offered wrappers around libraries such as the Orfeo Toolbox (OTB) and software such as GRASS GIS and SAGA GIS. This meant that through the OGC API - Processes interface, a client application could invoke the processing and analytical functions offered by these software packages. A screenshot of a landing page from a Geolabs ZOO Project instance is shown in Figure 5.

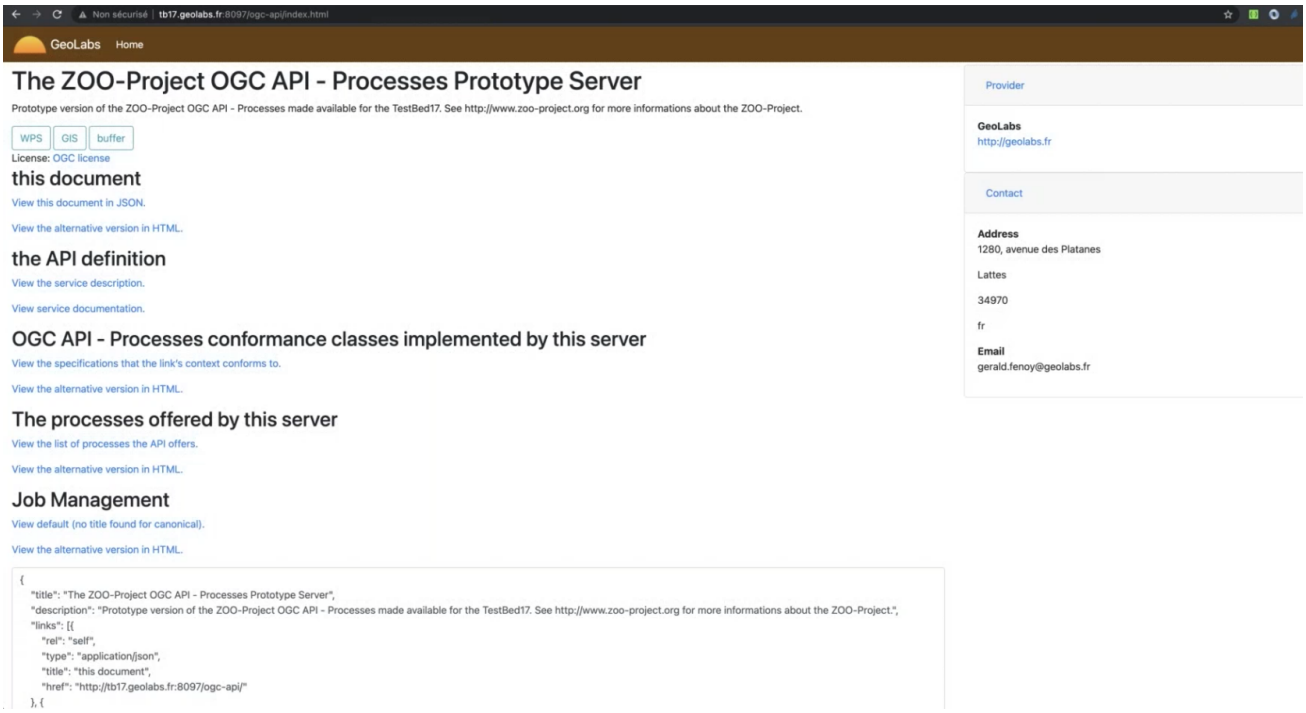

Figure 5. A landing page from a Geolabs ZOO Project instance

A screenshot of a request and response from a completed processing job executed on ZOO Project is shown in Figure 6.

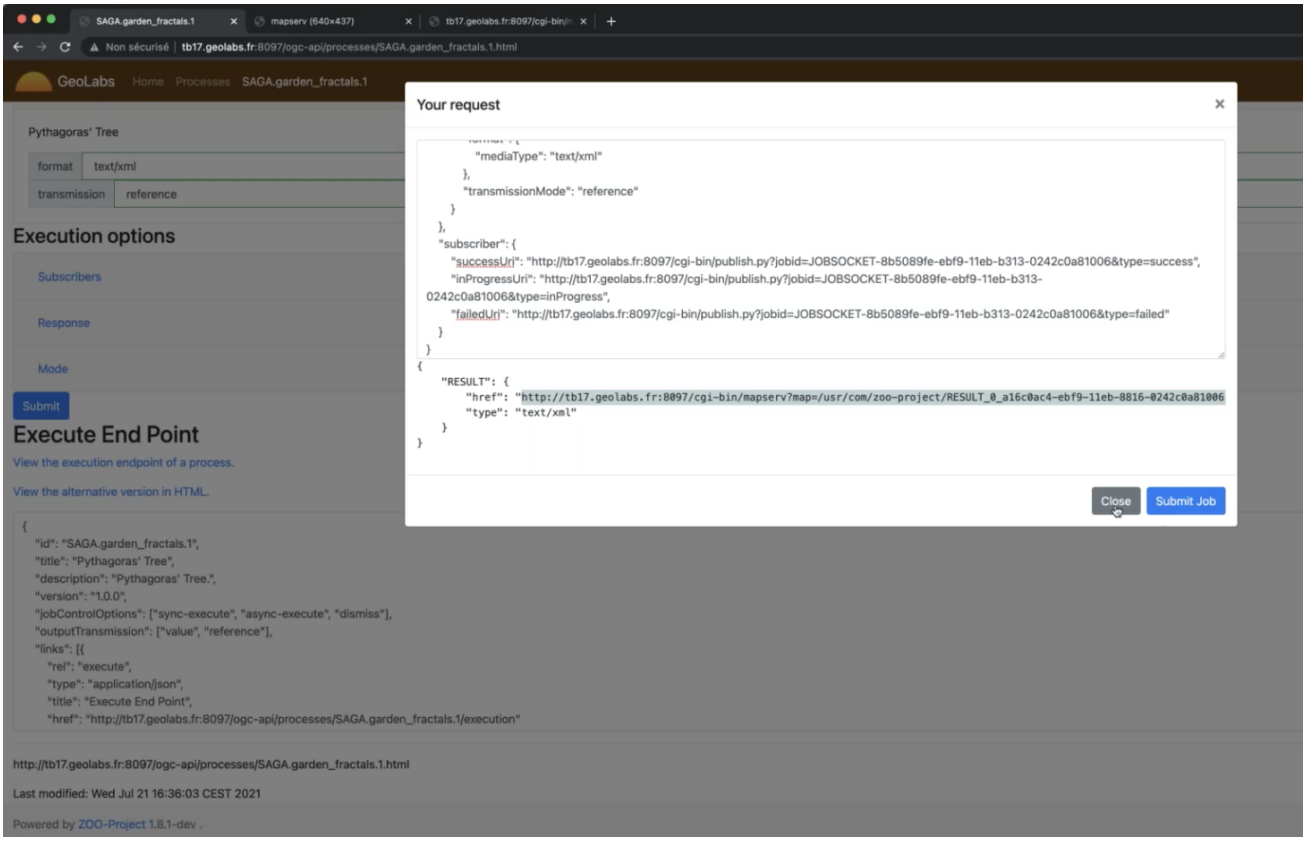

Figure 6. A request and response from a completed processing job executed on ZOO Project

7.4. GeoServer

GeoSolutions deployed an instance of the GeoServer product. GeoServer is a Java-based software server that allows users to view and edit geospatial data. GeoServer supports multiple OGC API specifications. For this code sprint, the server was configured to offer an endpoint supporting multiple conformance classes and recommendations from the draft OGC API - Coverages - Part 1: Core specification. The development focused on updating the OGC API - Coverages implementation to the latest evolution of the spec, building on GeoServer's existing support for the Web Coverage Service (WCS) standard.

