DRAFT

OGC API – Common and OGC API – Features Sprint 2020

Summary Engineering Report

Publication Date: 2021-02-23

Approval Date: 2020-12-14

Submission Date: 2020-11-16

Reference number of this document: OGC 20-091

Reference URL for this document: http://www.opengis.net/doc/PER/OGCAPIFeaturesCommonSprint1

Category: OGC Public Engineering Report

Editor: Gobe Hobona

Title: OGC API – Common and OGC API – Features Sprint 2020: Summary Engineering Report

OGC Public Engineering Report

COPYRIGHT

WARNING

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Public Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

LICENSE AGREEMENT

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

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

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

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

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the

Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Table of Contents

Chapter 1. Subject

The subject of this Engineering Report (ER) is a code sprint that was held from 29 to 30 September 2020 to advance the development of the OGC API - Common - Part 2: Geospatial Data draft standard and the OGC API – Features – Part 4: Simple Transactions 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). The code sprint was hosted online. The event was sponsored by Ordnance Survey (OS).

Chapter 2. Executive Summary

This Engineering Report (ER) summarizes the main achievements of the OGC API – Common and OGC API – Features Sprint, conducted between September 29 – 30, 2020. The sprint served to accelerate development of the OGC API - Common - Part 2: Geospatial Data draft standard and the OGC API – Features – Part 4: Simple Transactions draft standard.

The objectives of the code sprint were to:

- Develop prototype implementations of OGC API – Common – Part 2: Geospatial Data
- Develop prototype implementations of OGC API – Features – Part 4: Simple Transactions
- 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 OGC API - Common - Part 2: Collections draft standard provides a common connection between the API landing page and resource-specific details. That connection includes metadata that describes the collections of hosted resources, common parameters for selecting subsets of those collections, and URI templates for identifying the above. This common connection is sufficient to start the client down the path to resource discovery.

OGC API - Features is a multipart standard that provides API building blocks for creating, modifying, and querying geospatial features over the Web. The OGC API – Features – Part 4: Simple Transactions draft standard extends the core capabilities specified in the approved Part 1 standard with the ability to add, replace, modify, and/or delete individual feature instances from a single feature collection. The sprint produced very valuable results. Participants identified areas for improving the draft standards, allowing the editors of the draft standards to record their recommendations. The editors are expected to discuss the feedback from the sprint participants with the Standards Working Group (SWG).

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

2.2. Foreword

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

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

Chapter 3. References

The following normative documents are referenced in this document.

- OGC: OGC 17-069r3, OGC API - Features - Part 1: Core 1.0 (2019)
- OGC: OGC 20-002, OGC API - Features - Part 4: Simple Transactions candidate standard
- OGC: OGC 19-072, OGC API - Common - Part 1: Core candidate standard
- OGC: OGC 20-024, OGC API - Common - Part 2: Geospatial Data candidate standard
- 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)

- **feature**

  abstraction of real world phenomena [ISO 19101-1:2014]

  NOTE For those unfamiliar with the term 'feature', the explanations on Spatial Things, Features and Geometry [https://www.w3.org/TR/sdw-bp/#spatial-things-features-and-geometry] in the W3C/OGC Spatial Data on the Web Best Practice document [http://docs.opengeospatial.org/is/17-069r3/17-069r3.html#SDWBP] provide more detail.

- **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 OGC API – Common and OGC API – Features Sprint, conducted from 29 to 30 September 2020. The sprint had been organized to advance the development of the OGC API - Common - Part 2: Geospatial Data draft standard and the OGC API – Features – Part 4: Simple Transactions draft standard. 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 support OGC Standards Program activities.

This code sprint was held as part of a series in the third quarter of 2020. The other code sprints included:

- OGC API – Maps Sprint held July 28th to 29th, 2020. This sprint focused on the OGC API - Maps - Part 1: Core draft standard which defines a Web API for publishing and accessing digital maps.
- OGC API – Coverages Sprint held August 10th to 11th, 2020. This sprint focused on the OGC API - Coverages - Part 1: Core draft standard which defines a Web API for publishing and accessing coverage data.

5.1. Participants

Software developers and solutions architects from the following organizations registered to participate in the code sprint:

- Azavea
- Bezirksregierung Köln - Geobasis NRW
- Camptocamp
- Cognitics, Inc
- CubeWerx Inc.
- Ecere Corporation
- Element 84
- EOX IT Services GmbH
- Esri
- Geobeyond
- Geosolutions Consulting
- Giscorps
- Global Nomad GIS Services

- Heazeltech
- Hxart
- ICEYE
- IIT Bombay
- Image Matters LLC
- Indian Institute of Technology, Bombay
- interactive instruments GmbH
- ISRIC - World Soil Information
- Kitware Inc.
- lat/lon GmbH
- Met Office
- Meteorological Service of Canada
- NOAA/NOS/NGS
- Ordnance Survey
- Polaris Digitech Limited
- Princeton University
- Secure Dimensions
- Texas A&M University
- UAB-CREAF
- Uppsala university
- US Army Geospatial Center
- virtru
- Wuhan University

5.2. OGC API – Common – Part 2: Geospatial Data draft standard

Geospatial data is seldom considered as a single entity. Feature Collections, Coverages, Data Sets are typically considered as aggregations of Spatial or Temporal Things. A web API conforming to OGC API standards therefore should be able to expose its holdings as aggregates of spatial resources.

The purpose of the OGC API - Common - Part 2: Geospatial Data draft standard is to provide a common connection between the API landing page and resource-specific details. That connection includes metadata which describes the hosted geospatial resources, common parameters for selecting subsets of those resources, and URI templates for identifying the above.

This draft standard specifies the following conformance classes:

- The Collections conformance class specifies the requirements necessary to discover and

understand a generic collection and its contents.

- The HTML conformance class describes how to represent collections of geospatial data in HTML encoding.
- The JSON conformance class describes how to represent collections of geospatial data in JSON encoding.

5.3. OGC API – Features – Part 4: Simple Features draft standard

The OGC API – Features – Part 4: Simple Features draft standard specifies an extension to OGC API – Features that defines the behavior of a server that supports simple transactions. Simple transactions are transactions that add, replace, modify or delete individual resources from a single collection.

