OGC API Hackathon 2019 Engineering Report

Table of Contents

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/t14-ID

Category: OGC Public Engineering Report

Editor: Name(s)

Title: OGC API Hackathon 2019 Engineering Report

OGC Public Engineering Report

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.

Chapter 1. Subject

The subject of this Engineering Report (ER) is a hackathon event that was held from June 20th to 21st, 2019 to advance the development of OGC Application Programming Interface (API) specifications. An 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). The OGC API Hackathon 2019, as the event was called, was hosted by the Geovation Hub in London, United Kingdom. The event was sponsored by the European Space Agency (ESA) and the Ordnance Survey.

Chapter 2. Executive Summary

The following is, as all texts in double square brackets, a helper text. Please remove this and all other helper texts once done.

The Executive Summary clause shall contain the key findings and results in a concise form. A more detailed description of the findings should be in the body of the report.

The Executive Summary shall contain a business value statement that should describe the value of this Engineering Report to improve interoperability, advance location-based technologies or realize innovations.

This section shall include precise descriptions of the requirements that have been addressed by the work documented in this Engineering Report; together with the research motivation that answers the fundamental question: What motivated us to address this topic in this report?

This section provides an overview of recommendations on how to further proceed with the achievements documented in this ER.

This section shall be between 1-3 pages.

The output of this hackathon should lead to a solid, common core and advancement of a whole new generation of OGC standards that are flexible in modern IT environments.

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	TBA
Scott Simmons	Open Geospatial Consortium	TBA
Michael Gordon	Ordnance Survey	TBA
Simon Green	Ordnance Survey	TBA
Clemens Portele	interactive instruments GmbH	TBA
Jorge Samuel Mendes de Jesus	ISRIC - World Soil Information	TBA
Peter Baumann	rasdaman GmbH	TBA
Francesco Bartoli	Geobeyond	TBA
Dirk Stenger	lat/lon GmbH	TBA

Name	Organization	Role
Chuck Heazel	Heazeltech LLC	TBA
Matthias Mohr	University of Münster	TBA
Robin Houtmeyers	Hexagon Geospatial	TBA
TBA	TBA	TBA
TBA	TBA	TBA
TBA	TBA	TBA

NOTE Role = Editor and/or 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.

NOTE: Only normative standards are referenced here, e.g. OGC, ISO or other SDO standards. All other references are listed in the bibliography. Example:

- OGC: OGC 06-121r9, OGC® Web Services Common Standard [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2]
- OGC: OGC 17-069, OGC Web Feature Service 3.0: Part 1 - Core, Version 3.0.0-draft.1 [https://cdn.rawgit.com/opengeospatial/WFS_FES/3.0.0-draft.1/docs/17-069.html]
- OpenAPI Initiative: OpenAPI Specification 3.0.1 [https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.1.md]
- OGC: OGC 09-146r6, OGC® Coverage Implementation Schema, version 1.1, 2017 [http://docs.opengeospatial.org/is/09-146r6/09-146r6.html]
- OGC: OGC Web Coverage Service (WCS) 2.1 Interface Standard - Core, 2018 [http://docs.opengeospatial.org/is/17-089r1/17-089r1.html]
- OGC: OGC 14-065r2, OGC® WPS 2.0.2 Interface Standard Corrigendum 2, 2018 [http://docs.opengeospatial.org/is/14-065/14-065.html]
- OGC: OGC OGC 07-057r7, OpenGIS® Web Map Tile Service Implementation Standard, 2010 [http://portal.opengeospatial.org/files/?artifact_id=35326]
- OGC 09-025r2, OGC® Web Feature Service 2.0 Interface Standard – With Corrigendum, 2014 [http://docs.opengeospatial.org/is/09-025r2/09-025r2.html]

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.

- **application programming interface**

  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)

- **feature**

  abstraction of real world phenomena (source: ISO 19101-1:2014)

- **OpenAPI definition | OpenAPI document**

  a document (or set of documents) that defines or describes an API and conforms to the OpenAPI Specification [derived from the OpenAPI Specification]

- **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 (OGC 07-011).

- **Regular grid**

  grid whose grid lines have a constant distance along each grid axis

- **process**

  A process p is a function that for each input returns a corresponding output

  $$p: X \to Y$$

  where X denotes the domain of arguments x and Y denotes the co-domain of values y. Within the Web Processing Service (WPS) standard, process arguments are referred to as process inputs and result values are referred to as process outputs. Processes that have no process inputs represent value generators that deliver constant or random process outputs.

- **service**

  distinct part of the functionality that is provided by an entity through interfaces (source: ISO/IEC TR 14252)

- **operation**

  specification of a transformation or query that an object may be called to execute (source: ISO 19119)

- **request**

  invocation of an operation by a client

- **response**

  result of an operation, returned from a server to a client

4.1. Abbreviated terms

- API Application Programming Interface
- CRS Coordinate Reference System
- GML Geography Markup Language
- HTML Hypertext Markup Language
- HTTP Hypertext Transfer Protocol
- JSON Java Script Object Notation
- WCS Web Coverage Service
- WFS Web Feature Service
- WMTS Web Map Tile Service
- OWS OGC Web Services
- REST Representational State Transfer
- XML Extensible Markup Language

Chapter 5. Overview

Section 6 introduces the OGC API Hackathon by describing the challenge, the scenario adopted, and the infrastructure used by the participants. The section also presents overviews of the datasets and services identified to support participants during the Hackathon. The section also presents a list of the organizations represented by the participants.

Section 7 presents the solution architecture developed in this hackathon. The section identifies the client and service implementations of the OGC API specifications.

Section 8 describes each of the OGC API specifications that were involved in the hackathon.

Section 9 provides summary of the main findings and discusses alternative approaches that could have been taken, as well as experiences and lessons learnt.

Appendix A provides reports from each participating organisation, covering their motivation for participating, a description of their implementation of the OGC API specifications, alternative approaches, their experiences, impressions and recommendations.

Chapter 6. Introduction

The development of OGC API specifications is not a new activity within the Consortium, as OGC members and staff have been investigating OpenAPI (and its commercial equivalent, Swagger) in a concentrated effort since 2016. This effort was the result of a recognition that although the existing OGC web service standards are in effect web APIs, there are a number approaches adopted by modern web API frameworks that would require a fairly fundamental change in underlying design.

Two documents really provided the initial energy to get serious about redesign: the OGC Open Geospatial APIs White Paper, edited by George Percivall [1], and the Spatial Data on the Web Best Practices, jointly developed by OGC and the World Wide Web Consortium (W3C) [2]. These documents highlighted how geospatial data should be more native to the web. Further, OGC staff were working on “implementer-friendly” views of OGC standards and experimented with an OpenAPI definition for the Web Map Tile Service (WMTS).

But perhaps the most important impact was the leap of the OGC Web Feature Service (WFS) and Filter Encoding Service (FES) Standards Working Group (SWG) that rebuilt the WFS standard with an integrated OpenAPI definition as core to description of how to build against the standard. The work on WFS, which has resulted in the OGC API - Features specification (formerly called WFS 3.0), benefited from a two-day Hackathon held in 2018. Since then, other OGC web service SWGs have begun to independently develop API specifications based on their relevant OGC web service standards.

Numerous discussions occurred at OGC quarterly Technical Committee (TC) Meetings to consider those elements being developed in each SWG which should be common to all web API standards. These discussions came to a head at the February 2019 TC Meeting in Singapore, where a series of working group meetings and common sessions for the whole TC Membership reinforced the desire to work on a common framework for many OGC web standards and to develop a nomenclature for labeling these standards. Thus, the pattern “OGC API [resource]” was coined. The discussions in Singapore also resulted in the planning of the OGC API Hackathon to define and test common elements from Coverages, Map Tiles, and Processing standards work using foundational material from the Features work.

The OGC API Hackathon 2019 was hosted by the Geovation Hub in London, United Kingdom, from June 20th to 21st, 2019. The hackathon was sponsored by the European Space Agency (ESA) and the Ordnance Survey. The **goal** of the hackathon was to advance the development of OGC API specifications.

The objectives of the hackathon were set out as to:

- develop, deploy and test services/clients that support OGC APIs
- suggest improvements for a common core
- define rules/guidance that can be documented
- validate work that has been completed to date
- contribute to the GitHub repositories

6.1. Overview of the Challenge

The challenge of the Hackathon was to define and test common elements from Coverages, Map Tiles, and Processing standards work using foundational material from the Features work. The magnitude of this challenge was reflected by the fact that the OGC API specifications for Coverages, Map Tiles, and Processing were all at different stages of development. Therefore the Hackathon had to advance the development of all of the specifications to a stage where common elements across all of the specifications could be identified.

6.2. Scenario

NOTE	This section is a working draft.

To facilitate the development of the OGC API specifications, the scenario presented in this section was provided as a reference for the teams. The scenario is based on flood risk management and is motivated by recents events such as the 2018 floods that affected parts of Europe (including the United Kingdom, Italy, France, Spain and Portugal) [3] and the 2019 floods that affected parts of the United States [4]. The scenario draws from the OGC's Disasters Interoperability Concept Development Study (CDS) which assessed geospatial Web services across the disaster domain, defining the core components of National Spatial Data Infrastructure (SDI) architecture for disasters (Disasters SDI), and defining use cases and scenarios for future implementations as part of a follow-on pilot phase [5].

Risk mitigation is one of the phases in the 'life cycle' of disaster management [5], which includes the steps shown in Figure 1. *Mitigation* refers to taking sustained actions to minimise or completely eliminate the long-term risk from hazards and their effects to individuals and property. *Preparedness* refers to building the emergency management capabilities to respond effectively to any hazard, as well as to recover from the hazard. *Response* refers to conducting emergency operations that reduce the hazard to acceptable levels (or eliminate it entirely) in order to save lives, through evacuation of potential victims, and provision of food, water, medical care and shelter to those affected by the disaster. *Recovery* refers to the rebuilding of communities that have been affected by a disaster so that those communities, as well as their governments, can return to normality and function on their own. A more detailed discussion on disaster management can be found in the OGC Development of Disaster Spatial Data Infrastructures for Disaster Resilience Engineering Report [5].

Figure 1. Disaster management cycle

As part of Government flood risk management policy, Local Authorities have to carry out a preliminary flood risk analysis. Using satellite imagery, flood risk data, along with asset information, vulnerable property information and topographic data, Local Authorities carry out analysis to improve resilience and promote a more efficient use of resource.

A Local Authority is tasked with identifying at-risk residential properties in order to assist in flood prevention and amelioration. By carrying out this task, the Local Authority aims to reduce the number of residential properties affected by floods, as well as to decrease the economic and social costs associated with such devastating events. The Geospatial Specialists at the Local Authority embark on the steps presented in Table 1 in order to carry out the task.

Table 1. Steps in the flood risk management scenario

Step	Description	Notes
1	Receive satellite imagery, digital terrain model, Flood Risk Zone, address, and topographic data	
2	Overlay flood assets such as culverts, levees etc.	
3	Combine multiple datasets together.	
4	Data analysis to assess/quantify flood risk.	A number of hydrology approaches may be applied e.g. run-off modelling
5	Identify at risk properties and possible remediation strategies.	

Step	Description	Notes
6	Execute cost-benefit analysis to determine priorities.	
7	Commission work for on-the-ground implementation. This may be carried out by internal or external teams.	
8	Impact of remediation work assessed by external engineering consultant.	