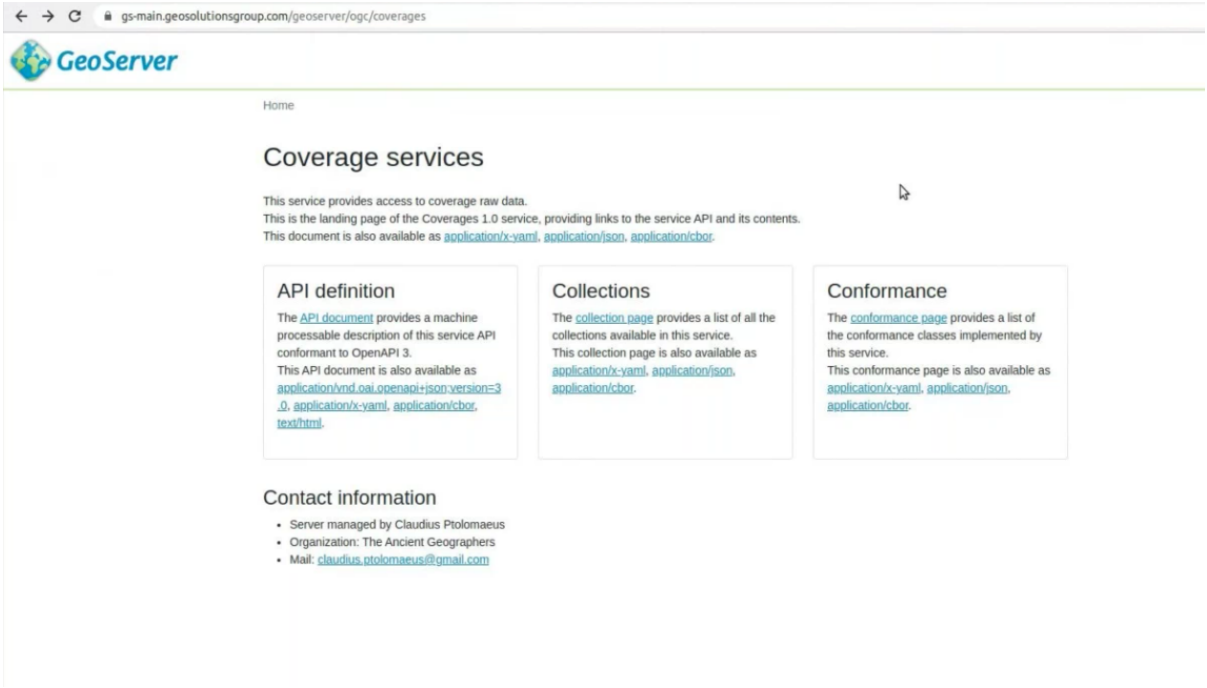

Figure 7. The landing page of the OGC API - Coverages implementation of GeoServer

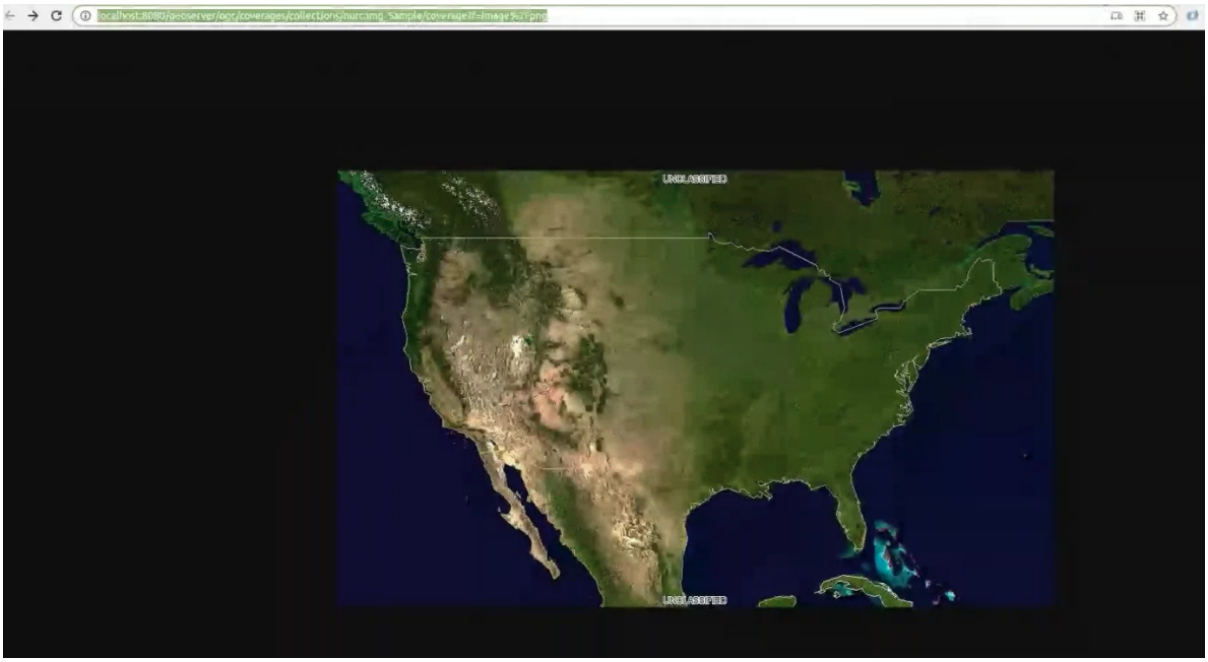

Figure 8. A coverage retrieved from the OGC API - Coverages implementation of GeoServer

7.5. Ecere GNOSIS Map Server

The GNOSIS Map Server is written in the eC programming language and supports multiple OGC API specifications. The API allows for retrieval of coverages, maps and tiles (both coverage and map tiles) of a dataset. Ecere configured the server to offer coverages through an interface conforming to OGC API - Coverages. Multiple encodings are supported including GNOSIS Map Tiles (which can contain either vector data, gridded coverages, imagery, point clouds or 3D meshes), GeoTIFF, GeoJSON, Mapbox Vector Tiles and MapML. For this code sprint, the server was used in support of the development of compliance tests.

7.6. Hexagon Geoprocessing Suite

Developers from Hexagon took part in the code sprint, focusing on the OGC API - Processes candidate standard. The developers configured the Hexagon Geoprocessing Suite, which includes a number of Hexagon products, to offer a variety of image analysis capabilities. The suite includes ERDAS APOLLO, which provides an end-to-end workflow to execute algorithms that operate on geospatial data. The algorithms were exposed as processes through an interface conforming to OGC API - Processes. The suite also includes ERDAS IMAGINE, a software product for image analysis and processing. The workflows were designed using the Spatial Model Editor from ERDAS IMAGINE product, a screenshot of which is shown in Figure 9.

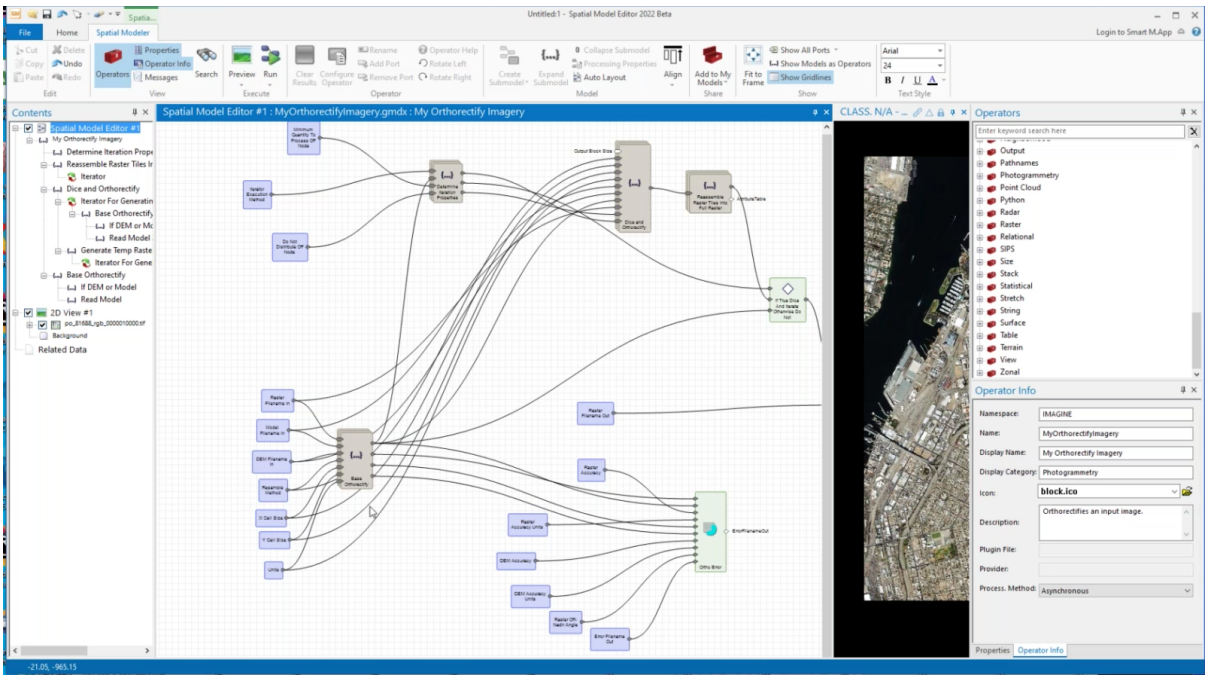

Figure 9. A screenshot of a workflow designed using Hexagon's Spatial Model Editor