Specifically, OGC API – Features – Part 4: Simple Features describes:

- How to add a new resource instance to a collection.
- How to modify an existing resource from a collection; this includes entirely replacing the existing resource or simply modifying one or more parts of a resource.
- How to remove an existing resource from a collection.

This draft standard covers the following conformance classes:

- The Simple Transactions conformance class provides the ability to add, replace and remove individual resources from a collection.
- The PATCH Updates conformance class provides the ability to use the HTTP PATCH method to modify parts of an existing resource without the need to transmit a complete replacement resource.
- The Features conformance class describes requirements for the case when the resource type is a feature.

Chapter 6. High-Level Architecture

The focus of the sprint was on support of the development of the OGC API - Common - Part 2: Geospatial Data draft standard and the OGC API – Features – Part 4: Simple Transactions draft standard. 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.

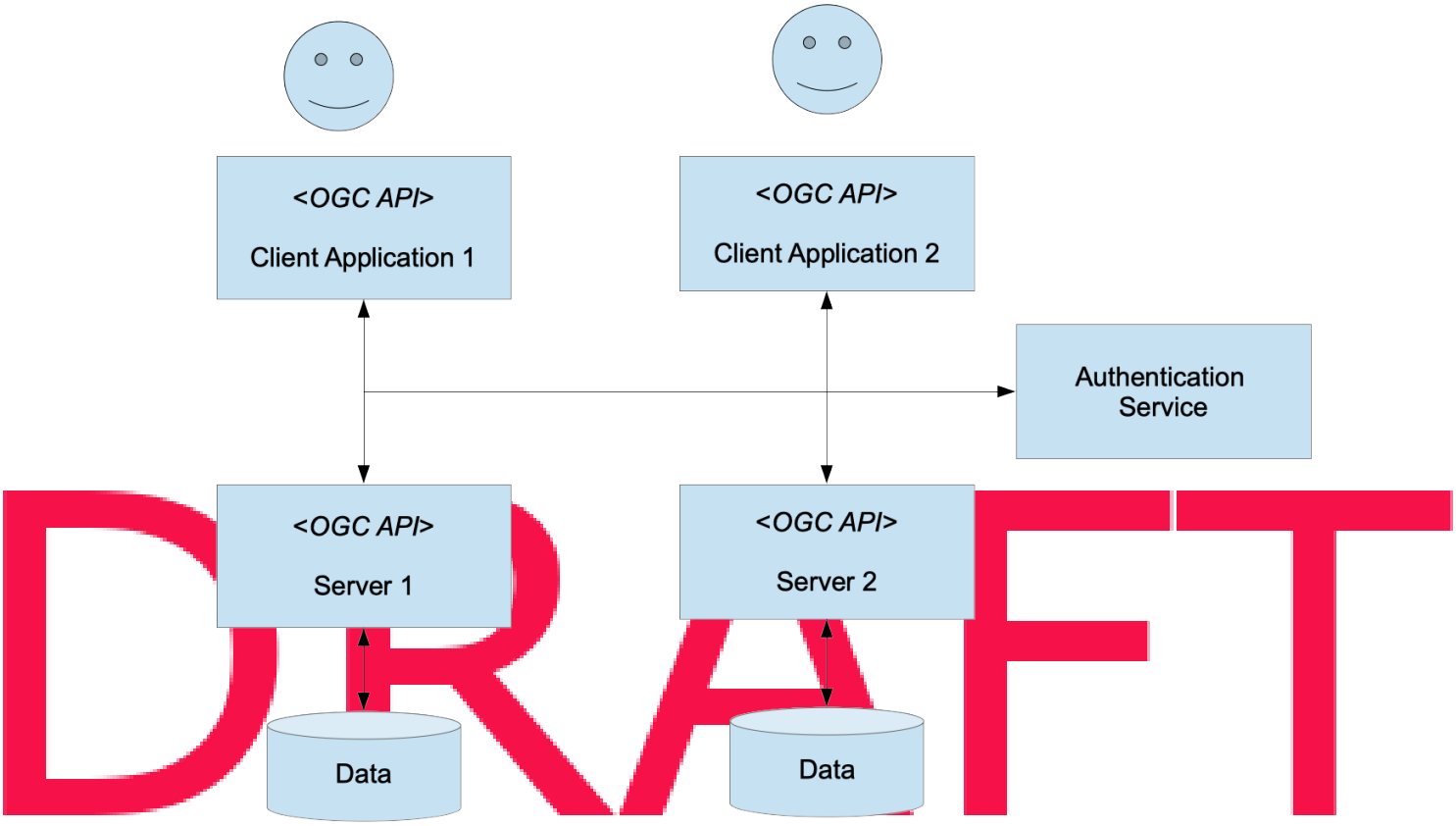

Figure 1. High level architecture implemented during the sprint

A detailed view of the security aspects of the sprint architecture is shown in Figure 2. This aspect of the sprint architecture partly addressed the observation made in Testbed-14 that there was a lack of a component that could utilize OpenAPI and capabilities documents, combined with a catalog of service references to completely automate the acquisition of user tokens and their use on workflows [1].