The illustration in Figure 2 shows the Area of Interest (AOI) that was selected to facilitate prototyping, demonstration and briefings. The AOI covered the region of Carmarthenshire, Wales and focused on the town of Carmarthen. The region was the site of significant flooding in October 2018 and hence provided an appropriate location to based the flood-based scenario adopted for the Hackathon.

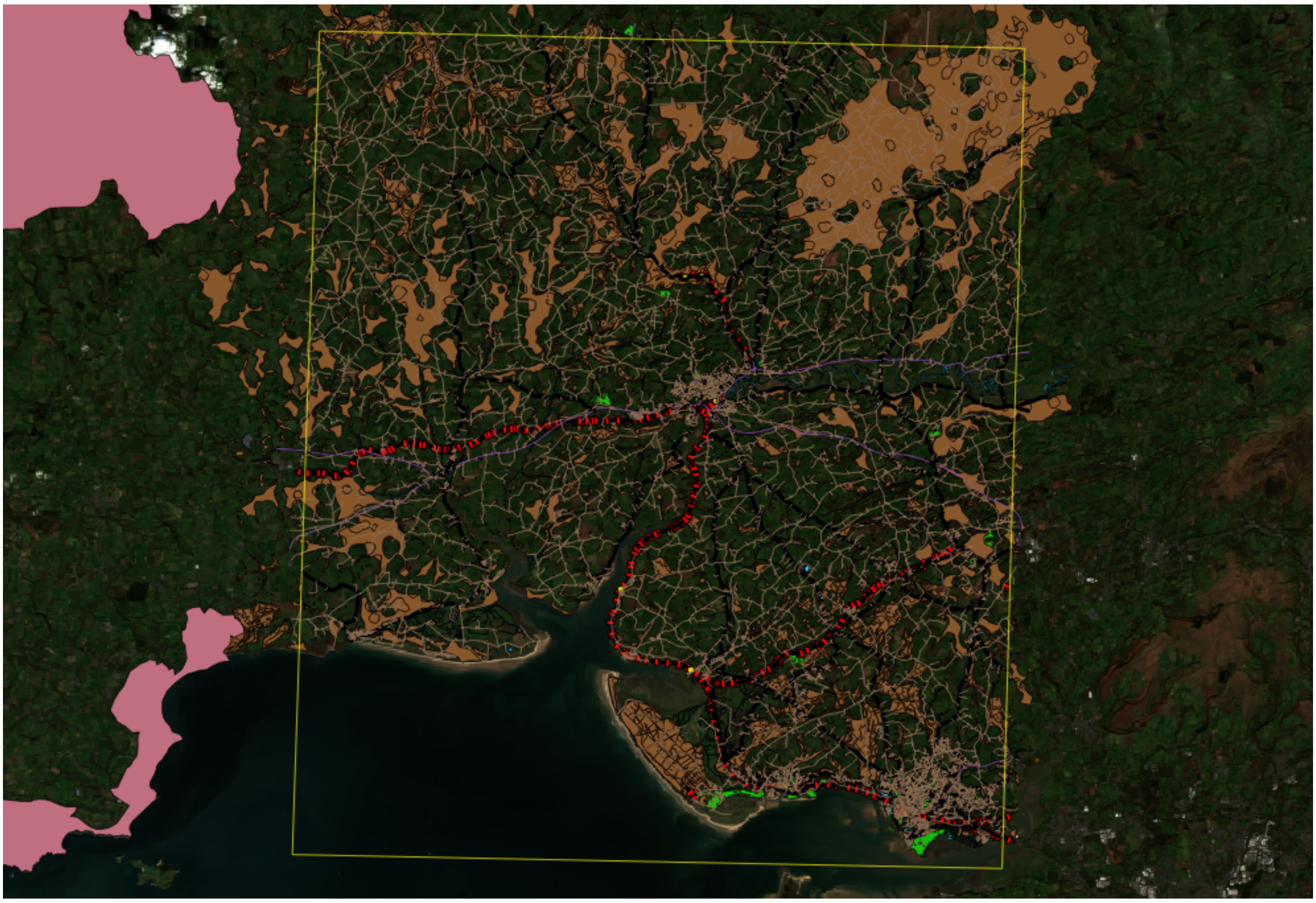

Figure 2. Area-of-Interest (Contains OS data © Crown copyright and database right 2019; Satellite image: ESA Copyright)

Whereas the Time-Of-Interest (TOI) was October 2018, the AOI had the polygonal bounds in World Geodetic System 1984 (WGS84) coordinates:

```
-4.09247619415462,51.6507504017036
-4.59606172257991,51.6468710002251
-4.59750580025958,52.0105126182078
-4.09303085864973,52.0127870676365
-4.09247619415462,51.6507504017036
```

6.3. What was provided

6.3.1. Supporting Datasets

The following datasets were identified as relevant to the scenario, and thus recommended for testing implementations of the specifications.

- ESA Sentinel Data: The Sentinels are a family of missions developed by ESA for Corpenicus, the European Union's Earth Observation programme. The data supplied to the OGC API Hackathon included imagery from the Sentinel-2 mission. Launched on 23 June 2015, the Sentinel-2 mission is a polar-orbiting, multispectral high-resolution imaging mission for land monitoring to support emergency services, imagery of vegetation, soil and water cover, inland waterways and coastal areas [6]. The Sentinel imagery was supplied by Sinergise, the providers of sentinelhub.com [7].
- UK Met Office DataPoint: DataPoint is a freely available service that offers meteorological feeds for use by professionals, the scientific community, and developers. It is an unsupported service, with a primary goal of facilitating research, development and innovation [8].
- UK Met Office Atmospheric Deterministic and Probabilistic Forecasts: This dataset includes atmospheric deterministic and probabilistic forecasts provided as downloadable gridded data [9]. The data includes 2km deterministic high-resolution atmospheric data for the UK and 10km deterministic high-resolution atmospheric data for the Globe. There is also data from the Global and Regional Ensemble Prediction System.
- Ordnance Survey - OS Open Zoomstack: This dataset provides a single, customisable map of Great Britain to be used at national and local levels. The dataset is available in OGC GeoPackage format. The dataset includes vector data at a variety of scales, from a whole-country view to a street-level view (1:10,000) [10].
- Meteorological Service of Canada Datamart: A variety of raw meteorological data types and forecast data provided by the Meteorological Service of Canada (MSC). It is aimed at specialized users with good meteorological and Information Technology knowledge. The datasets are available through direct download from an HTTP server, as well as through a Web Map Service (WMS) [11].

6.3.2. Supporting Services

The following datasets were identified as relevant to the scenario, and thus recommended for testing implementations of the specifications.

- Meteorological Service of Canada Geospatial web services: This service provides access to the MSC's open data, including raw numerical weather prediction (NWP) model data layers and the weather radar mosaic. The service provides meteorological layers through a Web Map Service (WMS) interface to enable end-users to display meteorological data within their own tools, on interactive web maps and in mobile apps [12].

- National Oceanic and Atmospheric Administration (NOAA) National Weather Service Data as OGC Web Services: These web services provide meteorological data covering the United States, through interfaces that conform to the Web Map Service (WMS), Web Feature Service (WFS) and Web Coverage Service (WCS) standards of the OGC [13].

6.3.3. Deployment Infrastructure

Participants were advised to bring their own laptops to the hackathon. To support testing, the following infrastructure options were available to participants:

- Participants could deploy services into their own computers.
- Participants could deploy services into their own Cloud infrastructure.
- By prior arrangement, participants could deploy services into Ordnance Survey-sponsored Cloud infrastructure.

6.4. Hackathon Participants

NOTE	This list will be updated at the start of the Hackathon

The Hackathon was sponsored by the European Space Agency (ESA) and the Ordnance Survey.

The following organizations participated in the Hackathon:

- 52°North GmbH
- akouas
- ARC
- Arup
- blockdore
- Board Adviser
- British Antarctic Survey
- Cicy
- CREAF
- CubeWerx Inc.
- Deimos Space UK
- developer
- District Government Cologne - Geobasis NRW
- Defence Science and Technology Laboratory (Dstl)
- Duisburg Essen university

- Ecere Corporation
- ECMWF
- El Toro
- EOS Data Analytics
- EOX IT Services GmbH
- Esri UK
- Eurac Research
- European Space Agency (ESA)
- Geobeyond
- GeoCat B.V.
- GeoLabs
- GeoSeer
- GeoSolutions
- Geovation
- Heazeltech
- Helyx SIS
- Hexagon
- Infinity Corporation Limited
- interactive instruments GmbH
- ISRIC - World Soil Information
- Jet Propulsion Laboratory (JPL)
- JRC, European Commission
- Land Information New Zealand
- Landcare Research, New Zealand
- lat/lon GmbH
- Met Office
- Meteorological Service of Canada
- National Aeronautics and Space Administration (NASA)
- National Geospatial Intelligence Agency (NGA)
- National Land Survey of Finland
- Natural Resources Canada
- NOAA/NWS
- Open Geospatial Consortium
- Ordnance Survey
- OSGeo

- Princeton University
- Princeton University Library
- Quick Caption
- rasdaman GmbH
- Secure Dimensions
- Sigma Bravo
- Simms Reeve
- Sinergise
- Solenix
- Spacebel s.a.
- Strategic Alliance Consulting Inc
- University College London
- University of Birmingham
- University of Münster
- University of Notre Dame
- WebGeoDataVore
- West University of Timisoara

Chapter 7. Architecture

The focus of the hackathon was on development of the OGC API - Common, the OGC API - Features, OGC API – Processes, the OGC API – Coverages and the OGC API – Map Tiles standards. Implementations of these specifications were deployed in the Hackathon infrastructure in order to build a solution with the architecture shown in Figure 3. As shown on the illustration, the architecture adopted a multi-tier approach that included one or more implementations of each OGC API specification deployed on the wider Internet (e.g. in participants' own Cloud environments), as well as some implementations deployed in the Ordnance Survey's Cloud which is hosted on Microsoft Azure.