ERDAS IMAGINE allows geospatial analysts to create custom models and to publish the models to the ERDAS APOLLO catalog, from where they can be executed by end-users as illustrated in Figure 10. While each processing job is running, an administrator can monitor the jobs through a status view as shown in Figure 11. This capability demonstrated the ability of OGC API - Processes to support both synchronous and asynchronous execution modes.

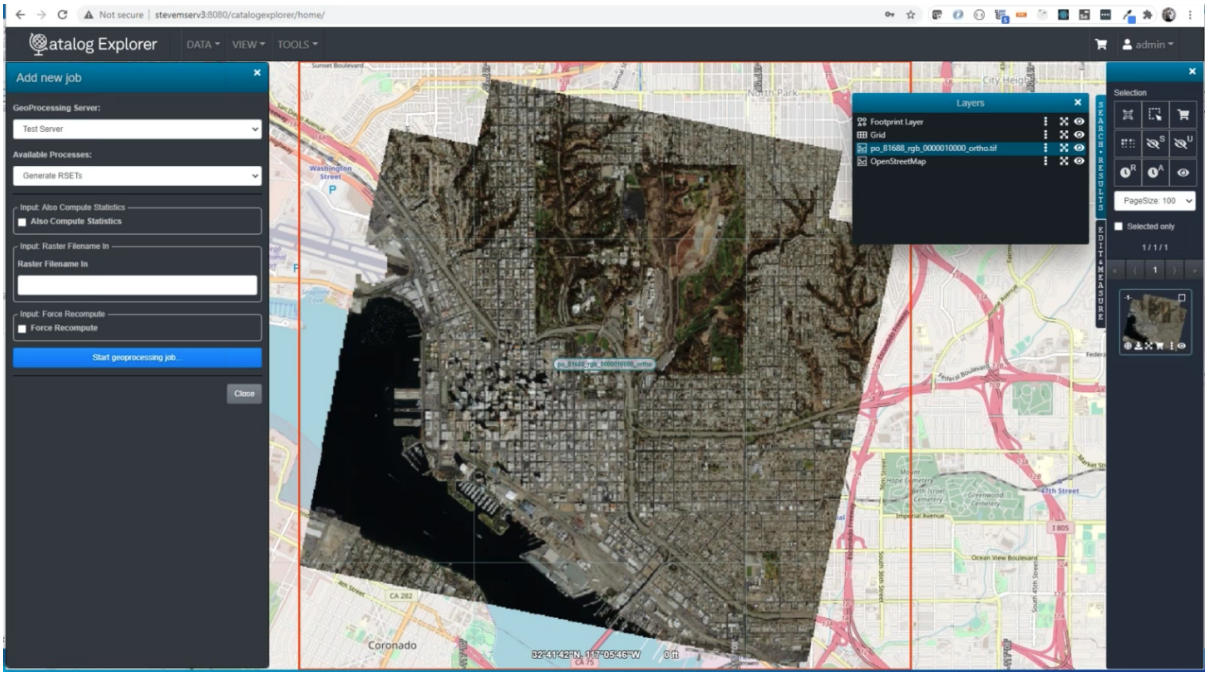

Figure 10. A screenshot of the output of a completed processing job in Hexagon Geoprocessing suite

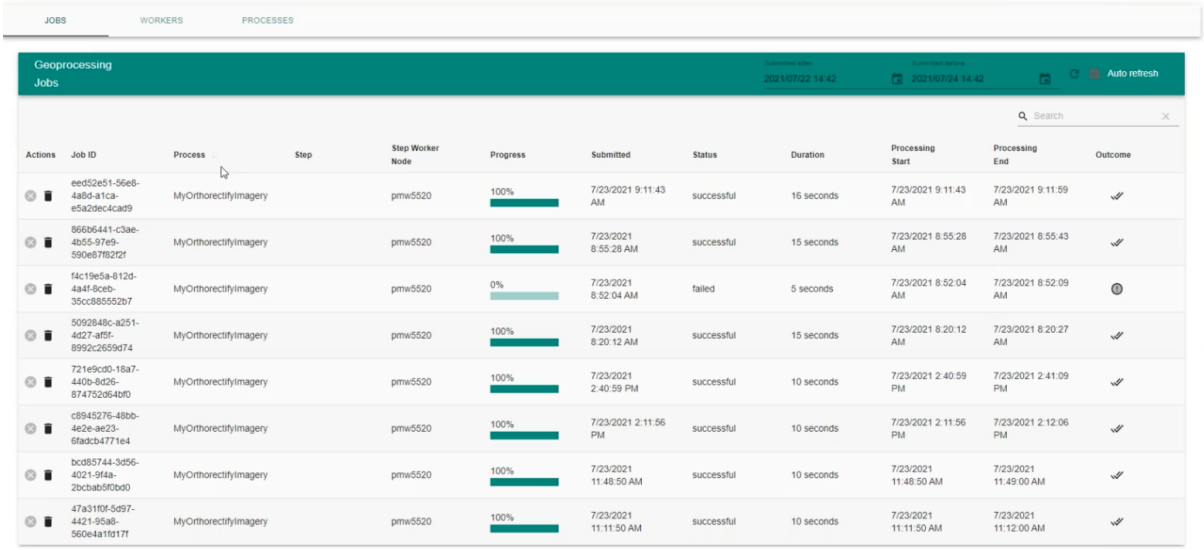

Figure 11. A screenshot of an overview of the status of processing jobs in Hexagon Geoprocessing suite

7.7. QGIS MetaSearch Plugin

During the code sprint, developers from OSGeo and MSC worked on an enhancement of the QGIS MetaSearch Plugin to enable support for OGC API - Records. MetaSearch is a QGIS plugin that enables QGIS to act as a client application for interacting with metadata catalogue services. A screenshot of the search dialog window of the QGIS MetaSearch plugin is shown in Figure 12.

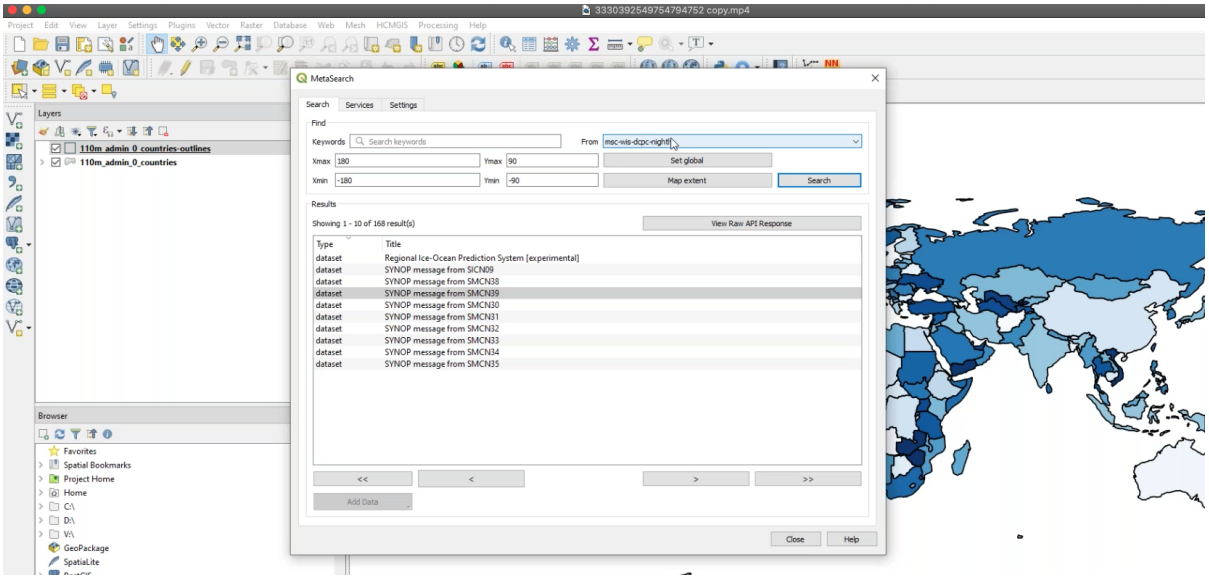

Figure 12. The search dialog window of the QGIS MetaSearch plugin

A screenshot of the metadata view window of the QGIS MetaSearch plugin is shown in Figure 13.