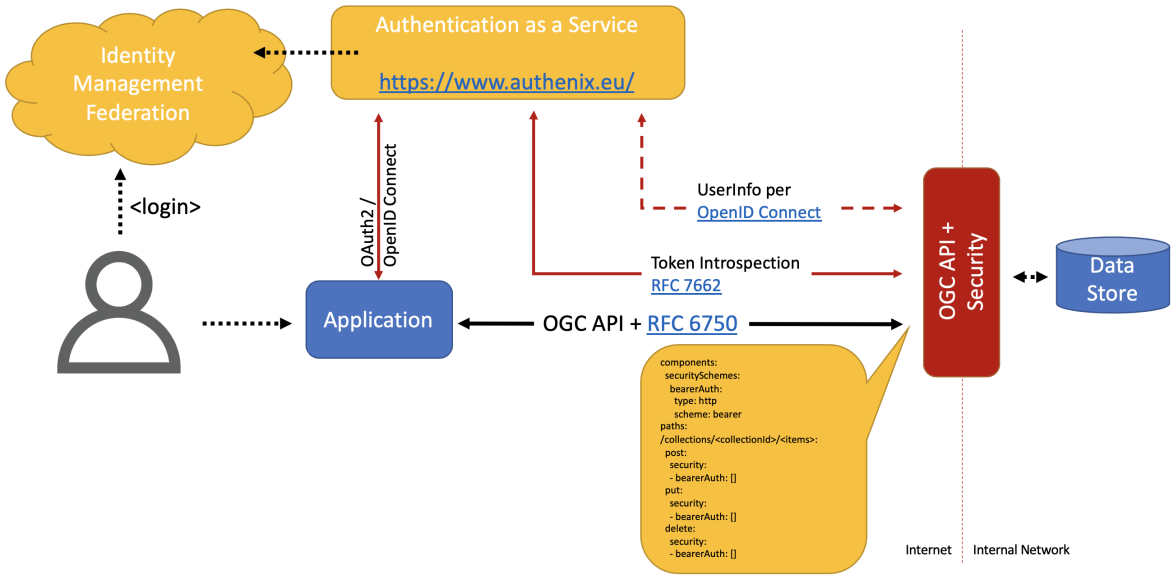

Figure 2. Detailed view of the security aspects of architecture

As illustrated the sprint architecture was designed with the view of allowing client applications to connect to different servers that implement OGC API - Features and OGC API - Common. The servers were provisioned with vector feature data, some of which was from the OS Open Zoomstack data product. The client applications were then configured to submit create, update and delete requests to the servers.

Chapter 7. Results

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

The Centre for Ecological Research and Forestry Applications (CREAF) at the Autonomous University of Barcelona (UAB) deployed an instance of the MiraMon Map Server that implements support for multiple OGC API standard. The server is implemented as a CGI application encoded in C language as a part of the MiraMon Geographic Information System (GIS) & Remote Sensing (RS) suite and is interoperable with other vendors' clients. During the sprint, the CREAF team was able to modify the client application to enable it to support security schemes offered by OpenAPI and supported by OGC API – Features – Part 4: Simple Transactions. A screenshot of the user interface is shown in Figure 3.

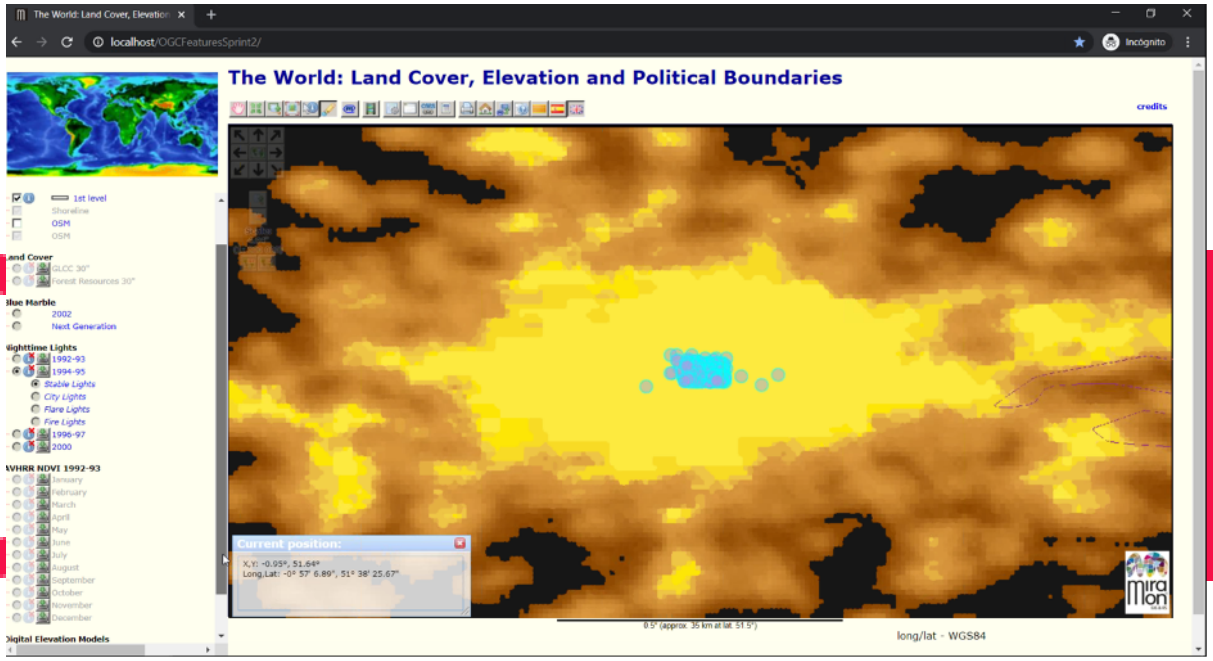

Figure 3. MiraMon client application by UAB-CREAF.

CubeWerx deployed an implementation of multiple OGC API standards including OGC API – Common – Part 2 and OGC API – Features – Part 4. The CubeWerx server ("cubeserv") is implemented in the C programming language. The server exposes an interactive API description created from the OpenAPI definition document which enables both users and client applications to query the server. During the sprint, the CubeWerx team was able to modify the server to enable it to support security schemes offered by OpenAPI and supported by OGC API – Features – Part 4: Simple Transactions. The server was also configured to expose an OS Open Zoomstack dataset as a collection that is consistent with OGC API – Common – Part 2: Geospatial Data. A screenshot of the API definition page is shown in Figure 4.