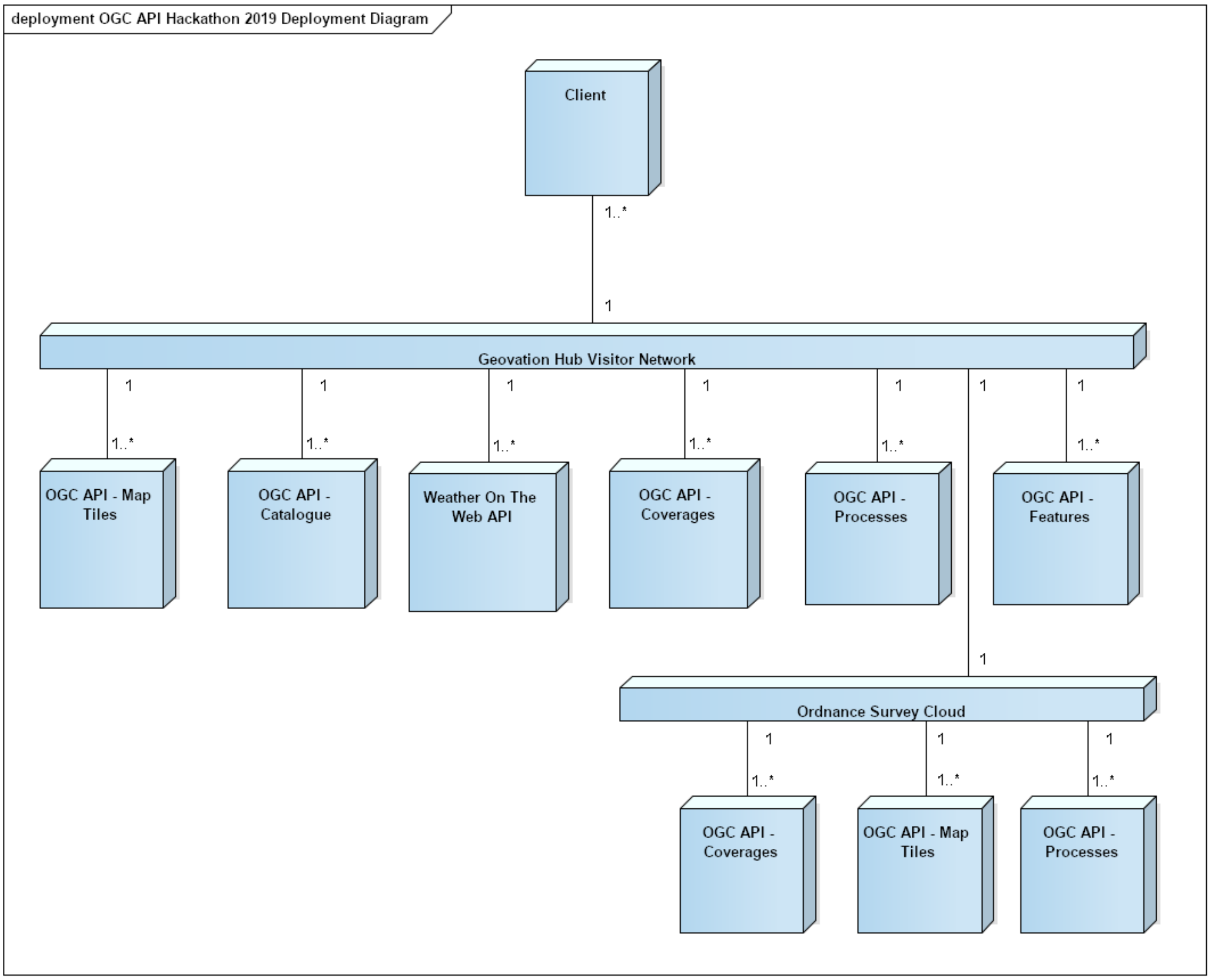

Figure 3. Solution Architecture of the OGC API Hackathon

The OGC API - Common specification 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. The OGC API - Processes specification defines how a client application can request the execution of a process, how the inputs to that process can be provided, and how the output from the process is handled. The OGC API - Map Tiles specification defines an OGC standard for a Web Map Tile API that can serve map tiles of spatially referenced data using tile images with predefined content, extent, and resolution. The OGC API - Coverages specification defines a Web API for accessing coverages that are modelled according to the Coverage Implementation Schema (CIS)

1.1 [http://docs.opengeospatial.org/is/09-146r6/09-146r6.html]. The hackathon also sought to maintain consistency between the aforementioned specifications and the OGC API - Features specification. The OGC API - Features specification offers the capability to create, modify and query geospatial feature data on the Web.

7.1. Service Implementations

7.1.1. pygeoapi

pygeoapi is a Python server implementation of the emerging OGC API specifications. Early versions of this software implemented the OGC API - Features specification (formerly WFS 3.0). Recent versions of the software have included support for both the OGC API - Features and the OGC API - Processes specifications. Support for these specifications makes it possible publish feature data and geospatial processes. As the name suggests the software is built using the Python programming language and is supported by a developer toolchain that includes Docker and Git/Github.

7.1.2. 52°North JavaPS

The 52°North JavaPS product is an open source implementation of both the Web Processing Service (WPS) standard and the OGC API - Processes specification. JavaPS enables the deployment of geospatial processes on the Web in a way that conforms to OGC standards. The software is built using the Java programming language and is supported by a developer toolchain that includes Maven and Git/Github. The software features a pluggable architecture for processes and data encodings that is based on the 52°North Iceland [https://wiki.52north.org/SensorWeb/Iceland] project which represents a generic Java framework for OGC Web Services. By virtue of being based on the Iceland project, JavaPS provides components that are associated with processing geospatial data, for example request objects, response objects, decoders, encoders, parsers.

7.1.3. Esri

Esri produce both an installable product (ArcGIS Enterprise) and SaaS product (ArcGIS Online) which support adopted OGC specifications. ArcGIS Enterprise as a server implements WMS, WMTS, WCS2, WPS, WFS2, KML, Geopackage etc as well as other de-facto standards. ArcGIS Online as a server implements WMTS, WFS2 as well as other de-facto standards. Until the OGC API standards are stable and formally adopted, it is not feasible to implement them in the released products. Therefore for the purposes of the Hackathon and further R&D, Esri have implemented the OGC-API tiles as a facade on to ArcGIS Online tiled services. Technically this is currently implemented as a node.js server running in Microsoft Azure. https://ogc-tiles-esri-server.azurewebsites.net

7.1.4. rasdaman

rasdaman ("raster data manager") is a flexible, scalable datacube engine with location-transparent federation capabilities. Open-source rasdaman is available from www.rasdaman.org. Being WCS reference implementation rasdaman supports OGC WMS, WCS, and WCPS (the OGC datacube analytics standard). For experimentation with the emerging OGC OpenAPI interface for coverages a rasdaman server has been made available on Amazon with an OpenAPI facade to WCS; access has been demonstrated with EURAC and Sinergise OpenAPI clients.

7.1.5. TBA

TBA

7.2. Client Implementations

7.2.1. OpenSphere OGC API Plugin

OpenSphere [https://github.com/ngageoint/opensphere] is a pluggable, single-page, GIS web application that supports both 2D and 3D views. It supports hooking up to many common servers and formats such as ArcGIS, Geoserver (and other OGC WMS/WFS services), XYZ, TMS, KML, GeoJSON, Shapefiles and CSVs. Other features include animation of both raster and vector data, import and export of various formats, and saving files and layers between sessions. Sigma Bravo extended OpenSphere to support OGC API - Features and OGC API - Map Tiles.

7.2.2. Hexagon LuciadLightspeed

LuciadLightspeed provides a foundation for advanced geospatial analytics applications. It allows users to create high performance command & control and location intelligence applications with clean design implementation and rapid application development. A desktop client application was implemented using LuciadLightspeed and configured to interface services implementing the OGC API - Map Tiles specification.

7.2.3. Solenix WPS Demo Client

The Solenix WPS Demo client is an adaptation of the OGC Testbed 14 client, accounting for some of the changes introduced with the OGC API - Processing.

The client application runs from a web browser.

7.2.4. Esri OGC API-Tiles Demo Client

The Esri client application is a simple Leaflet application which connects to the Esri OGC API-Tiles server implementation for testing purposes.

7.2.5. TBA

TBA

7.2.6. TBA

TBA

Chapter 8. OGC API Specifications

This chapter describes each of the OGC API specifications that were advanced by the Hackathon. The section presents an overview of each specification and is not intended to be a substitute for reading the complete specification.

8.1. OGC API - Common

The OGC API - Common specification 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. As such, the OGC API - Common standard serves as the "OWS Common" standard for OGC Resource Oriented APIs. 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 [https://www.openapis.org/].

In addition to identifying core resources, the OGC API - Common specification goes on to specify HTTP status codes that may be supported by an OGC API, as well as how to handle web caching, coordinate reference systems and encodings. The specification also describes how to handle common parameters such as bounding boxes and date-time constraints.

The following subsection presents a summary of the core resources.

8.1.1. Key Resources

A summary of the paths offered by the OGC API - Common specification is presented below:

- Path = /
  - Returns landing page
- Path = /api
  - Returns API Description document (OpenAPI)
- Path = /conformance
  - Returns a set of conformance class URIs.
- Path = /collections
  - Returns metadata describing the collections accessible through this API
- Path = /collections/{collectionId}
  - Returns metadata describing the collection identified by {collectionId}
- Path = /collections/{collectionId}/items
  - Returns --- TBD. This may be where Common leaves off and resource specific standards take over.

8.2. OGC API - Features

The OGC API - Features specification offers the capability to create, modify and query spatial data on the Web. This standard specifies requirements and recommendations for APIs that want to follow a standard way of sharing feature data. The specification is a multi-part document. The Core part of the specification describes the mandatory capabilities that every implementing service has to support and is restricted to read-access to spatial data. 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, richer queries, and additional coordinate reference systems. This specification builds on the Web Feature Service (WFS) standard and has previously been referred to as WFS 3.0.

8.2.1. Key Resources

A summary of the paths offered by the OGC API - Features specification is presented below:

- Path = /
  - Returns the landing page of this API (inherited from OGC API - Common)
- Path = /conformance
  - Returns information about standards that this API conforms to (inherited from OGC API - Common)
- Path = /collections
  - Returns a description of the feature collections in the dataset (inherited from OGC API - Common)
- Path = /collections/{collectionId}
  - Returns a description of the {collectionId} feature collection (inherited from OGC API - Common)
- Path = /collections/{collectionId}/items
  - Returns features of feature collection {collectionId}
- Path = /collections/{collectionId}/items/{featureId}
  - Returns a feature; using content negotiation to request HTML, GeoJSON or other

8.3. OGC API - Processes

The OGC API - Processes specification provides defines how a client application can request the execution of a process, how the inputs to that process can be provided, and how the output from the process is handled. The specification allows for the wrapping of computational tasks into an executable process that can be invoked by a client application. Examples of computational processes that can be supported by implementations of this specification include raster algebra, geometry buffering, constructive area geometry, routing and several others. This specification builds on the Web Processing Service (WPS) standard.

8.3.1. Key Resources

A summary of the paths offered by the OGC API - Processes specification is presented below:

- Path = /
  - Returns landing page (inherited from OGC API - Common)
- Path = /api
  - Returns API Description document (OpenAPI) (inherited from OGC API - Common)
- Path = /conformance
  - Returns a set of conformance class URIs. (inherited from OGC API - Common)
- Path = /processes
  - Returns available processes
- Path = /processes/{id}/
  - Returns a process description
- Path = /processes/{id}/jobs
  - Returns the list of jobs for a process.
- Path = /processes/{id}/jobs/{jobID}
  - Returns the status of a job
- Path = /processes/{id}/jobs/{jobID}/result
  - Returns the result(s) of a job

8.4. OGC API - Map Tiles

The OGC API - Map Tiles specification defines an OGC standard for a Web Map Tile API that can serve map tiles of spatially referenced data using tile images with predefined content, extent, and resolution. The specification describes the discovery and query operations of an API that provides access to Map Tiles in a manner independent of the underlying data store. The discovery operations allow the API to be interrogated to determine its capabilities and retrieve metadata about the organisation and distribution of tiles. The query operations allow tiles to be retrieved from the underlying data store based upon simple selection criteria, defined by the client. This specification builds on the Web Map Tile Service (WMTS) standard.