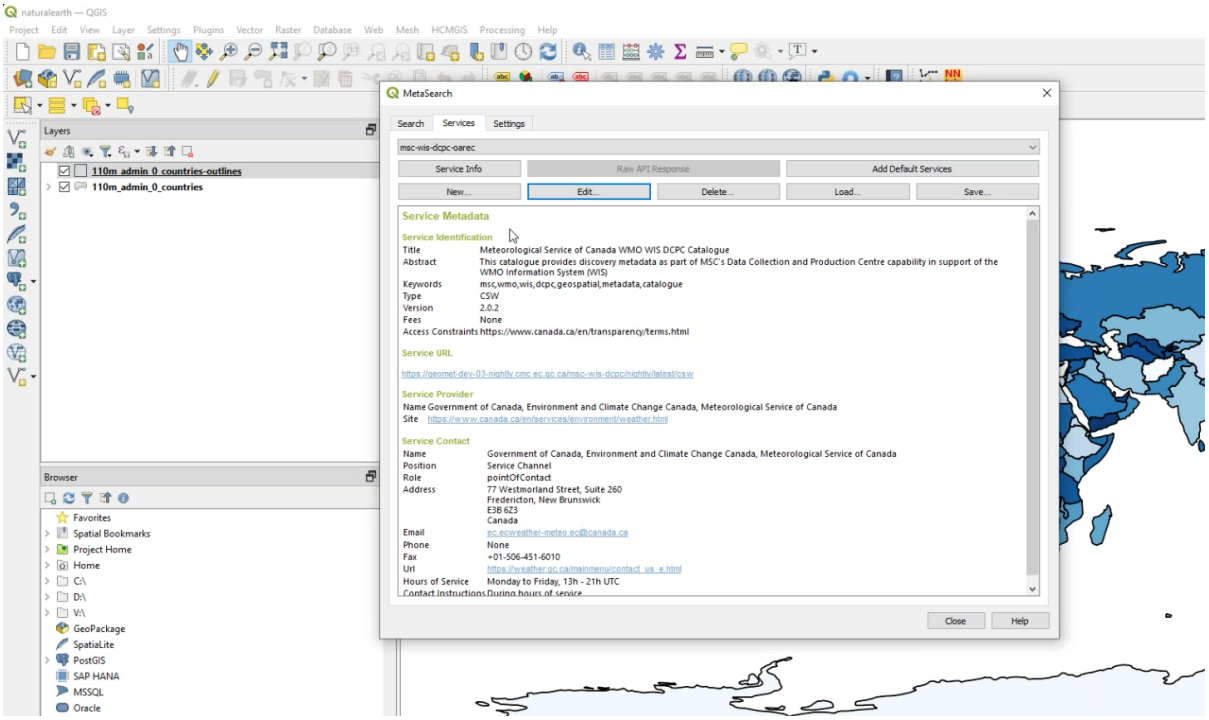

Figure 13. The metadata view window of the QGIS MetaSearch plugin

Another screenshot of the QGIS MetaSearch plugin windows is shown in Figure 14.

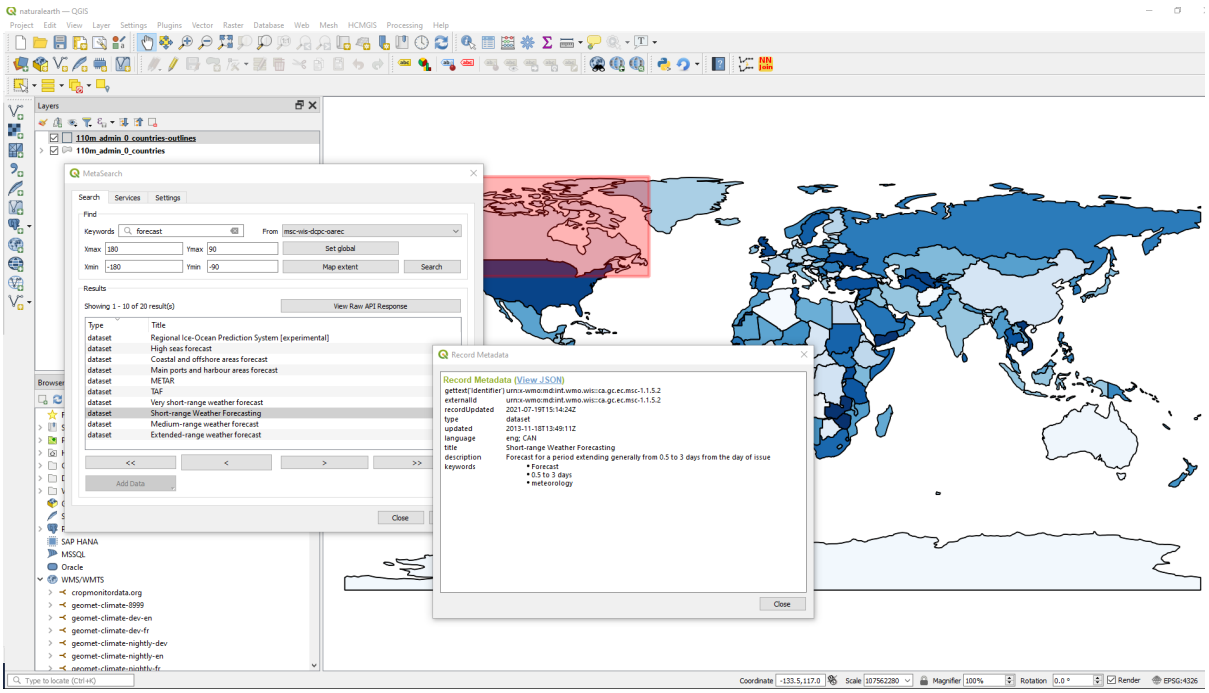

Figure 14. Another screenshot of the QGIS MetaSearch plugin windows

7.8. Mappings between ISO 19115-3 and OGC API - Records Queryables

Developers from OpenWork took part in the code sprint, with focus on metadata and OpenSearch Syndication. The work on metadata involved designing mappings between the 'queryables' model implemented by OGC API - Records and ISO 19115-3, the XML Schema of ISO 19115-1 the international geospatial metadata standard. The mappings have been archived on the sprint's GitHub repository [https://github.com/opengeospatial/ogcapi-code-sprint-2021-07/blob/main/mappings/ISO19115-3_Mappings.md].

7.9. OpenSearch Syndication Format

Some of the prototyping by developers from OpenWork involved implementation of an OpenSearch description document. The OpenSearch description document is used to describe a search engine so that it can be used by search client applications. Once configured on an OpenSearch client (ideally a web browser), the search engine becomes accessible as illustrated on Figure 15.

Figure 15. The landing page of a search engine's user interface

A screenshot of search results retrieved through OpenSearch Syndication is shown in Figure 16.