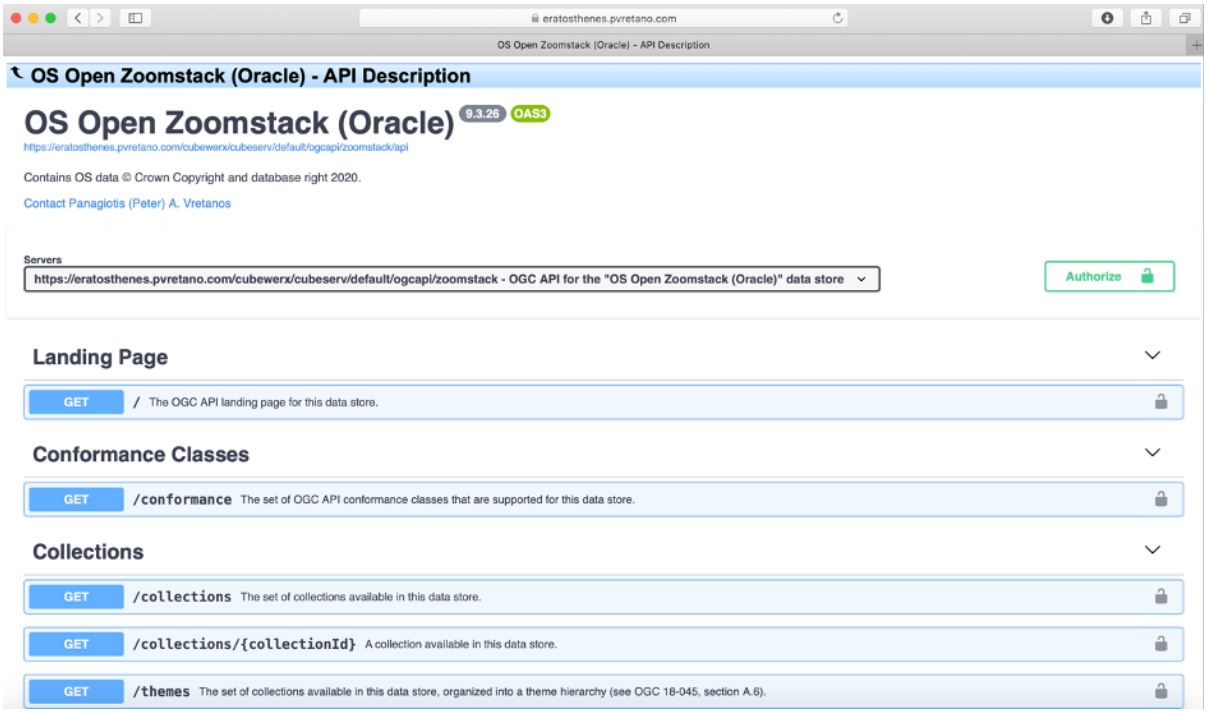

Figure 4. API definition presented by the CubeWerx server

Secure Dimensions deployed an instance of the Authentix server. The instance provided authentication capabilities for the other components of the sprint architecture. The screenshot below shows the Authentix server's landing page on the right hand-side and a terminal connected to the CubeWerx server on the left-hand side. The CubeWerx server was successfully authenticated by the Authentix server. A screenshot is shown in Figure 5.

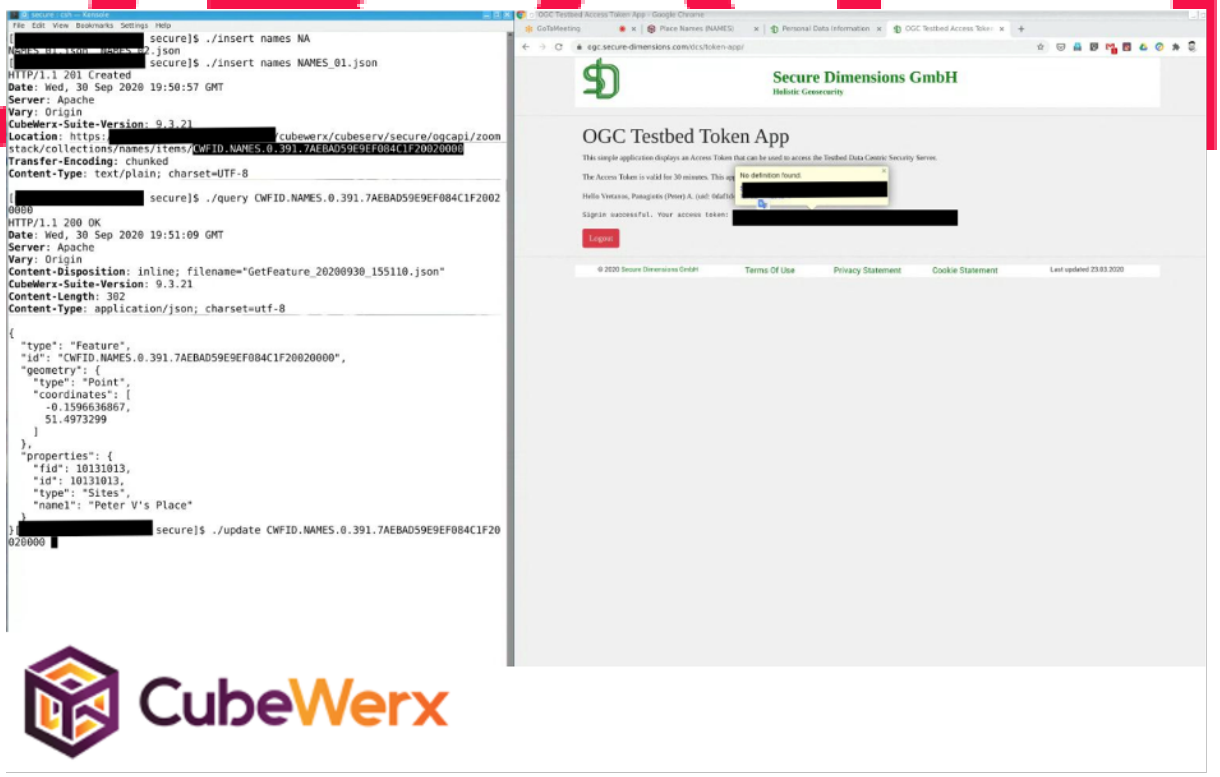

Figure 5. Terminal accessing CubeWerx server (left) and Landing page of the Secure Dimensions Authentix product (right)

Known as Geobasis NRW, division 7 of the Cologne Government Regional Office supplies data for a wide range of purposes including planning, construction projects, transport and the supply of basic

community services, nature conservation and environmental protection, valuation of real property, real estate transactions and mortgage borrowing. Participants from Geobasis NRW took part in the code sprint with a browser-based client application of OGC API - Features. A screenshot of the client application is shown below. Their application was able to authenticate with the Authentix server and then invoke transaction operations on the CubeWerx server. A screenshot is shown in Figure 6.