8.4.1. Key Resources

A summary of the paths offered by the OGC API - Processes specification is presented below:

- Path = /
  - Returns landing page (inherited from OGC API - Common)
- Path = /conformance
  - Returns a set of conformance class URIs. (inherited from OGC API - Common)
- Path = /collections

    - Returns metadata describing the collections accessible through this API (inherited from OGC API - Common)
- Path = /collections/{collectionId}
    - Returns metadata describing the collection identified by {collectionId}
- Path = /collections/{collectionId}/queryables
    - Returns the queryable properties of the feature collection
- Path = /collections/{collectionId}/items
    - Returns features of the feature collection
- Path = /collections/{collectionId}/items/{featureId}
    - Returns a feature
- Path = /tileMatrixSet
    - Returns all available tile matrix sets (tiling schemes)
- Path = /tileMatrixSet/{tileMatrixSetId}
    - Returns a tiling scheme by id
- Path = /tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}
    - Returns a tile of the dataset
- Path = /collections/{collectionId}/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}
    - Returns a tile of the collection with or without style
- Path = /tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}/info
    - Returns information on a point of a tile with or without style
- Path = /collections/{collectionId}/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}/info
    - Returns information of a point in a tile of the collection with or without style
- Path = /tiles/{tileMatrixSetId}
    - Returns tiles from several collections.
- Path = /collections/{collectionId}/tiles/{tileMatrixSetId}
    - Returns tiles of a collection
- Path = /map
    - Returns a map of collections with or without style
- Path = /collections/{collectionId}/map
    - Returns a maps from the collection with or without style
- Path = /map/info
    - Returns information about a map of the collection with or without style
- Path = /collections/{collectionId}/map/info
    - Returns information about a map from the collection with or without style

8.5. OGC API - Coverages

The OGC API - Coverages specification defines a Web API for accessing coverages that are modelled 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 modelled as a gridded coverage, for example. The OGC API - Coverages specification builds on the Web Coverage Service (WCS) standard.

8.5.1. Key Resources

A summary of the paths offered by the OGC API - Coverages specification is presented below:

- Path = /
  - Returns landing page (inherited from OGC API - Common)
- Path = /api
  - Returns API Description document (OpenAPI) (inherited from OGC API - Common)
- Path = /conformance
  - Returns a set of conformance class URIs. (inherited from OGC API - Common)
- Path = /collections
  - Returns metadata describing the collections accessible through this API (inherited from OGC API - Common)
- Path = /collections/{collectionId}
  - Returns metadata describing the collection identified by {collectionId}
- Path = /collections/{collectionId}/coverages
  - Returns metadata about each coverage in the collection
- Path = /collections/{collectionId}/coverages/{coverageID}
  - Returns the coverage itself. Typically as an image file.
- Path = /collections/{collectionId}/coverages/{coverageID}/metadata
  - Returns metadata about a coverage.
- Path = /collections/{collectionId}/coverages/{coverageID}/domainset
  - Returns a description of the domain set of the coverage
- Path = /collections/{collectionId}/coverages/{coverageID}/rangetype
  - Returns a description of the range type of the coverage

Chapter 9. Key Findings

This section presents the key findings from the Hackathon.

9.1. Results

9.1.1. OGC API - Processes

The participants made an observation that some attributes for referenceValue were missing. As a result, changes were made to add attributes for identifying the MIME type, schema and encoding in a referenceValue object. These changes were needed for providing input in a specific format or returning output in a specific format.

There was a suggestion [https://github.com/opengeospatial/wps-rest-binding/issues/42] to add ows:additionalProperties and ows:context to metadata. It was observed that ows:additionalParameters allows a service to provide key value pairs metadata information. It was also observed that Testbed 13 and testbed 14 had demonstrated the utility of such a capability. As a result, the hackathon participants committed changes to the OGC API - Process specification adding the definition of additionalParameters.

There was a discussion [https://github.com/opengeospatial/wps-rest-binding/issues/37] about whether process output arrays are fully supported. The participants observed that currently the output-value-cardinality [http://www.opengis.net/spec/WPS/2.0/req/conceptual-model/process/output-value-cardinality] requirement limits cardinality of the output to one. A number of workarounds were suggested including returning a list of outputs, returning an archive (zip), returning a Metalink, or returning Multipart/mixed responses. The participants concluded that a solution would need to handle all kind of outputs, including outputs by reference, as well as binary files.

There was a discussion [https://github.com/opengeospatial/wps-rest-binding/issues/30] about how to specify synchronous execution OGC API Hackathon. The options identified included i) Either use a query parameter or a HTTP header to specify the execution mode, ii) Return a result object, iii) Do not return a header with the location (as there technically is no location, i.e. its a temp. This issue was also related to the broader issue of how to choose between synchronous and asynchronous mode for job creation. The participants observed that indeed the WPS 2.0 specification has a jobID in the the status object, and therefore the OGC API - Processes specification should also include a jobID into the statusInfo object.

There was a suggestion [https://github.com/opengeospatial/wps-rest-binding/issues/31] for an extension supporting triggers according to job status. In particular the suggestion was for the user/client to be able to specify triggers for conditions like: `onSuccess` a Url to be triggered upon process completion, or `onFail` a URL to be triggered on process failure, `progressUpdate` a URL to be triggered by the job while progressing and should contain progress status (eg. proccent of job completion). Participants considered whether such a capability might be more appropriate as an extension, perhaps associated with a pub/sub notification capability.

There was a suggestion [https://github.com/opengeospatial/wps-rest-binding/issues/32] to add the exception information to the status information at GET `/processes/{id}/jobs/{jobID}` and remove the GET `/processes/{id}/jobs/{jobID}/exception` endpoint. The participants considered whether an HTTP

201 code should be returned with POST /processes/{id}/jobs. The participants resolved that allowing for plural form results could address this issue, for example `processes/{processId}/jobs/{jobId}/results/{resultId}`. Use of an HTTP 201 code was however inconclusive.

9.1.2. OGC API - Map Tiles

The participants identified three roots for API building blocks, namely the root of the service, collection ID, and collection ID combined the coverage ID. A need to combine these root paths with maps and tiles was identified. There was an observation made that if the roots are combined with maps and tiles, there is an opportunity to provide much more information through data tiles such as vector tiles or coverage tiles. There was also an observation made that if a style and style id are concatenated with the path then the API would be able to combine data tiles with portrayals of the information. There was a third alternative identified, which is the concatenation of both the aforementioned approaches.

These paths have specific query parameters associated with them. Participants observed that such a capability may not be fully supported by OpenAPI 3.0.

There was a discussion about whether the response for maps should return raster or vector maps. Related to this was whether a map was at the same level as a collection. There was agreement that maps are not at the same level as collections.

9.1.3. OGC API - Coverages

The participants agreed that OGC API - Coverages should inherit from OGC API - Common as much as possible. This meant that wherever a requirement is specified in OGC API - Common, if it is relevant to coverages, the OGC API - Coverages specification would reference the requirement in the OGC API - Common specification.

There was also agreement that a request for the coverages path would return a list of all of the coverage identifiers included in the collection. Regarding coverage descriptions, it was agreed that a request for the coverage description would only return the essential information instead of the complete metadata associated with the coverage.

There was discussion about how to support bounding box (BBOX) filters on multidimensional coverages. The participants expressed the need to inherit the BBOX and time parameters from OGC API - Common, however also they also observed that there would be a need to identify a CRS for height. One of the suggestions was for each axis to have a separate coverage filter. The participants concluded that there is currently no construct in the Core part of the OGC API - Common specification that supports filtering of coverages.

The was discussion about the retrieval of coverages. The OGC API - Coverages specification was updated to allow for different ways for getting the coverages individually. Since not every format is suitable for transferring all of the coverage information, participants identified different ways for getting the different types of coverages. It was also noted that for applications that do not want to use collections, they can just use the `coverages/{coverageid}` path.

There was a discussion about whether parameters, values and URL bases were case sensitive. This issue was observed to be applicable to all of the specifications. There was a suggestion that the OGC API - Common specification should specify a rule for case sensitivity and that that rule should be consistent with the RFC.

9.1.4. OGC API - Common

There was a discussion about whether OGC API - Common should support the CRS:84 coordinate system (WGS84) by default. The participants observed that the collectionInfo metadata (returned for each collection) allows one to specify the CRS supported by the collection. The client can specify one of the other CRS if they do not support the default. For coverages, the default CRS was observed to be the native CRS. The participants concluded that there will be a default CRS for the API and the OGC API - Common specification should have complete control over the CRS and the default CRS should not constrain the resource.

The participants observed that there is a need for something like CRS:84 that has ellipsoid height. EPSG:4327 was suggested as a possible basis for such a CRS, with the caveat that it has been deprecated. It was agreed to discuss the issue of a height or elevation CRS with the WFS SWG and the CRS DWG at the OGC TC meeting in Leuven.

9.1.5. Technology Integration Experiments (TIE)

The following table presents Technology Integration Experiments (TIE) that were completed during the Hackathon.

Table 2. Technology Integration Experiments (TIE) for OGC APIs

Services\ Client	Hexagon	Helyx	OpenSphere	Esri	Solenix	EURAC	Sinergise
52 North		Processes			Processes		
CubeWerx		Processes			Processes		
Esri	Map Tiles		Map Tiles	Map Tiles			
Helyx		Processes					
pygeoapi			Features		Processes		
Geoserver			Features				
Spacebel			Features		Processes		
West University of Timisoara					Processes		
rasdaman						Coverages	Coverages

NOTE Services on rows and Clients on columns

9.2. What occurred

9.2.1. Processes

The decision to hold the OGC API Hackathon was made by the TC at the February 2019 TC meeting in Singapore. Following this decision, OGC staff engaged a number of potential sponsors from the OGC membership. Having identified sponsors and hosts, a series of teleconferences were held for planning the event. These teleconferences discussed venue logistics, computing infrastructure, data, scenarios, catering and other topics. A Gantt chart of the planning and execution of the hackathon is shown in Figure 4.

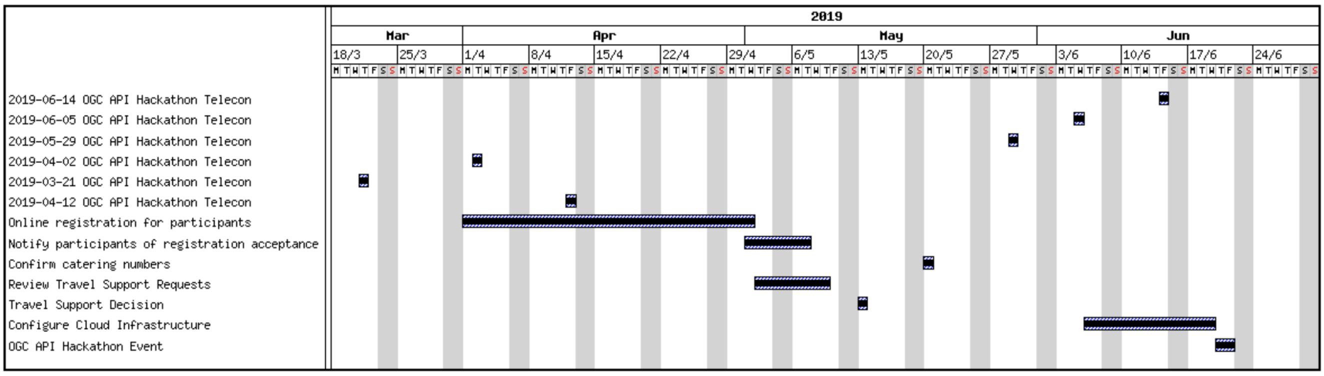

Figure 4. A Gantt chart of the planning and execution of the hackathon

During the hackathon, the process involved a series of briefings, discussions and coding sessions. On the first day of the hackathon, three back-briefs were held, that is one in the morning, another in the afternoon and another in the evening. These briefings provided an opportunity for issues to be discussed across teams. Agreements and resolutions from the discussions triggered by the briefings were then fed back into the team-specific work.

9.2.2. Organization

By the event date, 76 individuals had been registered to participate in-person and 35 participants had been registered to participate remotely. A questionnaire sent out just before the hackathon to collect information about which OGC API specifications participants planned to focus on received 27 responses, with the spread of interest as shown in Figure 5, Figure 6 and Figure 7.