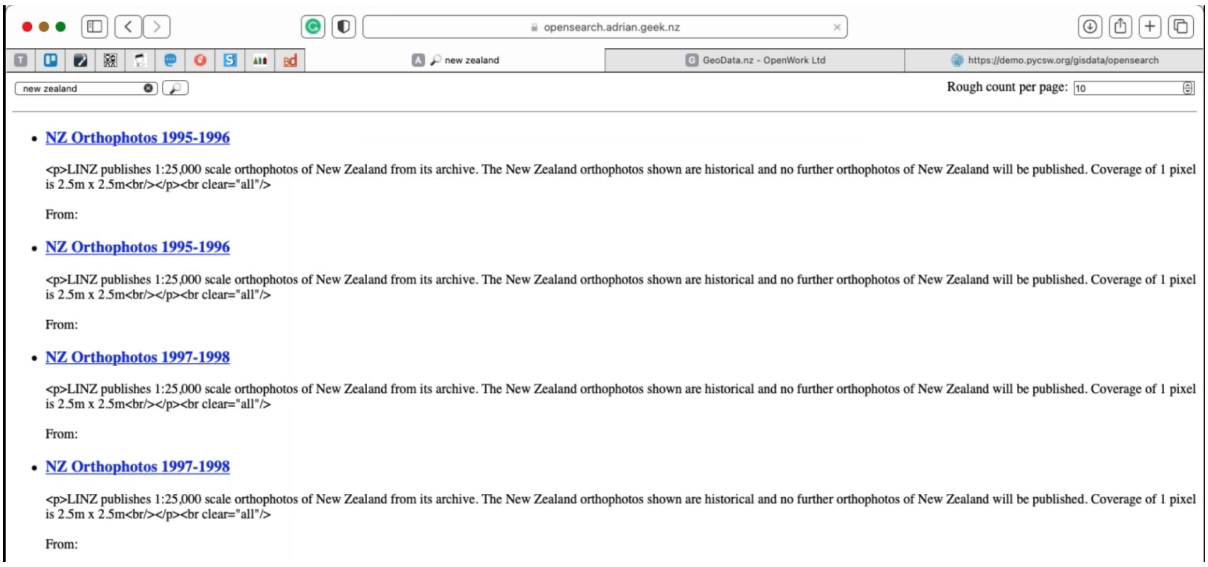

Figure 16. A screenshot of search results retrieved through OpenSearch Syndication.

7.10. pycsw

During the code sprint, developers from the Open Source Geospatial Foundation (OSGeo) deployed an instance of pycsw [https://pycsw.org], an open source server-side python implementation of the OGC Catalogue Services for the Web (CSW) standard. The pycsw software product allows for the publishing and discovery of geospatial metadata via numerous APIs (CSW 2.0.2, CSW 3.0.0, OpenSearch Syndication Protocol, and others), providing a standards-based metadata and catalogue component of spatial data infrastructures. The OSGeo developers extended the pycsw code base to include support for OGC API - Records. A screenshot of the landing page of one of the pycsw instances is shown in Figure 17.

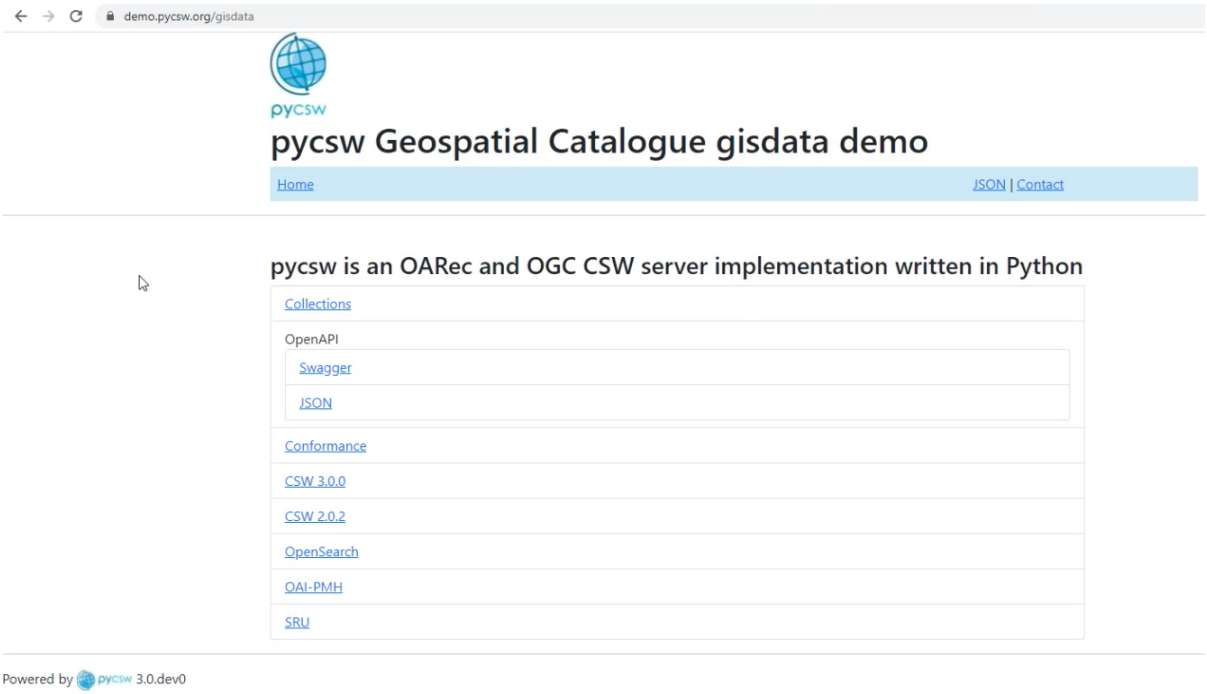

Figure 17. The landing page of one of the pycsw instances

A screenshot of a series of metadata records from a collection accessed on an instance pycsw is shown in Figure 18.

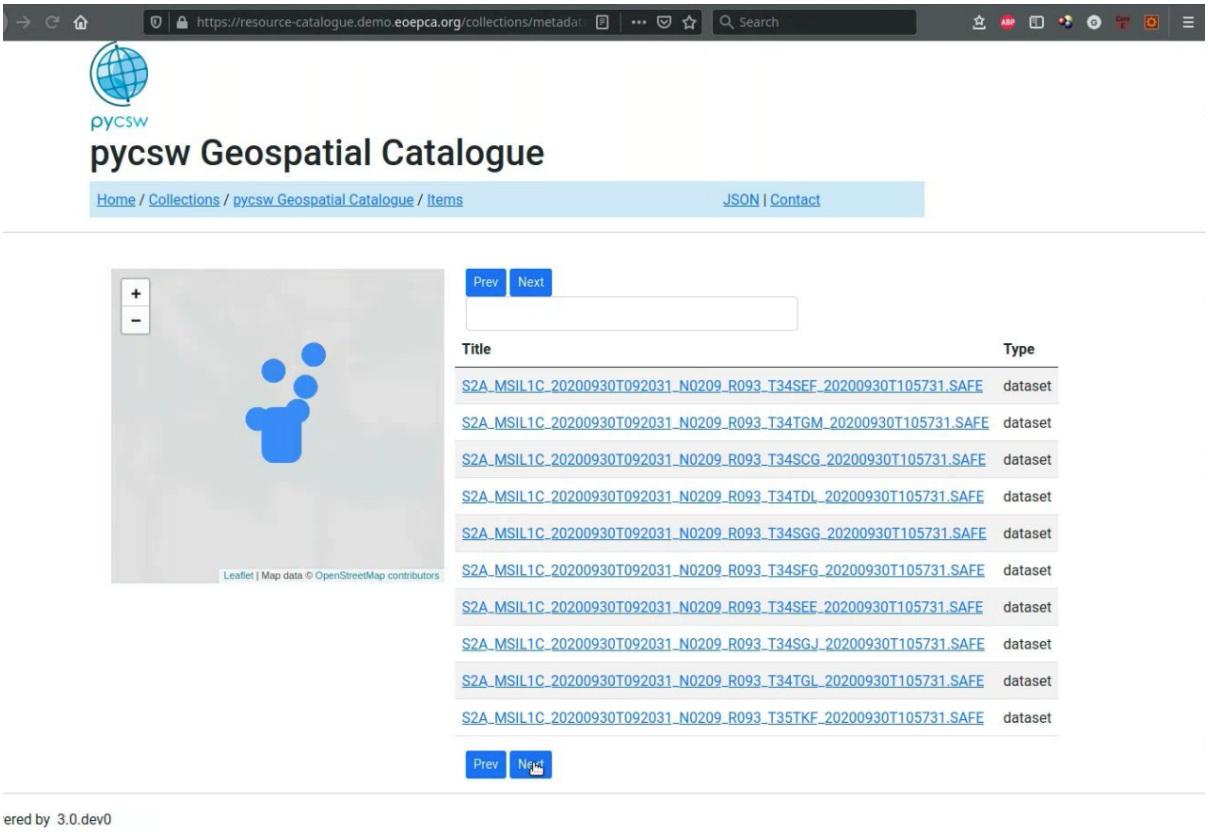

Figure 18. A series of metadata records from a collection accessed on an instance pycsw