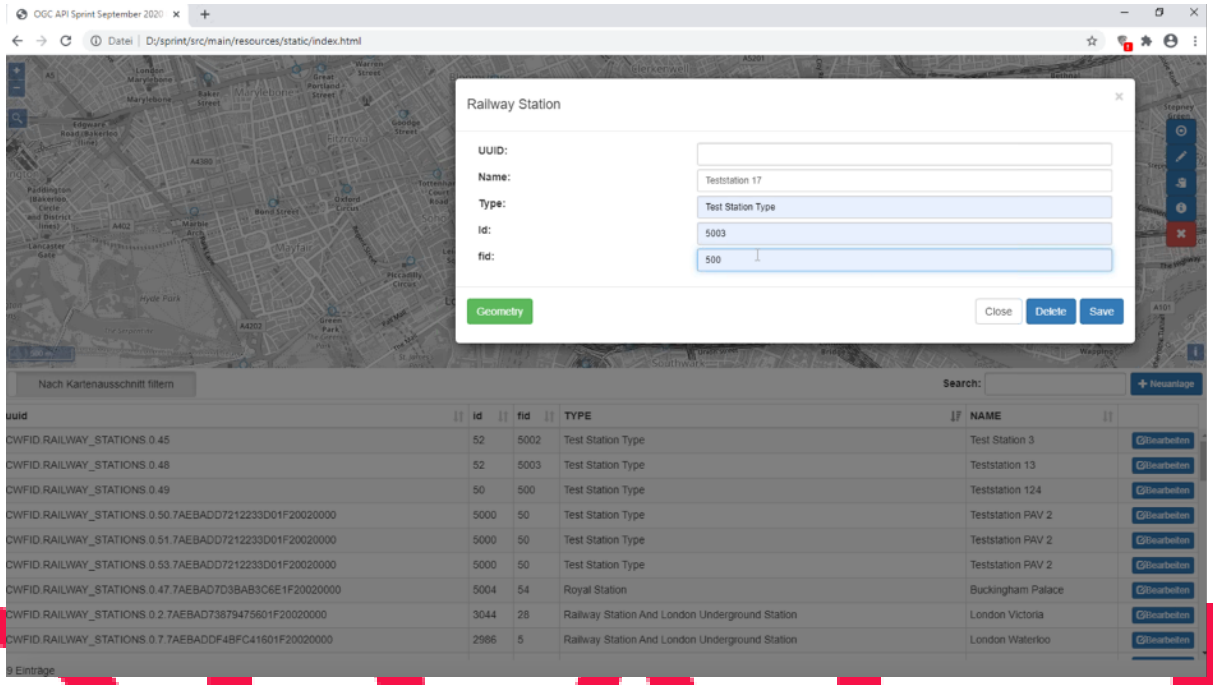

Figure 6. Screenshot of the client application from Geobasis NRW

Global Nomad deployed an instance of GeoServer. The instance was configured to offer support for OGC API - Features through the OGC API community extension of GeoServer. A screenshot is shown in Figure 7.

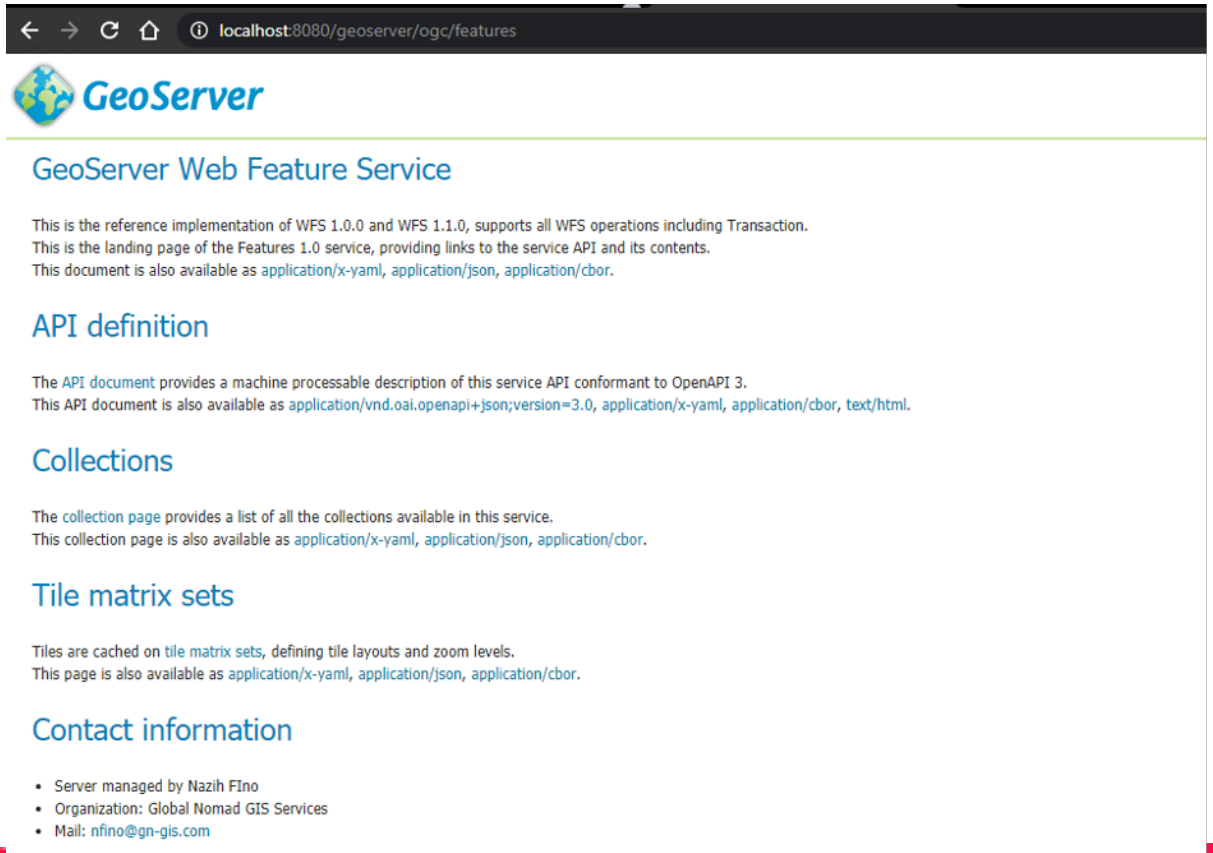

Figure 7. Landing page of GeoServer instanced deployed by Global Nomad

interactive instruments GmbH deployed their ldproxy product to support the sprint. The product implements OGC API – Features. A screenshot is shown in Figure 8.