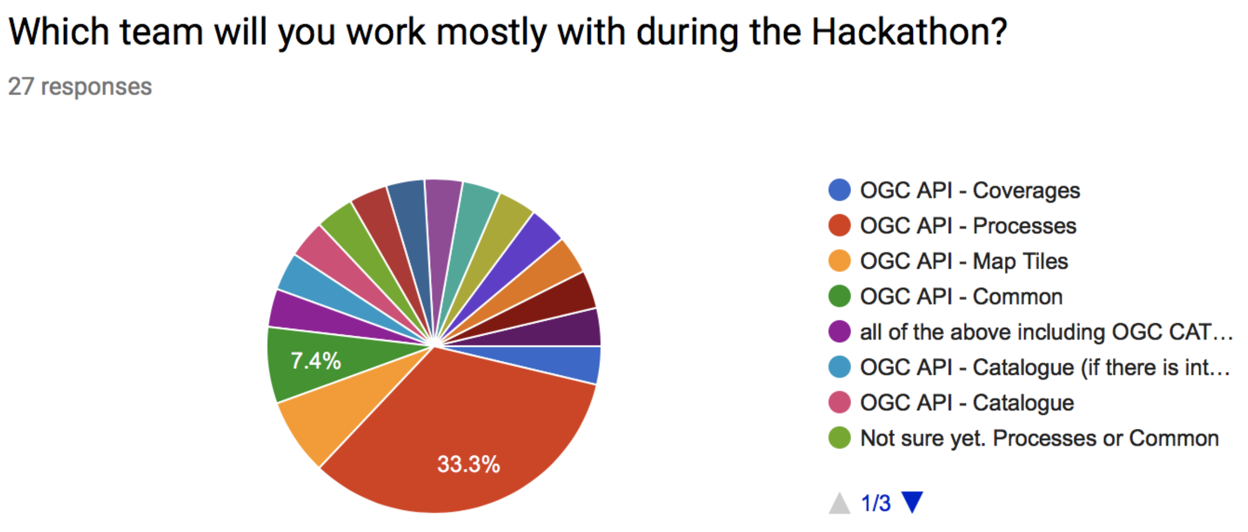

Figure 5. Participants' interests (1 of 3)

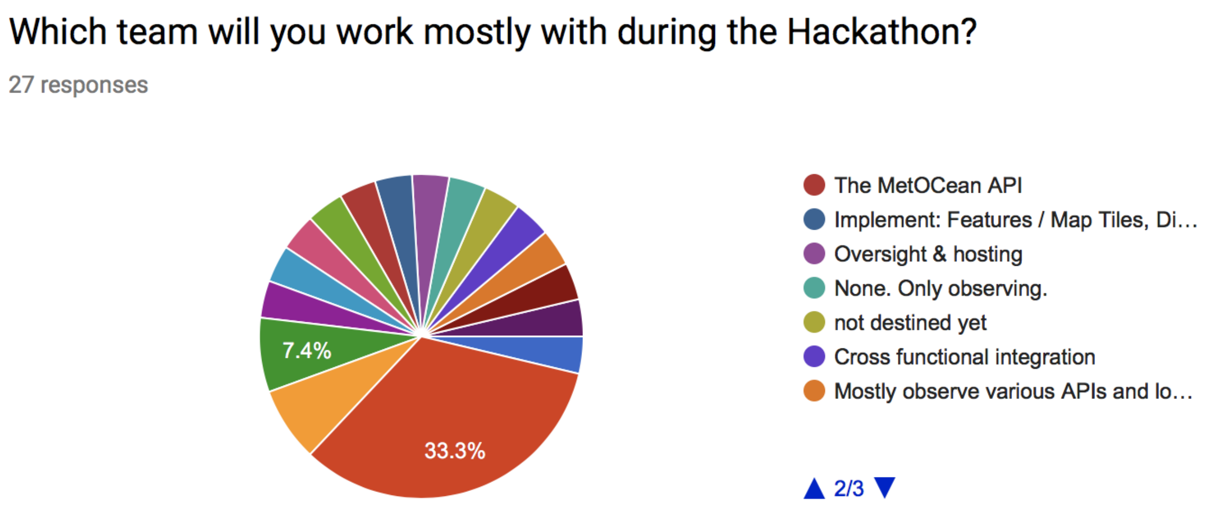

Figure 6. Participants' interests (2 of 3)

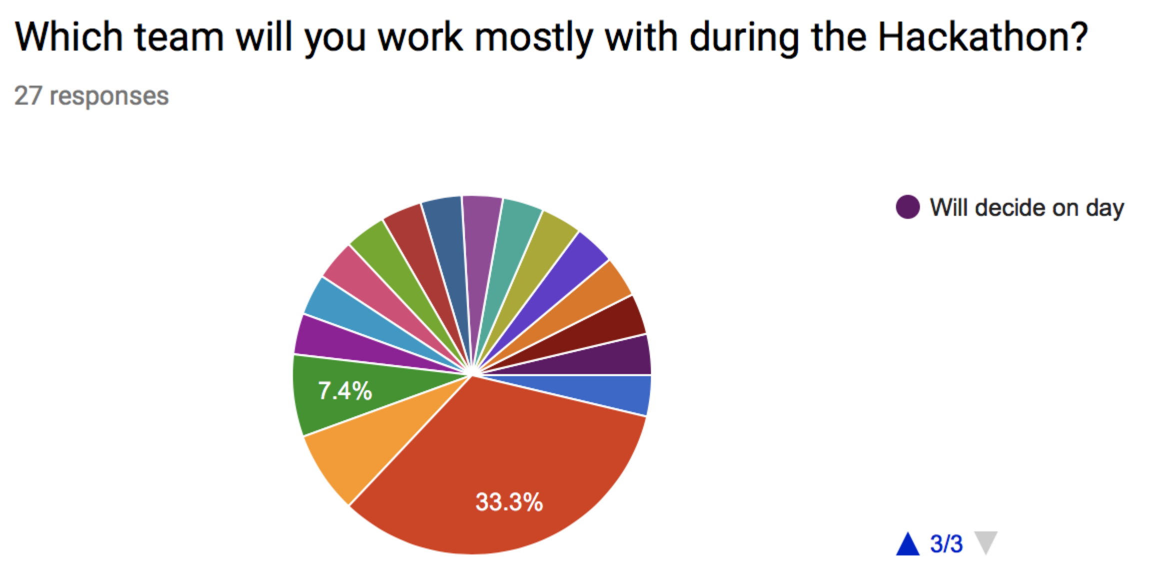

Figure 7. Participants' interests (3 of 3)

The hackathon was therefore organized around teams based on the OGC API specifications. Participants interested in APIs other than those for coverages, processes, and map tiles, were asked to contribute to the work on advancing the OGC API - Common specification. This would help ensure that the OGC API - Common specification provides an appropriate base for all future OGC APIs.

9.2.3. Technology

At least 7 client applications were deployed during the hackathon. The client applications included a mixture of standalone and web browser based applications. At least 9 server-side applications were deployed. Some of the programming languages that were used by these applications included Python, C++, Java and NodeJS.

Some of the participants implemented proxies in front of implementations of WCS, WMTS and WPS to provide an adapter for OGC API applications.

It was observed that, although the OGC API specifications do not prescribe where an API definition should be hosted, Swaggerhub proved a popular choice amongst hackathon participants.

9.2.4. Information

9.3. Experiences

TBA

9.4. Lessons learnt

TBA

9.5. What are the next steps?

TBA

Appendix A: Implementations of OGC APIs

A.1. 52°North GmbH

TBA

A.1.1. Motivation to Participate

TBA

A.1.2. Implemented Solution

52°North implemented and deployed an instance of the 52°North JavaPS software, which is an implementation of the OGC Web Processing Service (WPS) standard and the OGC API - Processes specification. A screenshot of the deployed service is shown in Figure 8. The software was configured to offer an interface that conforms to the OGC API - Processes specification.

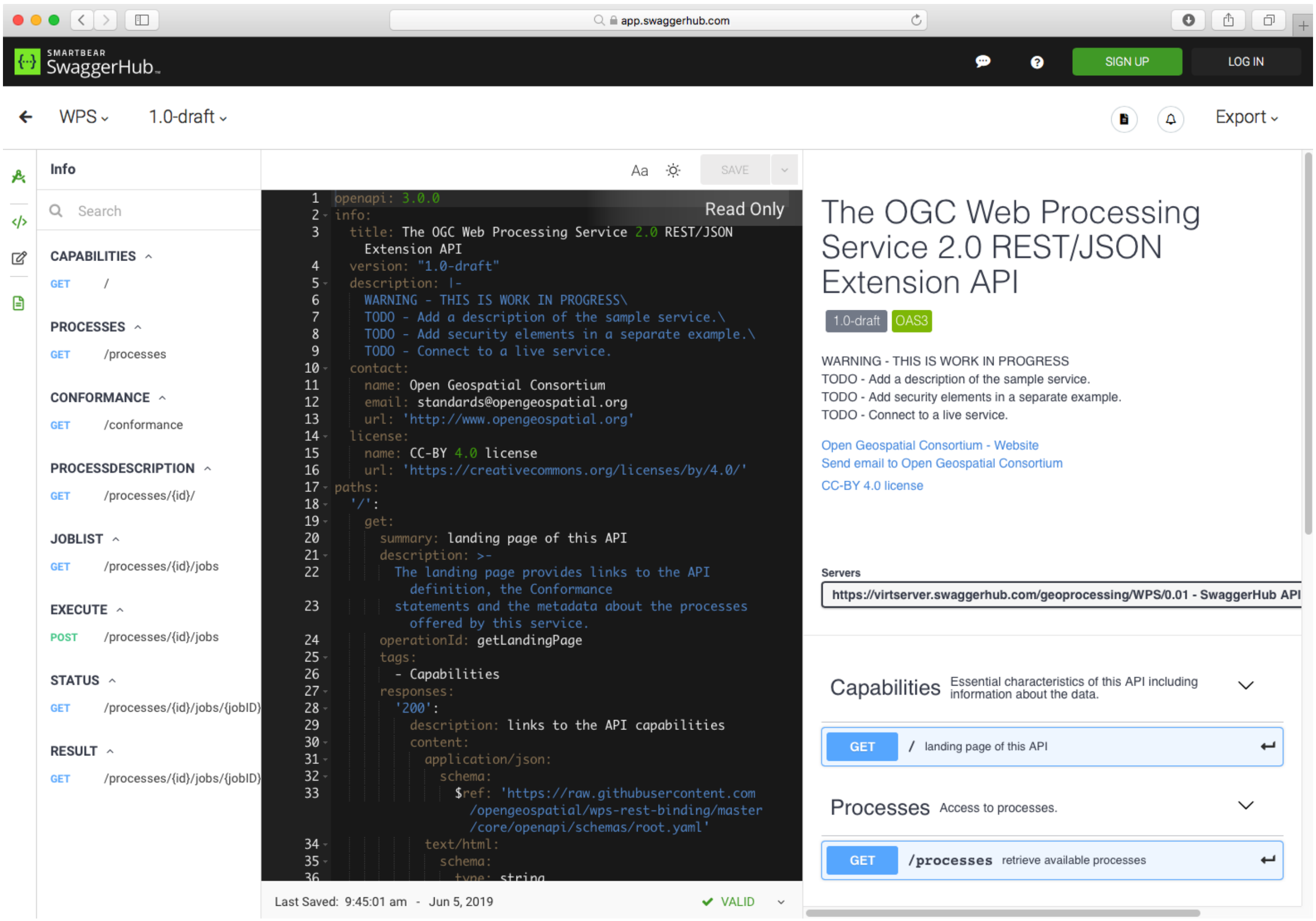

Figure 8. Swagger Hub page of the OGC API - Processes implementation deployed by 52°North GmbH