7.11. pygeoapi

A number of sprint participants deployed instances of pygeoapi [https://pygeoapi.io] and worked

collaboratively to enhance the software's support for OGC APIs. Participants working on pygeoapi included developers from OSGeo, Meteorological Service of Canada (MSC), 52 North, GeoCat BV and Geobeyond Srl. pygeoapi is an open source Python server implementation of the OGC API suite of standards. The product supports the microservices approach and allows for scalability and cloud friendly deployment.

7.11.1. Accessing metadata records

Developers from MSC, OSGeo, GeoCat BV and Geobeyond Srl configured an instance of pygeoapi to enable access to metadata records. A screenshot of the landing page of one of the pygeoapi instances is shown in Figure 19.

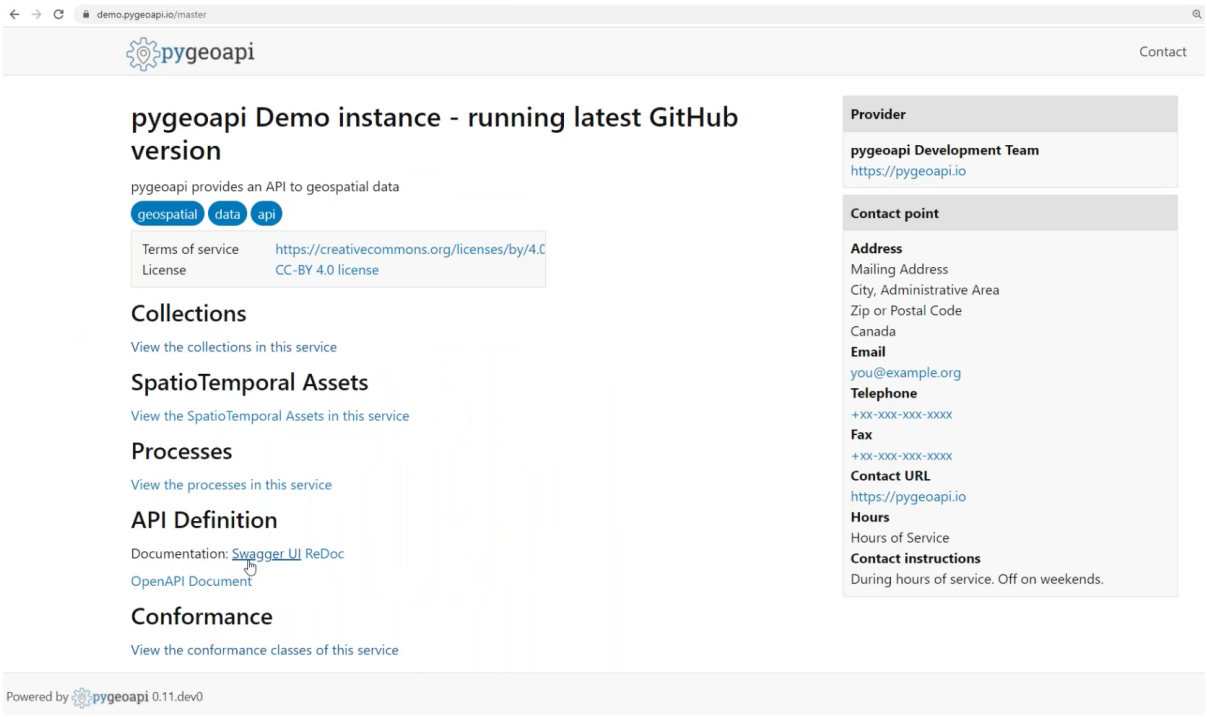

Figure 19. The landing page of one of the pygeoapi instances

A screenshot of an overview of one of the collections of metadata records offered by one of the pygeoapi instances is shown in Figure 20.

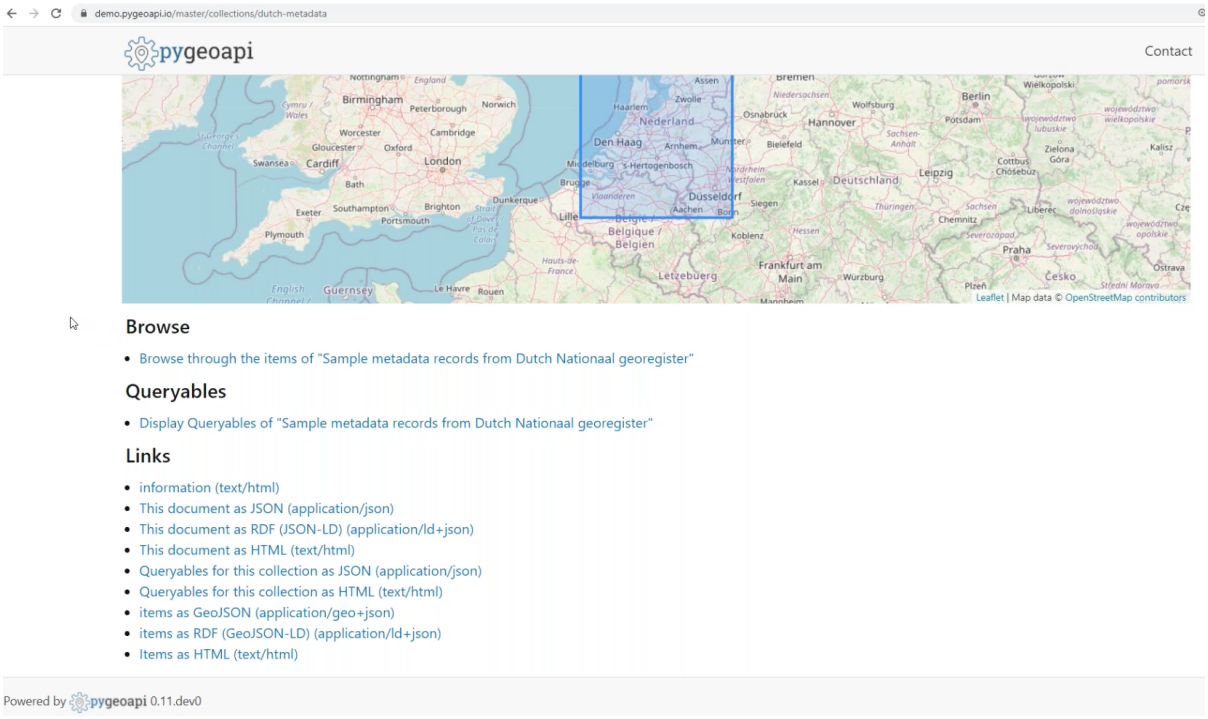

Figure 20. An overview of one of the collections of metadata records offered by one of the pygeoapi instances

7.11.2. Accessing processes

Developers from MSC, OSGeo, GeoCat BV and Geobeyond Srl configured an instance of pygeoapi to enable access to processes through an interface conforming to OGC API - Processes. A screenshot of a rendering of the API definition of one of the processes is shown in Figure 21.