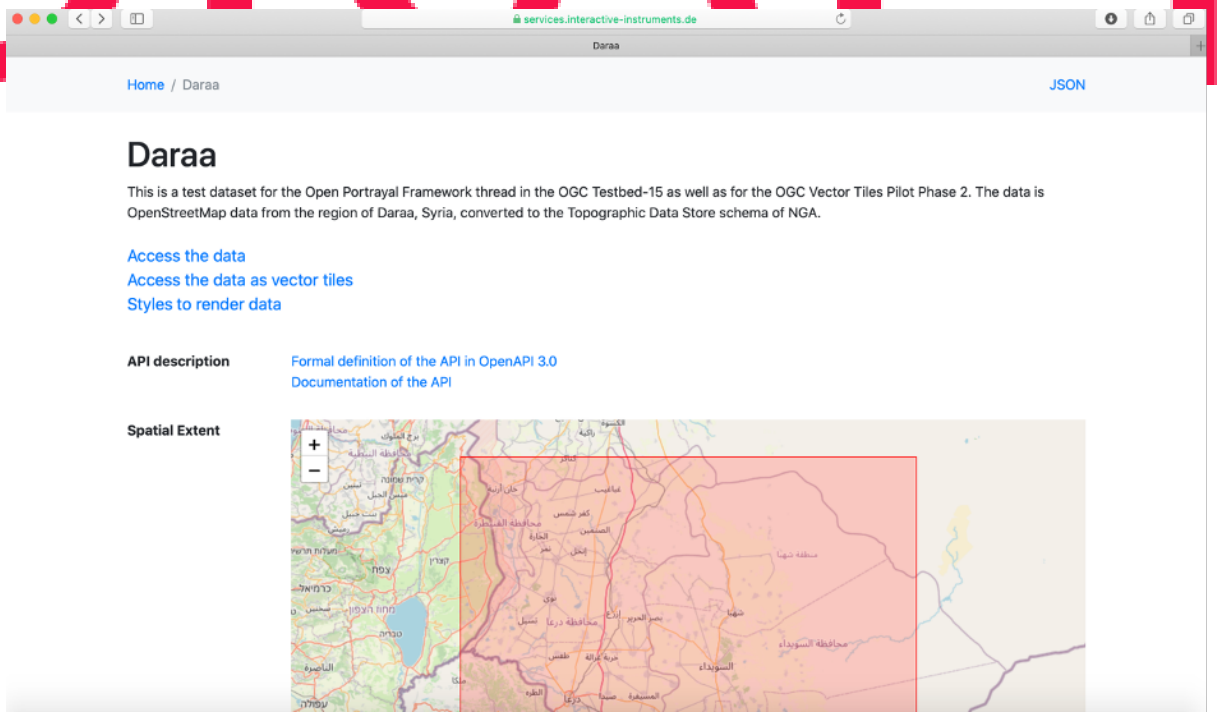

Figure 8. Landing page of interactive instruments ldproxy

Other software deployments during the code sprint included:

- An implementation of OGC API – Features developed using nodejs and deployed by Software

developer Haoliang Yu. The implementation offered a lightweight and modular JavaScript library supporting the Features API.

- The OGC Compliance Program provides a free online testing facility based on Team Engine and a set of test suites dedicated to specific protocols and versions, as well as specification profiles and extension. For this code sprint, the executable test suite of OGC API – Features was extended to introduce prototype support for the OGC API – Features – Part 4: Simple Transactions draft standard. Implementation of the prototype was led by lat/lon GmbH.

Chapter 8. Discussion

The participants used the Gitter platform for written discussion. This was in addition to using Gotomeeting for discussion during the scheduled teleconferences. Individual issues were recorded on the Issues board on GitHub. A screenshot of the Gitter channel is shown below. The Gitter channel can be found at https://gitter.im/opengeospatial/OGC-API-Sprint-September-2020