A.1.3. Proposed Alternatives

TBA

A.1.4. Experiences with OGC API Specifications

TBA

A.1.5. Other Impressions & Recommendations

TBA

A.2. akouas

TBA

A.2.1. Motivation to Participate

TBA

A.2.2. Implemented Solution

TBA

A.2.3. Proposed Alternatives

TBA

A.2.4. Experiences with OGC API Specifications

TBA

A.2.5. Other Impressions & Recommendations

TBA

A.3. ARC

TBA

A.3.1. Motivation to Participate

TBA

A.3.2. Implemented Solution

TBA

A.3.3. Proposed Alternatives

TBA

A.3.4. Experiences with OGC API Specifications

TBA

A.3.5. Other Impressions & Recommendations

TBA

A.4. Arup

TBA

A.4.1. Motivation to Participate

TBA

A.4.2. Implemented Solution

TBA

A.4.3. Proposed Alternatives

TBA

A.4.4. Experiences with OGC API Specifications

TBA

A.4.5. Other Impressions & Recommendations

TBA

A.5. blockdore

TBA

A.5.1. Motivation to Participate

TBA

A.5.2. Implemented Solution

TBA

A.5.3. Proposed Alternatives

TBA

A.5.4. Experiences with OGC API Specifications

TBA

A.5.5. Other Impressions & Recommendations

TBA

A.6. British Antarctic Survey

TBA

A.6.1. Motivation to Participate

TBA

A.6.2. Implemented Solution

TBA

A.6.3. Proposed Alternatives

TBA

A.6.4. Experiences with OGC API Specifications

TBA

A.6.5. Other Impressions & Recommendations

TBA

A.7. Cicy

TBA

A.7.1. Motivation to Participate

TBA

A.7.2. Implemented Solution

TBA

A.7.3. Proposed Alternatives

TBA

A.7.4. Experiences with OGC API Specifications

TBA

A.7.5. Other Impressions & Recommendations

TBA

A.8. CREAF

TBA

A.8.1. Motivation to Participate

TBA

A.8.2. Implemented Solution

TBA

A.8.3. Proposed Alternatives

TBA

A.8.4. Experiences with OGC API Specifications

TBA

A.8.5. Other Impressions & Recommendations

TBA

A.9. CubeWerx Inc.

TBA

A.9.1. Motivation to Participate

TBA

A.9.2. Implemented Solution

TBA

A.9.3. Proposed Alternatives

TBA

A.9.4. Experiences with OGC API Specifications

TBA

A.9.5. Other Impressions & Recommendations

TBA

A.10. Deimos Space UK

TBA

A.10.1. Motivation to Participate

TBA

A.10.2. Implemented Solution

TBA

A.10.3. Proposed Alternatives

TBA

A.10.4. Experiences with OGC API Specifications

TBA

A.10.5. Other Impressions & Recommendations

TBA

A.11. District Government Cologne - Geobasis NRW

TBA

A.11.1. Motivation to Participate

TBA

A.11.2. Implemented Solution

TBA

A.11.3. Proposed Alternatives

TBA

A.11.4. Experiences with OGC API Specifications

TBA

A.11.5. Other Impressions & Recommendations

TBA

A.12. Dstl

TBA

A.12.1. Motivation to Participate

TBA

A.12.2. Implemented Solution

TBA

A.12.3. Proposed Alternatives

TBA

A.12.4. Experiences with OGC API Specifications

TBA

A.12.5. Other Impressions & Recommendations

TBA

A.13. Duisburg Essen university

TBA

A.13.1. Motivation to Participate

TBA

A.13.2. Implemented Solution

TBA

A.13.3. Proposed Alternatives

TBA

A.13.4. Experiences with OGC API Specifications

TBA

A.13.5. Other Impressions & Recommendations

TBA

A.14. Ecere Corporation

TBA

A.14.1. Motivation to Participate

TBA

A.14.2. Implemented Solution

TBA

A.14.3. Proposed Alternatives

TBA

A.14.4. Experiences with OGC API Specifications

TBA

A.14.5. Other Impressions & Recommendations

TBA

A.15. ECMWF

TBA

A.15.1. Motivation to Participate

TBA

A.15.2. Implemented Solution

TBA

A.15.3. Proposed Alternatives

TBA

A.15.4. Experiences with OGC API Specifications

TBA

A.15.5. Other Impressions & Recommendations

TBA

A.16. El Toro

TBA

A.16.1. Motivation to Participate

TBA

A.16.2. Implemented Solution

TBA

A.16.3. Proposed Alternatives

TBA

A.16.4. Experiences with OGC API Specifications

TBA

A.16.5. Other Impressions & Recommendations

TBA

A.17. EOS Data Analytics

TBA

A.17.1. Motivation to Participate

TBA

A.17.2. Implemented Solution

TBA

A.17.3. Proposed Alternatives

TBA

A.17.4. Experiences with OGC API Specifications

TBA

A.17.5. Other Impressions & Recommendations

TBA

A.18. EOX IT Services GmbH

TBA

A.18.1. Motivation to Participate

TBA

A.18.2. Implemented Solution

TBA

A.18.3. Proposed Alternatives

TBA

A.18.4. Experiences with OGC API Specifications

TBA

A.18.5. Other Impressions & Recommendations

TBA

A.19. European Space Agency (ESA)

TBA

A.19.1. Motivation to Participate

TBA

A.19.2. Implemented Solution

TBA

A.19.3. Proposed Alternatives

TBA

A.19.4. Experiences with OGC API Specifications

TBA

A.19.5. Other Impressions & Recommendations

TBA

A.20. Esri UK

Esri make ArcGIS mapping and analytics software.

A.20.1. Motivation to Participate

Evolution of the OGC specifications to a modern, developer-friendly API is essential.

A.20.2. Implemented Solution

To date we have not implemented any of the new draft OGC API standards in our released software. We have been working on various R&D prototypes of server and client.

A.20.3. Proposed Alternatives

TBA

A.20.4. Experiences with OGC API Specifications

Esri has been involved in the OGC since its inception, and have widely implemented many other OGC standards in both server and client software.

A.20.5. Other Impressions & Recommendations

TBA

A.21. Eurac Research

TBA

A.21.1. Motivation to Participate

TBA

A.21.2. Implemented Solution

TBA

A.21.3. Proposed Alternatives

TBA

A.21.4. Experiences with OGC API Specifications

TBA

A.21.5. Other Impressions & Recommendations

TBA

A.22. Geobeyond

Participant: Francesco Bartoli

A.22.1. Motivation to Participate

- Port the GEE-Bridge API to OGC API - Commons and OGC API - Processes.
- Discuss with the community how to handle WPS remote processes and process chaining (local and remote)
- Understand and find the way how to implement several specifications together for the new GeoNode API
- Align with the CSW community about the new OpenAPI based specification

A.22.2. Implemented Solution

- GeoNode API draft work of OGC API - Processes which would implement a generic solution for algorithms as functions

A.22.3. Proposed Alternatives

- Process chaining and consequently workflows should be part of the perimeter of the specification or can be an abstract concept within GeoNode?
- Processing from cloud infrastructure (Google Earth Engine, IBM Geoscope, etc) have to be considered remote or local? Sort of WPS Cascading?

A.22.4. Experiences with OGC API Specifications

- Basic experience with OGC API - Features
- Basic experience with OGC API - Commons and OGC API - Processes

A.22.5. Other Impressions & Recommendations

TBA

A.23. GeoCat B.V.

TBA

A.23.1. Motivation to Participate

TBA

A.23.2. Implemented Solution

TBA

A.23.3. Proposed Alternatives

TBA

A.23.4. Experiences with OGC API Specifications

TBA

A.23.5. Other Impressions & Recommendations

TBA

A.24. GeoLabs

TBA

A.24.1. Motivation to Participate

TBA

A.24.2. Implemented Solution

TBA

A.24.3. Proposed Alternatives

TBA

A.24.4. Experiences with OGC API Specifications

TBA

A.24.5. Other Impressions & Recommendations

TBA

A.25. GeoSeer

TBA

A.25.1. Motivation to Participate

TBA

A.25.2. Implemented Solution

TBA

A.25.3. Proposed Alternatives

TBA

A.25.4. Experiences with OGC API Specifications

TBA

A.25.5. Other Impressions & Recommendations

TBA

A.26. GeoSolutions

TBA

A.26.1. Motivation to Participate

TBA

A.26.2. Implemented Solution

TBA

A.26.3. Proposed Alternatives

TBA

A.26.4. Experiences with OGC API Specifications

TBA

A.26.5. Other Impressions & Recommendations

TBA

A.27. Geovation

TBA

A.27.1. Motivation to Participate

TBA

A.27.2. Implemented Solution

TBA

A.27.3. Proposed Alternatives

TBA

A.27.4. Experiences with OGC API Specifications

TBA

A.27.5. Other Impressions & Recommendations

TBA

A.28. Heazeltech

TBA

A.28.1. Motivation to Participate

TBA

A.28.2. Implemented Solution

TBA

A.28.3. Proposed Alternatives

TBA

A.28.4. Experiences with OGC API Specifications

TBA

A.28.5. Other Impressions & Recommendations

TBA

A.29. Helyx SIS

TBA

A.29.1. Motivation to Participate

TBA

A.29.2. Implemented Solution

TBA

A.29.3. Proposed Alternatives

TBA

A.29.4. Experiences with OGC API Specifications

TBA

A.29.5. Other Impressions & Recommendations

TBA

A.30. Hexagon

Hexagon is a global leader in sensor, software and autonomous solutions. We are putting data to work to boost efficiency, productivity, and quality across industrial, manufacturing, infrastructure, safety, and mobility applications. Our technologies are shaping urban and production ecosystems to become increasingly connected and autonomous — ensuring a scalable, sustainable future.

Within Hexagon Geospatial, we create solutions that visualize location intelligence. From the desktop to the browser to the edge, we bridge the divide between the geospatial and the operational worlds.

A.30.1. Motivation to Participate

Hexagon Geospatial has been an active supporter of OGC and OGC standards for many years. Next to implementing a wide range of OGC standards, we have ample experience in applying OGC standards within industry solutions for a variety of domains, such as Aviation, Defense & Intelligence, Maritime, Transportation, Mining or Disaster Management. By being part of many testbeds and interoperability experiments the past decade and contributing numerous software components and engineering reports, we also learned that cooperation with other people on real-world use cases is an excellent way of testing and improving specifications. Hexagon Geospatial is motivated to share its expertise as long-term implementer and user to support the advancements of OGC standards related to map tiles, coverages and processes.

A.30.2. Implemented Solution

TBA

A.30.3. Proposed Alternatives

TBA

A.30.4. Experiences with OGC API Specifications

The Hexagon Geospatial product portfolios applied OGC standards from the start and currently implement more than a dozen OGC standards and candidate standards, including implementations of WFS, WMS, WCS, WMTS, WPS, CSW, GeoPackage, Filter Encoding, SLD / SE, GML, KML, 3D Tiles and NetCDF.