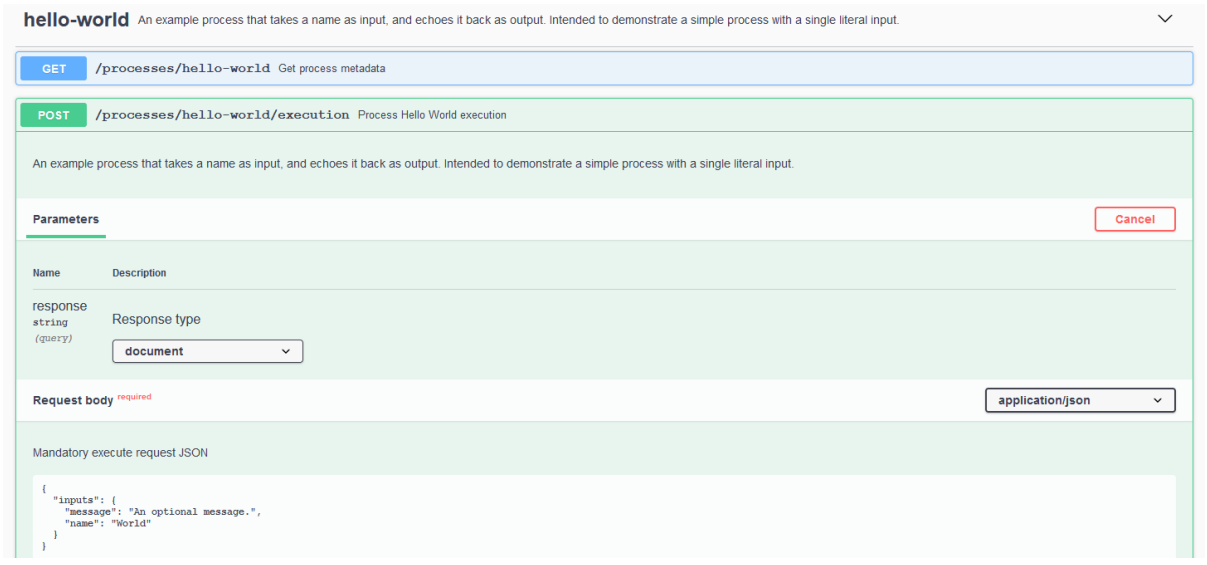

Figure 21. A screenshot of a rendering of the API definition of one of the processes

7.11.3. Accessing coverages

Developers from 52 North configured an instance of pygeoapi to enable access to Data Cubes, initially deployed for Testbed-17. A screenshot of the pygeoapi interface from the 52 North Data Cube demonstration is shown in Figure 22.

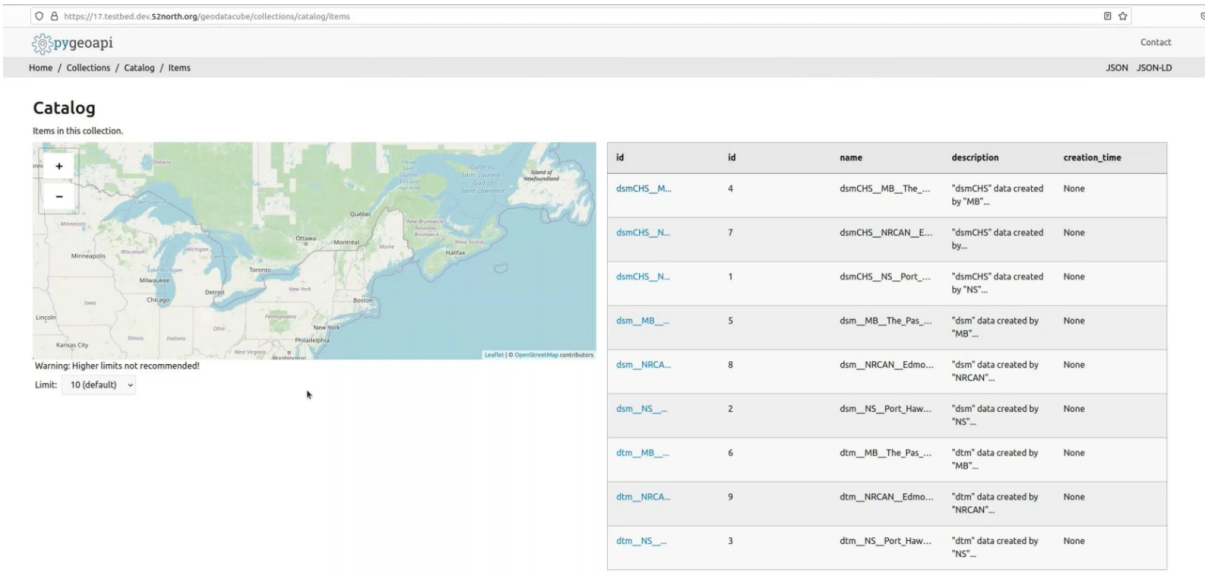

Figure 22. Screenshot of pygeoapi interface from the 52 North Data Cube demonstration

Once coverages from the data cube are published through the pygeoapi, they can be accessed through the OGC API - Coverages interface and displayed on a client application such as QGIS. A coverage displayed using QGIS after download from a data cube is shown in Figure 23.

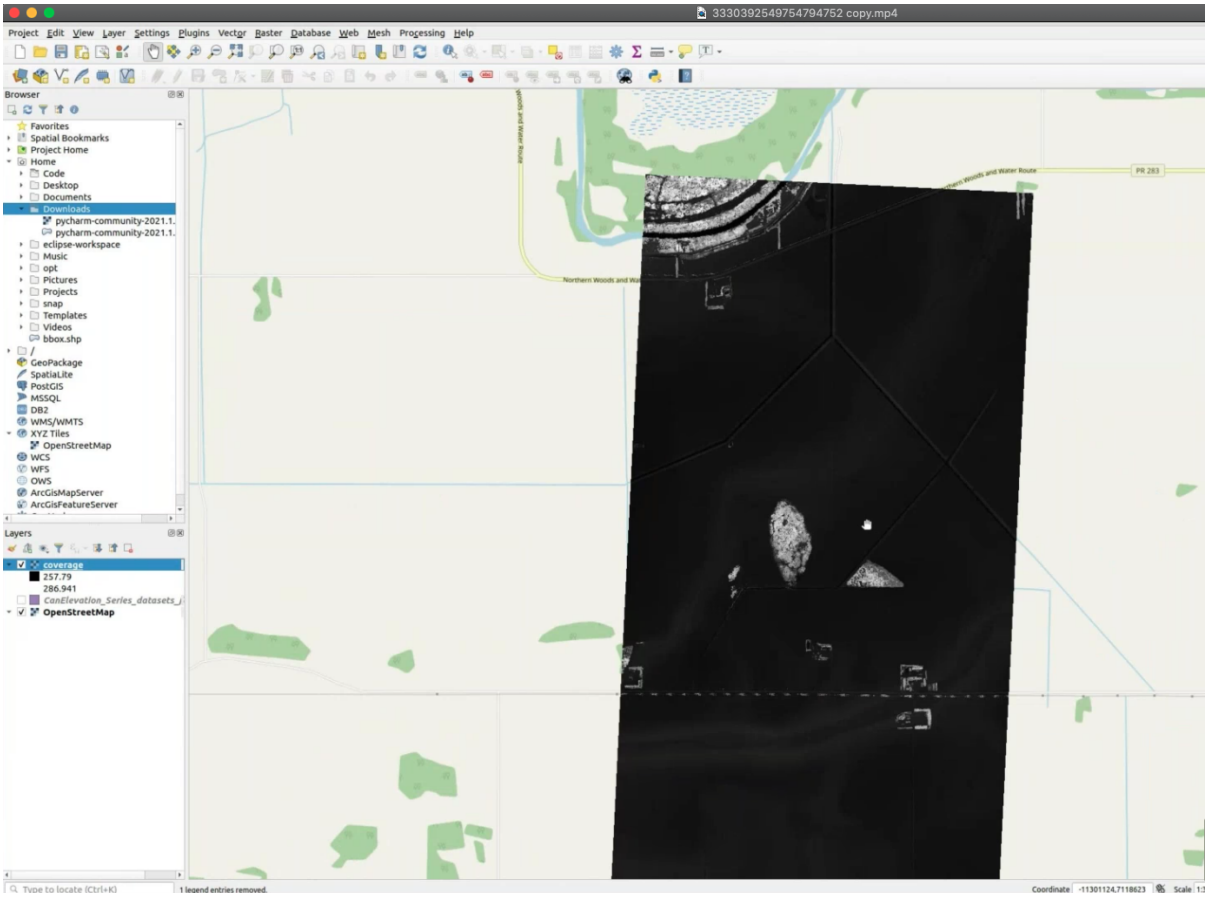

Figure 23. Coverage displayed using QGIS after download from the 52 North Data Cube supplied using pygeoapi