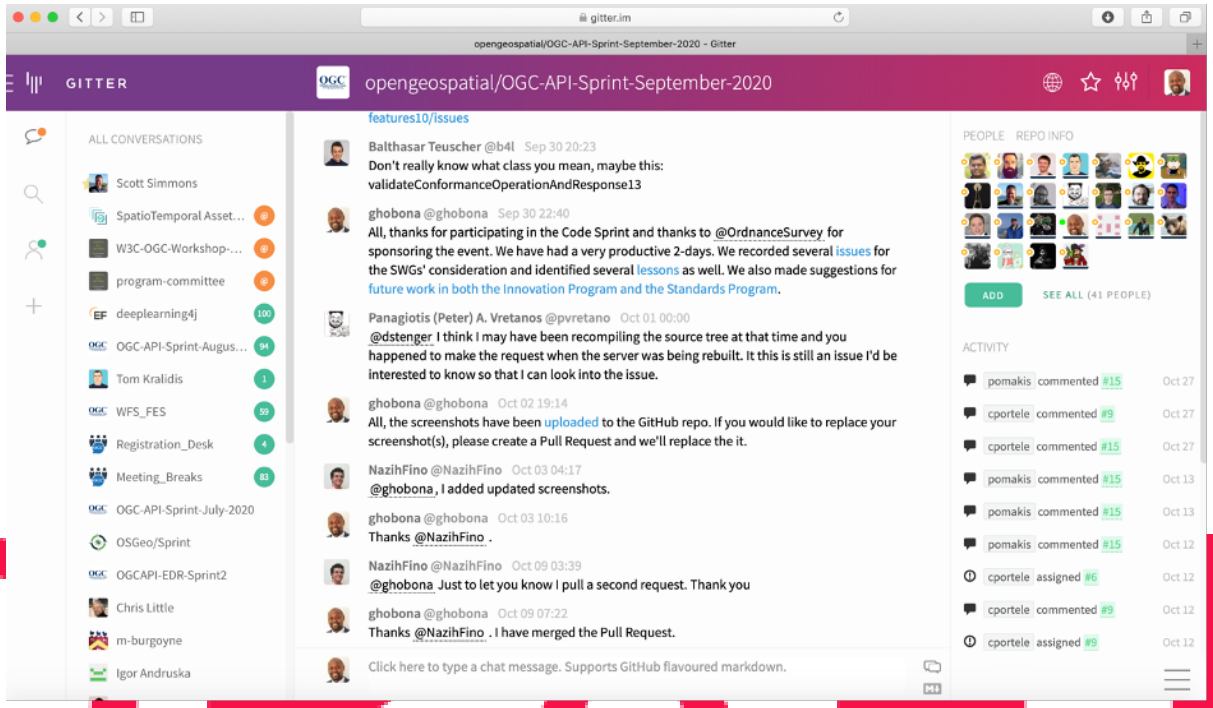

Figure 9. Screenshot of Gitter channel

A screenshot of the GitHub repository is shown below. The GitHub repository can be found at https://github.com/opengeospatial/OGC-API-Sprint-September-2020

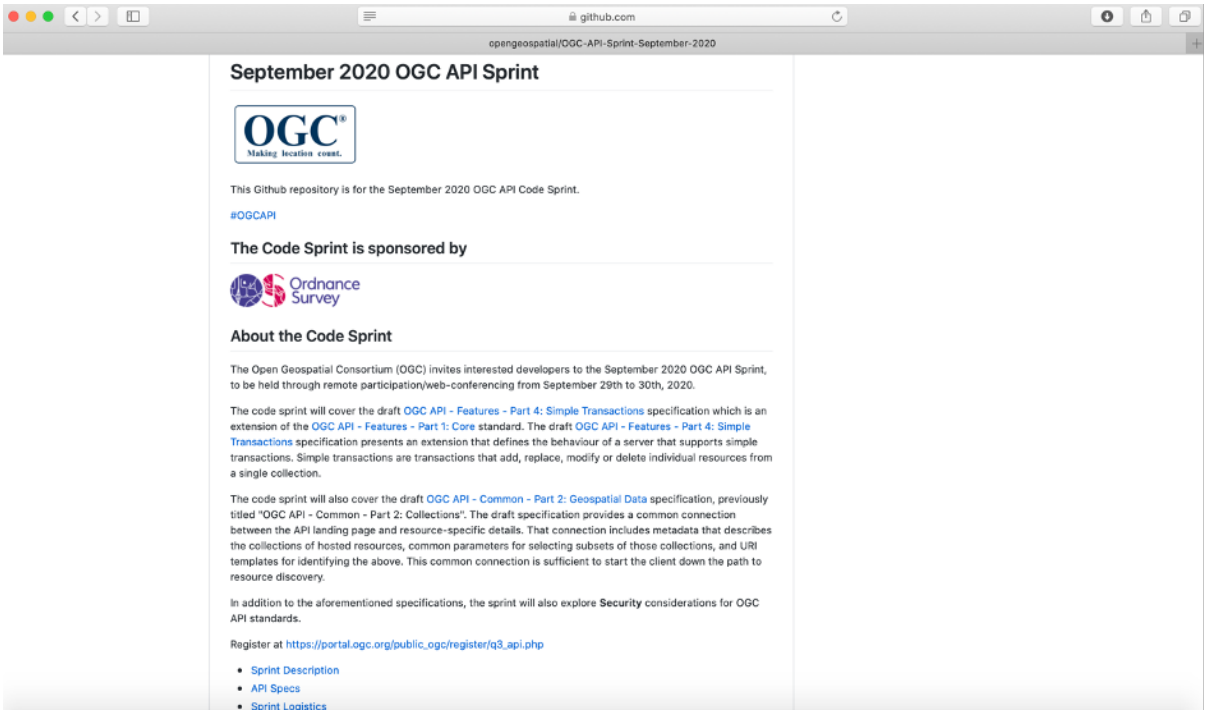

Figure 10. Screenshot of GitHub repository

The next subsections provide a summary of the discussion.

8.1. Discussion on OGC API - Common - Part 2: Geospatial Data

The sprint participants recommended [https://github.com/opengeospatial/OGC-API-Sprint-September-2020/issues/10] that the SWG defines a specific relation type to link to metadata documents which could be used to link to standard metadata types such as ISO 19115, properly identified by media types. This could possibly also include additional fields. Both of these could be done at both the dataset and at the collection level. The participants recommended that this should be specified in OGC API - Common - Part 2: Geospatial Data.

The sprint participants observed found [https://github.com/opengeospatial/OGC-API-Sprint-September-2020/issues/11] that there was no way to add attribution about the collections except through the description field. This issue was identified as a result of the need to add a copyright statement and other attribution about the collections and on the dataset landing page, for example "Contains OS data © Crown Copyright and database right 2020".

The need for service metadata was also identified [https://github.com/opengeospatial/OGC-API-Sprint-September-2020/issues/12] by the sprint participants. This however raised the question of how to define 'service metadata'. A proposal was put forward to define service metadata as "Any information which does not qualify the data itself, but the software and organization hosting and/or developing the software. Contact information (but not about the data)".

There was a suggestion [https://github.com/opengeospatial/OGC-API-Sprint-September-2020/issues/16] to consider revising the structure of Exception [https://github.com/opengeospatial/oapi_common/blob/master/core/openapi/schemas/exception.yaml] responses, so that they could be as informative as those from OWS Common. One suggestion was to have an array of descriptions. There was an observation that the 'code' property on Exception responses is not well defined and it appears to duplicate the role of the HTTP status code. Participants suggested that there could be a recommendation to use the JSON Schema for exception responses, and the Guide could provide more information.