A.30.5. Other Impressions & Recommendations

TBA

A.31. Infinity Corporation Limited

TBA

A.31.1. Motivation to Participate

TBA

A.31.2. Implemented Solution

TBA

A.31.3. Proposed Alternatives

TBA

A.31.4. Experiences with OGC API Specifications

TBA

A.31.5. Other Impressions & Recommendations

TBA

A.32. interactive instruments GmbH

TBA

A.32.1. Motivation to Participate

TBA

A.32.2. Implemented Solution

TBA

A.32.3. Proposed Alternatives

TBA

A.32.4. Experiences with OGC API Specifications

TBA

A.32.5. Other Impressions & Recommendations

TBA

A.33. ISRIC - World Soil Information

TBA

A.33.1. Motivation to Participate

TBA

A.33.2. Implemented Solution

TBA

A.33.3. Proposed Alternatives

TBA

A.33.4. Experiences with OGC API Specifications

TBA

A.33.5. Other Impressions & Recommendations

TBA

A.34. Jacobs University

TBA

A.34.1. Motivation to Participate

TBA

A.34.2. Implemented Solution

TBA

A.34.3. Proposed Alternatives

TBA

A.34.4. Experiences with OGC API Specifications

TBA

A.34.5. Other Impressions & Recommendations

TBA

A.35. Jet Propulsion Laboratory

TBA

A.35.1. Motivation to Participate

TBA

A.35.2. Implemented Solution

TBA

A.35.3. Proposed Alternatives

TBA

A.35.4. Experiences with OGC API Specifications

TBA

A.35.5. Other Impressions & Recommendations

TBA

A.36. JRC, European Commission

TBA

A.36.1. Motivation to Participate

TBA

A.36.2. Implemented Solution

TBA

A.36.3. Proposed Alternatives

TBA

A.36.4. Experiences with OGC API Specifications

TBA

A.36.5. Other Impressions & Recommendations

TBA

A.37. Landcare Research, New Zealand

TBA

A.37.1. Motivation to Participate

TBA

A.37.2. Implemented Solution

TBA

A.37.3. Proposed Alternatives

TBA

A.37.4. Experiences with OGC API Specifications

TBA

A.37.5. Other Impressions & Recommendations

TBA

A.38. Land Information New Zealand

TBA

A.38.1. Motivation to Participate

TBA

A.38.2. Implemented Solution

TBA

A.38.3. Proposed Alternatives

TBA

A.38.4. Experiences with OGC API Specifications

TBA

A.38.5. Other Impressions & Recommendations

TBA

A.39. lat/lon GmbH

TBA

A.39.1. Motivation to Participate

TBA

A.39.2. Implemented Solution

TBA

A.39.3. Proposed Alternatives

TBA

A.39.4. Experiences with OGC API Specifications

TBA

A.39.5. Other Impressions & Recommendations

TBA

A.40. Meteorological Service of Canada

The Meteorological Service of Canada (MSC) is the national meteorological agency of Canada. It is a division of Environment and Climate Change Canada. The MSC primarily provides public meteorological information, weather forecasts, and warnings of severe weather and other environmental hazards.

A.40.1. Motivation to Participate

TBA

A.40.2. Implemented Solution

The MSC implemented and deployed an instance of pygeoapi - a Python server implementation of the emerging OGC API specifications. A screenshot of the landing page of the deployed service is shown in Figure 9. The software was configured to offer an interface that conforms to the OGC API - Features and OGC API - Processes specifications.

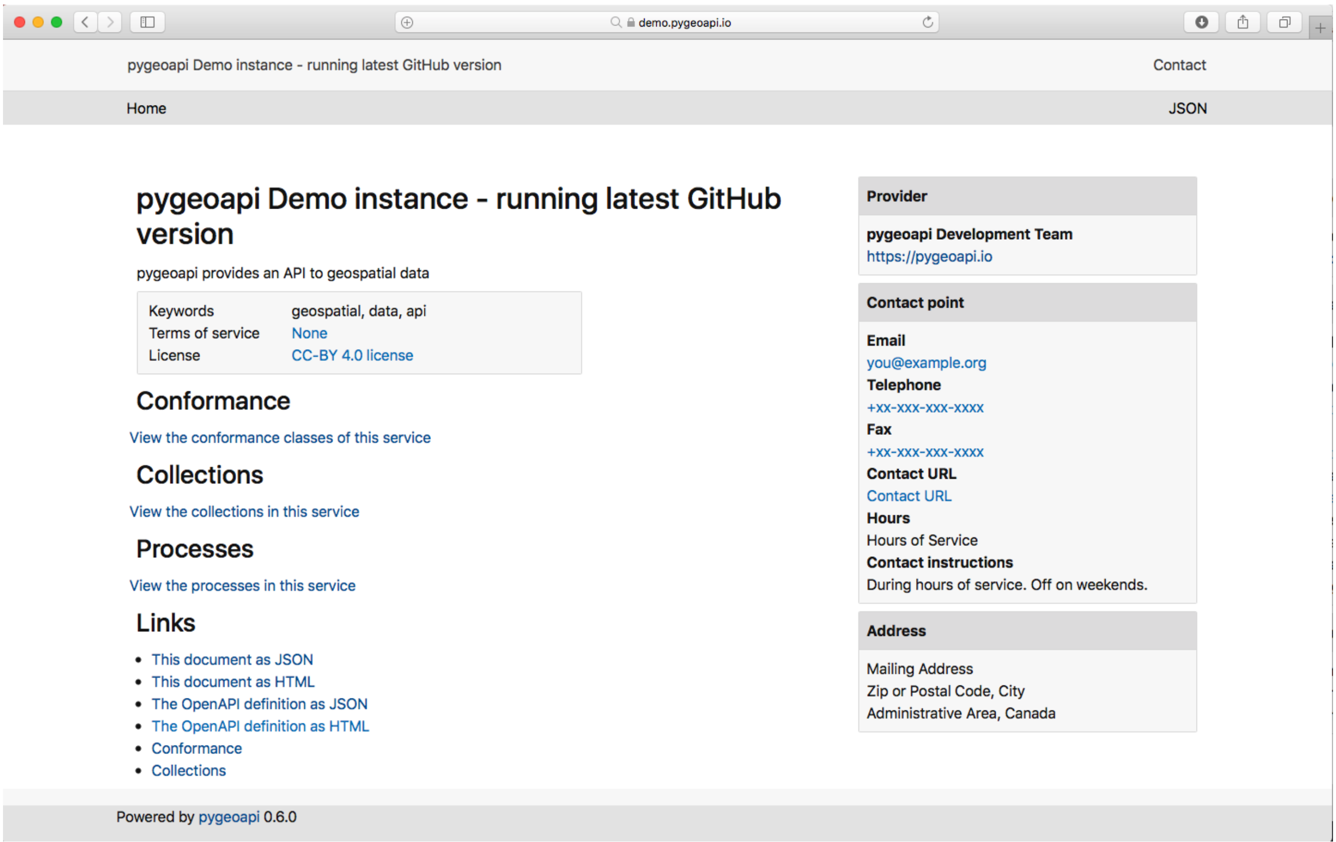

Figure 9. The landing page of the pygeoapi instance deployed by the Meteorological Service of Canada

A screenshot of the OpenAPI page of the deployed service is shown in Figure 10.

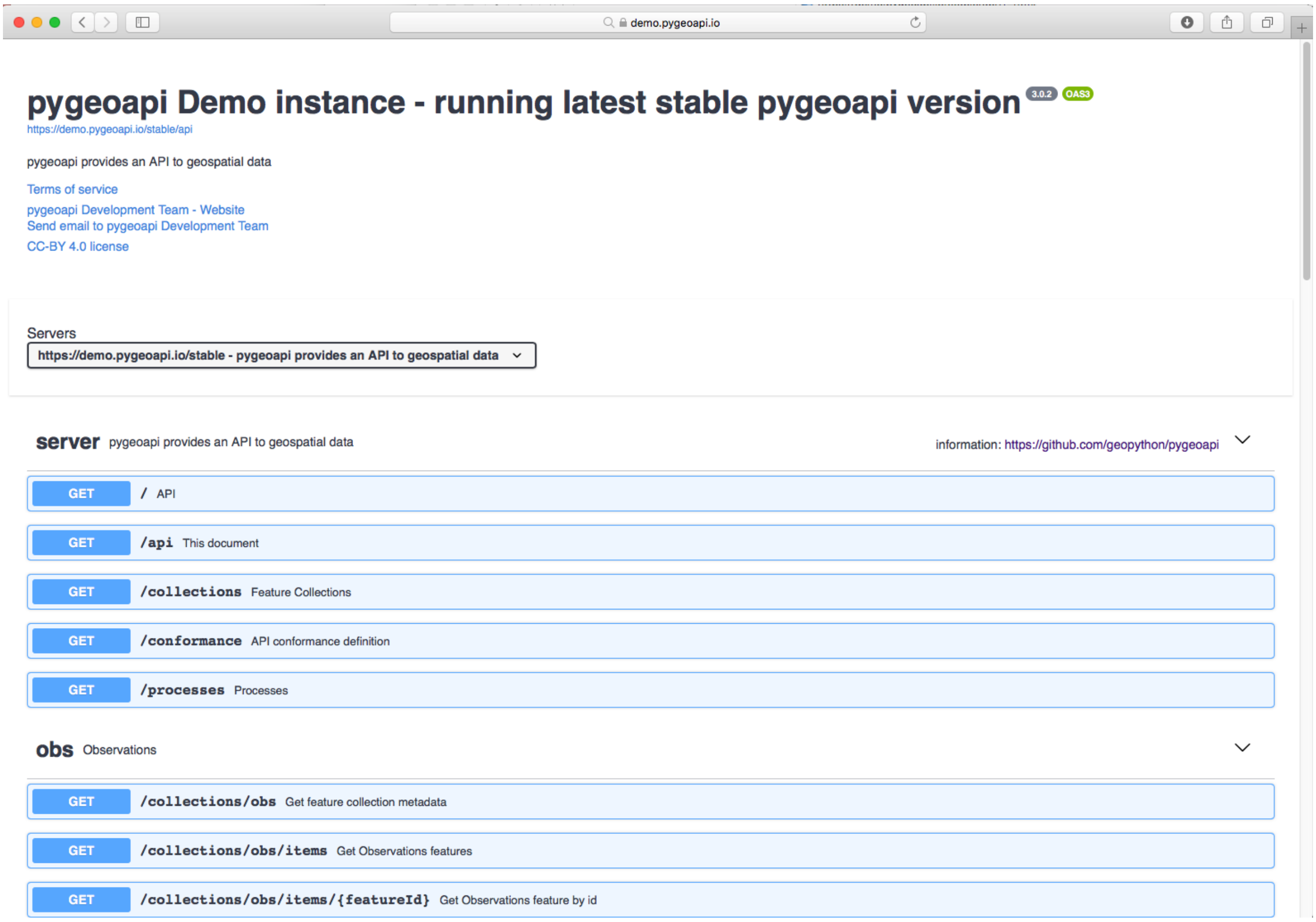

Figure 10. The OpenAPI definition of the pygeoapi instance deployed by the Meteorological Service of Canada

A.40.3. Proposed Alternatives

TBA

A.40.4. Experiences with OGC API Specifications

TBA

A.40.5. Other Impressions & Recommendations

TBA

A.41. Met Office

The Met Office is the national meteorological service for the United Kingdom. It produces both daily weather forecasts and long term climate predictions as well as performing fundamental science. It also runs several forcasting services for oceanography. Its customers cover a wide range of domains, such as aviation, defence, public sector, marine, private industry, media.