7.12. Other Outputs

Other developments and software deployments during the code sprint included:

- The OGC Compliance Program provides a free online testing facility, called the OGC Validator, that is based on Team Engine. The Compliance Program also provides a set of test suites dedicated to specific protocols and versions, as well as specification profiles and extension. For this code sprint, an Apache Maven [https://maven.apache.org] project was set up in a GitHub repository [https://github.com/opengeospatial/ets-ogcapi-coverages10] for the executable test suite of OGC API – Coverages draft standard.
- A draft user guide [https://github.com/opengeospatial/ogcapi-code-sprint-2021-07/tree/main/Draft_Spring_Guide_for_OGC_API_Proceses] was developed for software developers that use the Spring Framework [https://spring.io/projects/spring-framework] to build implementation of OGC API - Processes. The Spring Framework is an open source application framework and inversion of control container for the Java Platform. The user guide developed during the code sprint made use of Spring along with OpenAPI Tools Generator.

Chapter 8. Discussion

This sections provides a summary of the discussion.

8.1. Potential relationship between STAC and OGC API - Records

The sprint participants discussed the potential relationship between STAC and OGC API - Records, in order to identify opportunities for alignment. This subsection summarises that discussion.

The sprint participants noted that there had been previous discussions between OGC staff and the STAC community about the possibility of bringing the STAC API into the OGC's standardization process. This was partly because the STAC API is an extension of OGC API - Features - Part 1: Core. The current thinking is to submit STAC into the standardization process as a candidate OGC Community Standard. That would require STAC to be stable, at version 1.0 at least.

Some of the opportunities for alignment between STAC and OGC API - Records are static equivalents for Collections and Records. The sprint participants noted that, in a previous discussion, there was an observation that there is a need for a 'Dataset Record'. A 'Record' would be a flexible and generic concept, whereas a Dataset Record would represent dataset metadata such as in the context of ISO 19115. The grouping would be the Collection. With these concepts in place, it would potentially be possible to have STAC as a Record or an extension of OGC API - Records.

The sprint participants noted that in OGC API - Records, in contrast to STAC, the actual Record description is distinct from the API. The Record description is loosely bound to the API in OGC API - Records, for example, there is no requirement for the record to be encoded in GeoJSON as other encodings (e.g. ATOM) are allowed. The Record is also not bound to a particular dataset or asset. There is an expectation that developers would take the Record and enrich the content to suit specific needs. There was an observation made that it would benefit developers to extract the description of the Record content model and make it more standard-alone on either the specification document as a building block or in an accompanying document such as user guide.

What is the distinction between the STAC Catalog and the STAC Collections specifications? The STAC Catalog is a JSON file of links that provides a structure to organize and browse STAC Items. The STAC Collection is an extension of the STAC Catalog with additional information that describes STAC Items that fall within the Collection. The STAC Catalog specification is therefore less tied to the API. It is just a structured concept that says "this links to more items". If you have millions of records, instead of placing them in a single collection, the catalog lets you break them up in arbitrary ways. The approach is based on OGC API - Features, and so a STAC Collection can be served as a Collection from an OGC API - Features implementation.

The Records API does not do anything different to what the Features API does. That is, the two APIs are consistent. This suggests that the biggest need is for the Dataset Collection, with the

fields in the collection clearly defined. This is necessary because Collection is non-GeoJSON, whereas Record is GeoJSON. They should translate between each other easily. This could, for instance, lead to a static Features API involving a series of files in a Cloud Storage resource. The sprint participants observed that, from a Meteorological perpestive, it would be useful to have a static API that provides access to a series of files.

STAC inherits from an OGC API - Features collection, but ideally the preference would be to inherit from a Dataset Collection. So potentially there could be two separate representations of the collection, at different levels of the API hierarchy, using an 'alternate' relation type to provide links to the two different representations. The sprint participants suggested that Part 1 should be more generic and then consider the Dataset Collection for Part 2. Support for datasets should be optional and could be based on ISO 19115-1 metadata. This could then lead to a Part 3 that focuses on STAC. The work on STAC should not start until the Part 1 is stable.

There was also discussion about the structure of the Part 1 document, including whether some conformance classes (e.g. sort by) should be moved to OGC API - Features. This also included consideration of where a section describing the Collection could be placed in OGC API - Records - Part 1: Core. This capability would facilitat the creation of static catalogs. A static catalog would offer value to some user needs. Such a capability could have more utility and wider uptake if included in OGC API - Features, and possibly in the long term it could be part of OGC API - Common. The capability could also be specified in OGC API - Records and still have utility in OGC API - Features. In OGC API - Features, by default, every API implementing the standard will provide access to a single dataset.

8.2. TBA

TBA

8.3. TBA

TBA

8.4. TBA

TBA

8.5. TBA

TBA

8.6. TBA

TBA

8.7. Lessons Learnt

Towards the end of the sprint, participants held a discussion on the lessons learnt from the initiative. A summary of the lessons identified by the sprint participants is presented below:

- RecordsAPI: The relationship with STAC was clarified in this sprint. The direction of the RecordsAPI is aligning more with STAC.
- RecordsAPI: Crosswalk aligned ISO19115-3 with the OGC Record.
- There may be a little bit of work to do in aligning satellite imagery support between RecordsAPI and STAC.
- Clarity on what it means to have an ISO 19115 extension. The pycsw/pygeoapi? demo showed how.
- The change with OpenSearch.org introduces challenges in terms of usage/licence. Will mean change requests for existing OGC OpenSearch standards.
- OpenSearch is still an active thing in OGC. It will be placed in an extension of OGC API - Records and not the Core.
- OGC Member Meeting in September 2021 (Metadata Adhoc) will have a session on tackling metadata issues. It's linked to the Singapore Geo Festival.
- Lots of implementations of OGC API - Processes provides confidence that it works.
- The Spring work will help develop Guides.
- OGC API - Processes is sufficiently different from WPS
- The use of JSON Schema to describe inputs and outputs is going to be great for uptake.
- There is an increasing need for Best Practices and Profiles. e.g. Routing, Maps, NDVI, …
- OGC API - Processes weather toolbox (we're starting to think about this in MetOceanDWG)

Chapter 9. Conclusions

The code sprint successfully met all of its objectives and achieved its goal of progressing the development of the OGC API - Processes draft standard, OGC API - Records draft standard, and the OGC API – Coverages draft standard. There were no issues found on the Processes API, which helped to validate the maturity and stability of the API. Discussion about the Records API helped to improve understanding of the potential relationship to other related specifications (e.g. STAC and ISO 19115). Issues raised relating to the Coverages API, were focused around clarification, thereby identifying areas where the documentation could be improved. The sprint participants demonstrated that the OGC API pattern can effectively address the needs of communities that use, process and analyze geospatial data.

9.1. Future work

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

9.1.1. Ideas for the Innovation Program

Future work in Innovation Program could include:

- Development of Developer Guides (e.g. Spring, Python, nodeJS etc), including around toolchains
- Some experimentation on Workflows (OGC API - Processes - Part 3), covering chaining
- The filtering and coverage processing. Filtering at the metadata level and at the coverage cell value level
- GeoDataClass (aka StylableLayerSet) concept in relation to discovering Processes & Data that can be used together, including in the context of Workflows
- Some along the workflow of a search process (e.g. ISO 19115, STAC, etc).

9.1.2. Ideas for the Standards Program

Future work in the Standards Program could include:

- ETS of OGC API - Coverages, Records
- Merge the work done on the openapi schema for the Coverages API
- Transactions on OGC API - Processes (Part 2)
- OGC API - Processes - Part 3: Workflows & Chaining

Appendix A: Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
2020-11-12	G. Hobona	.1	all	initial version

Table 1. Revision History

Appendix B: Bibliography