8.2. Discussion on OGC API - Features - Part 4: Simple Transactions

When implementing a client application it is often useful to be able to retrieve the schema of the feature or collection that is going to be updated. An end-user or client application would need to know the properties of the feature (or collection), however OGC API - Features - Part 1: Core does not provide the means to describe a schema. This was noted [https://github.com/opengeospatial/OGC-API-Sprint-September-2020/issues/9] as an issue to be addressed. The sprint participants also discussed an approach whereby the schema for feature collections is included in the API Definition (in JSON or YAML). In this approach, implementations would use JSON Schema to constrain the JSON keys representing feature properties. The sprint participants acknowledged that the approach could work but also cautioned that it could lead to very large API Definition documents.

The sprint participants suggested [https://github.com/opengeospatial/OGC-API-Sprint-September-2020/issues/13] that there is a need for the SWG to provide clarification on the difference between the HTTP

methods of PUT, PATCH and POST. The following rules were suggested:

- PATCH does not allow you to update the identifier
- PUT does allow you to set the identifier
- POST allows you to send to a parent resource

The question of whether to allow use of HTTP POST for retrieving content, a question also referred to as the 'HTTP dilemma', was discussed [https://github.com/opengeospatial/OGC-API-Sprint-September-2020/issues/14] at length. HTTP POST was noted as being potentially useful when an HTTP GET request could result in a URL that is too long. The sprint participants suggested allowing HTTP POST to be used for retrieving data, in addition to allowing it to be used for inserting data.

Servers typically advertise which formats they support (e.g. GeoJSON, GML, etc.) through the conformance declaration. However, not all formats (e.g. HTML) are a suitable representations for use with a HTTP POST. The sprint participants discussed [https://github.com/opengeospatial/OGC-API-Sprint-September-2020/issues/15] how a a client application might determine which formats may be used with a POST/PUT.

One approach that was suggested [https://github.com/opengeospatial/OGC-API-Sprint-September-2020/issues/15] is to use the OpenAPI document, however this was observed to be undesirable as it would make clients (especially ones that don't use OpenAPI-base tool chains) have to read the OpenAPI document to figure this out. Other approaches suggested included a variant of OPTIONS, the use of `/queryables` or a parameterized conformance document that crosswalks the format-based conformance classes with the methods they support (e.g. GeoJSON with GET,PUT,POST).

During the sprint, a number of participants encountered the issue of Cross-Origin Resource Sharing (CORS [https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS]). This is an issue that affects many applications that run from a web-browser. CORS is a mechanism, relating to HTTP headers, that allows a server to indicate any other origins (domain, protocol, or port) than its own from which a browser should permit loading of resources. The sprint participants noted [https://github.com/opengeospatial/OGC-API-Sprint-September-2020/issues/19] that it would be helpful to developers if the Security Considerations section of the OGC API - Features - Part 4: Simple Transactions draft standard mentioned CORS. The participants also observed that this could also benefit other OGC API standards as well.

8.3. 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:

- OGC API – Features is much simpler to implement than alternatives. The combination of GeoJSON and OGC API – Features is simple to use.
- It is possible to implement typical OpenAPI security schemes in OGC API – Features. This was demonstrated using an Authentication Service to authenticate clients and servers.
- On the client side, the hello.js JavaScript library made it possible and easy to integrate the authentication calls into the client.
- On the client side, authentication was required at the moment the user wanted to insert a

feature.

- There will be a need to outline the capabilities that will be included in the Complex Transactions extension. This may include locking, as well as other capabilities.
- There is also an issue of several people doing updates from different places. How a client will be aware of new features emerging in the database. PubSub approaches could help: clients getting notifications on changes from others
- The sprint has highlighted the key role of the client application in a process that involves transactions.

Chapter 9. Conclusions

The sprint successfully demonstrated the ability to support the creation, deletion and updating of features through the OGC API – Features – Part 4: Simple Transactions draft standard. The sprint also successfully demonstrated support for collections of geospatial data, as required by OGC API – Common – Part 2: Geospatial Data. The implementation and demonstration of these draft standards during the sprint was therefore a significant achievement.

The participants demonstrated that the OGC API pattern can indeed address the need for simple transactions and collections of geospatial data. Further, the successful binding of client applications to services protected by security schemes described through OpenAPI was also a significant achievement. It is envisaged that the outputs of this sprint will help to accelerate the development of OGC API - Common and OGC API - Features standards.

9.1. Future work

The following general recommendations for future work items have been made.

9.1.1. Ideas for the Innovation Program

Future work in Innovation Program could include:

- Publish-Subscribe: Approaches such as OGC PubSub, e-mail notification, Linked Data Notification could be useful. These could also be query-based on the delta updates/changesets approach from recent testbeds.
- Web sockets: Support for the notification of a client that new content has appeared.
- Schema vs Schema-less: Prototyping of an extension to the API to support schemas, and perhaps fully schema-less APIs.
- Change Only Updates: Potential support for change-only updates could be supported through the Delta Updates and/or the ChangeSet API from Testbed-15. This could be done in future testbeds with Ordnance Survey data, for example. See this video for an overview https://eratosthenes.pvretano.com/Projects/tb15/Videos/CubeWerx_DU_Demo_Video_TB15_DEC2019.mp4
- Metadata: Harmonization between OGC API – Common and OGC API – Records with regard to metadata is something to look at.
- Security: Experiment with additional authentication schemes.
- Other HTTP methods: Experiment with use of a HEAD and/or OPTIONS request to support metadata.

9.1.2. Ideas for the Standards Program

Future work in the Standards Program could include:

- The sprint has helped to provide initial validation. The editors will be looking at the issues from the sprint and will then push the document forward.

- The Features API SWG should focus on Part 3 and 4 next.
- OGC API - Common SWG should consider revision of the Exception report structure defined in OGC API – Common.
- Architecture DWG should hold a session on Security Considerations for OGC API.

Appendix A: Revision History

Table 1. Revision History

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

Appendix B: Bibliography

[1] Doval, J.J., Rodríguez, H.: OGC Testbed-14: Security Engineering Report. OGC 18-026r1,Open Geospatial Consortium, https://docs.opengeospatial.org/per/18-026r1.html (2018).