It has been an active Member of OGC since 2009, and, along with several other national meteorological services, founded the OGC Met-Ocean Domain Working Group and initiated a Memorandum of Understanding between OGC and the World Meteorological Organisation (WMO). WMO is the United Nations international treaty organisation that coordinates 193 national hydrometeorological services. It also defines international standards for meteorology, hydrology and oceanography.

The Met Office is also active in the World Wide Web Consortium (W3C) standards development organisation.

A.41.1. Motivation to Participate

Meteorology has had a long tradition of inventing and using technologies for its specific and very demanding requirements. As weather and climate information become ever more important to many aspects of society, it has become increasingly important to make information available using widespread technologies and approaches. We think using OpenAPI version 3 for defining public and private APIs to our data and information will have many benefits, for both customers and our infrastructure, especially when supporting services from computing clouds. We would like the APIs to be consistent with, and conformant to, any OGC OpenAPI standards.

A.41.2. Implemented Solution

Weather on the Web [https://github.com/opengeospatial/weather-on-the-web] (WotW) is an initiative to agree APIs for common data retrieval patterns, such as points, time-series, polygons, and trajectories, in 2, 3 and 4 Dimensions. The intent is to make it a global standard for meteorological and environmental services.

Point Data has been working for some time, as have Time-series at a point, and Polygons. Trajectories and Data Tiles are being developed.

A.41.3. Proposed Alternatives

Fall-back is to develop our meteorological APIs separetely from the OGC proposals and standardise through WMO, though this will be much slower.

A.41.4. Experiences with OGC API Specifications

A Hackathon was held in December 2018 in Washington DC, USA, building on the OGC WFS3.0 specification. Servers and clients based on existing WFS software were readily developed. Agreement was reached on the initial data retrieval patterns to support: point, timeseries, etc..

A.41.5. Other Impressions & Recommendations

In the WFS3.0 spec, /Items/ needs to be replaced by 'point', 'timeseries', trajectory', etc. This is consistent with the approach advocated by Joan Maso for maps and tiles.

A.42. National Aeronautics and Space Administration (NASA)

TBA

A.42.1. Motivation to Participate

TBA

A.42.2. Implemented Solution

TBA

A.42.3. Proposed Alternatives

TBA

A.42.4. Experiences with OGC API Specifications

TBA

A.42.5. Other Impressions & Recommendations

TBA

A.43. National Land Survey of Finland

TBA

A.43.1. Motivation to Participate

TBA

A.43.2. Implemented Solution

TBA

A.43.3. Proposed Alternatives

TBA

A.43.4. Experiences with OGC API Specifications

TBA

A.43.5. Other Impressions & Recommendations

TBA

A.44. Natural Resources Canada

TBA

A.44.1. Motivation to Participate

TBA

A.44.2. Implemented Solution

TBA

A.44.3. Proposed Alternatives

TBA

A.44.4. Experiences with OGC API Specifications

TBA

A.44.5. Other Impressions & Recommendations

TBA

A.45. NOAA/NWS

TBA

A.45.1. Motivation to Participate

TBA

A.45.2. Implemented Solution

TBA

A.45.3. Proposed Alternatives

TBA

A.45.4. Experiences with OGC API Specifications

TBA

A.45.5. Other Impressions & Recommendations

TBA

A.46. OSGeo

TBA

A.46.1. Motivation to Participate

TBA

A.46.2. Implemented Solution

TBA

A.46.3. Proposed Alternatives

TBA

A.46.4. Experiences with OGC API Specifications

TBA

A.46.5. Other Impressions & Recommendations

TBA

A.47. Princeton University

TBA

A.47.1. Motivation to Participate

TBA

A.47.2. Implemented Solution

TBA

A.47.3. Proposed Alternatives

TBA

A.47.4. Experiences with OGC API Specifications

TBA

A.47.5. Other Impressions & Recommendations

TBA

A.48. Princeton University Library

TBA

A.48.1. Motivation to Participate

TBA

A.48.2. Implemented Solution

TBA

A.48.3. Proposed Alternatives

TBA

A.48.4. Experiences with OGC API Specifications

TBA

A.48.5. Other Impressions & Recommendations

TBA

A.49. Quick Caption

TBA

A.49.1. Motivation to Participate

TBA

A.49.2. Implemented Solution

TBA

A.49.3. Proposed Alternatives

TBA

A.49.4. Experiences with OGC API Specifications

TBA

A.49.5. Other Impressions & Recommendations

TBA

A.50. Secure Dimensions

TBA

A.50.1. Motivation to Participate

TBA

A.50.2. Implemented Solution

TBA

A.50.3. Proposed Alternatives

TBA

A.50.4. Experiences with OGC API Specifications

TBA

A.50.5. Other Impressions & Recommendations

TBA

A.51. SigmaBravo

Sigma Bravo is a specialist provider of ICT services focused specifically on military operations. We enable and support an integrated, informed and agile 5th generation force.

A.51.1. Motivation to Participate

Sigma Bravo has a strong support of open standards as a path to interoperability. We recognise the value of simple, implementer-friendly standards.

A.51.2. Implemented Solution

Sigma Bravo extended the OpenSphere web application to support OGC API - Features and OGC API - Map Tiles. Interoperability with GeoServer, pygeoapi and SpaceBel OGC API - Features servers was demonstrated. Interoperability with ESRI OGC API - Map Tiles was also demonstrated.

A.51.3. Experiences with OGC API Specifications

Sigma Bravo has significant experience with previous OGC Specifications, particulary on GeoPackage. The most mature part of our OGC API implementation is OGC API - Features.

A.51.4. Other Impressions & Recommendations

Sigma Bravo commends the OGC on the OGC API program of work, and thanks OGC staff and GeoVation for their work in organising and hosting the hackathon. Sigma Bravo recommends that OGC seek to mature the OGC API standards through ongoing engagement and outreach activities, development of comprehensive Executable Test Suites and ongoing "in the open" standards development.

A.52. Simms Reeve

TBA

A.52.1. Motivation to Participate

TBA

A.52.2. Implemented Solution

TBA

A.52.3. Proposed Alternatives

TBA

A.52.4. Experiences with OGC API Specifications

TBA

A.52.5. Other Impressions & Recommendations

TBA

A.53. Sinergise

TBA

A.53.1. Motivation to Participate

TBA

A.53.2. Implemented Solution

TBA

A.53.3. Proposed Alternatives

TBA

A.53.4. Experiences with OGC API Specifications

TBA

A.53.5. Other Impressions & Recommendations

TBA

A.54. Solenix

TBA

A.54.1. Motivation to Participate

TBA

A.54.2. Implemented Solution

TBA

A.54.3. Proposed Alternatives

TBA

A.54.4. Experiences with OGC API Specifications

TBA

A.54.5. Other Impressions & Recommendations

TBA

A.55. Strategic Alliance Consulting Inc

TBA

A.55.1. Motivation to Participate

TBA

A.55.2. Implemented Solution

TBA

A.55.3. Proposed Alternatives

TBA

A.55.4. Experiences with OGC API Specifications

TBA

A.55.5. Other Impressions & Recommendations

TBA

A.56. University College London

TBA

A.56.1. Motivation to Participate

TBA

A.56.2. Implemented Solution

TBA

A.56.3. Proposed Alternatives

TBA

A.56.4. Experiences with OGC API Specifications

TBA

A.56.5. Other Impressions & Recommendations

TBA

A.57. University of Birmingham

TBA

A.57.1. Motivation to Participate

TBA

A.57.2. Implemented Solution

TBA

A.57.3. Proposed Alternatives

TBA

A.57.4. Experiences with OGC API Specifications

TBA

A.57.5. Other Impressions & Recommendations

TBA

A.58. University of Münster

Participant: Matthias Mohr

A.58.1. Motivation to Participate

- Align the openEO API with OGC API - Commons.
- Start discussions with the WPS community about alignment and their take on process chaining.
- Figure out future steps of WFS3 to port them back to the STAC specification.
- Discuss with CSW/CAT people about the planned steps and alignment with STAC.

A.58.2. Implemented Solution

- openEO API implementation, which partly implements and aligns with OGC API - Commons / WFS3.

A.58.3. Proposed Alternatives

- Workflows (process chaining) for WPS?
- Base CSW/CAT work on STAC?
- …

A.58.4. Experiences with OGC API Specifications

- WFS3 experience through STAC, contributed to WFS3 with some proposals for changes.
- Basic understanding of WPS and Commons

A.58.5. Other Impressions & Recommendations

TBA

A.59. University of Notre Dame

TBA

A.59.1. Motivation to Participate

TBA

A.59.2. Implemented Solution

TBA

A.59.3. Proposed Alternatives

TBA

A.59.4. Experiences with OGC API Specifications

TBA

A.59.5. Other Impressions & Recommendations

TBA

A.60. WebGeoDataVore

TBA

A.60.1. Motivation to Participate

TBA

A.60.2. Implemented Solution

TBA

A.60.3. Proposed Alternatives

TBA

A.60.4. Experiences with OGC API Specifications

TBA

A.60.5. Other Impressions & Recommendations

TBA

A.61. West University of Timisoara

TBA

A.61.1. Motivation to Participate

TBA

A.61.2. Implemented Solution

TBA

A.61.3. Proposed Alternatives

TBA

A.61.4. Experiences with OGC API Specifications

TBA

A.61.5. Other Impressions & Recommendations

TBA

Appendix B: Revision History

Table 3. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
June 2, 2019	G. Hobona	.1	5	Adapted Scott Simmons's blog
June 3, 2019	G. Hobona	.2	all	initial version
TBA	TBA	TBA	TBA	TBA
TBA	TBA	TBA	TBA	TBA

Appendix C: Bibliography

1. Percivall, G.: OGC® Open Geospatial APIs - White Paper. OGC 16-019r4,Open Geospatial Consortium, http://docs.opengeospatial.org/wp/16-019r4/16-019r4.html (2017).
2. Tandy, J., Brink, L. van den, Barnaghi, P.: OGC/W3C Spatial Data on the Web Best Practices. OGC 15-107,Open Geospatial Consortium & World Wide Web Consortium, https://www.w3.org/TR/sdw-bp/ (2017).
3. Wikipedia: 2018 European floods, https://en.wikipedia.org/wiki/2018_European_floods.
4. Wikipedia: 2019 Midwestern U.S. floods, https://en.wikipedia.org/wiki/2019_Midwestern_U.S._floods.
5. Idol, T., Thomas, R.: OGC Development of Disaster Spatial Data Infrastructures for Disaster Resilience. OGC 18-087r5,Open Geospatial Consortium, https://portal.opengeospatial.org/files/18-087r5 (2018).
6. European Space Agency: Copernicus - Observing the Earth, https://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Overview4, (2018).
7. Sinergise: Sentinel Hub, https://www.sentinel-hub.com/develop/capabilities/wms.
8. MetOffice: Data Point, https://www.metoffice.gov.uk/datapoint.
9. MetOffice: UK Met Office Atmospheric Deterministic and Probabilistic Forecasts, https://registry.opendata.aws/uk-met-office/.
10. Ordnance Survey: OS Open Zoomstack, https://www.ordnancesurvey.co.uk/business-and-government/products/os-open-zoomstack.html.
11. Meteorological Service of Canada: Meteorological Service of Canada Datamart, https://dd.weather.gc.ca.
12. Meteorological Service of Canada: Geospatial web services, https://www.canada.ca/en/environment-climate-change/services/weather-general-tools-resources/weather-tools-specialized-data/geospatial-web-services.html.
13. National Oceanic and Atmospheric Administration: National Weather Service Data as OGC Web Services, https://www.weather.gov/gis/WebServices.