May 2021 OGC API Code Sprint Summary Engineering Report

Publication Date: YYYY-MM-DD

Approval Date: YYYY-MM-DD

Submission Date: YYYY-MM-DD

Reference number of this document: OGC 21-042

Reference URL for this document: http://www.opengis.net/doc/PER/202105APISprintER-ID

Category: OGC Public Engineering Report

Editor: Gobe Hobona

Title: May 2021 OGC API Code Sprint Summary Engineering Report

---

OGC Public Engineering Report

LICENSE AGREEMENT

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 26 to 28 May 2021 to advance the development of the OGC API - Maps draft standard, OGC API - Tiles draft standard, and the OGC API – Styles 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) and Natural Resources Canada (NRCan).

Chapter 2. Executive Summary

This Engineering Report (ER) summarizes the main achievements of the May 2021 OGC API Virtual Code Sprint, conducted between May 26 – 28, 2021. The goal of the code sprint was to progress the development of the draft OGC API standards for Maps, Tiles and Styles. The sprint also sought to help to identify issues and options for addressing those issues.

The objectives of the code sprint were to:

- Develop prototype implementations of OGC API – Maps
- Develop prototype implementations of OGC API – Tiles
- Develop prototype implementations of OGC API – Styles
- 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 have proven to be popular and very effective enabler of rapid software development
- There is an increasing need for optimizing geospatial interoperability between Web APIs
- There is phenomenal adoption of location-handling capabilities in software within and outside of geospatial developer communities

The draft OGC API – Maps specification describes an API that presents data as maps by applying a style. The draft specification enables a client application to request maps as images. This includes the ability to specify or change parameters such as the size of an image and coordinate reference systems at the time of request.

The draft OGC API – Tiles specification describes an API building block that can enable other OGC API implementations to serve maps or tiled feature data divided into individual tiles. The draft specification includes concepts such as tile matrix sets and tile schemes. The draft standard can be used to publish map tiles and tiled feature data (e.g. GeoJSON Vector Tiles and Mapbox Vector Tiles).

The draft OGC API – Styles specification defines a Web API that enables map servers, clients as well as visual style editors, to manage and fetch styles that consist of symbolizing instructions that can be applied by a rendering engine on features and/or coverages.

The outcomes of the sprint can be summarized as follows:

- The Tiles API is reasonably stable. We have seen different interpretations of how to apply styles to collections maps and the dataset maps.
- Evolution of WKSS into common TMS (the ones that are going to be registered). The evolution has taken us to a conclusion that WKSS may no longer be necessary.
- The concept of buildings blocks has been completely demonstrated. The three APIs have been successfully demonstrated together.

- The sprint has shown that a lot that is common can be shared across the APIs i.e. how much OGC API - Common - Part 2 facilitates the client implementation.
- The interaction between Maps, Tiles, Styles worked well. No major issues came up that could not be verified.
- More work needs to be done on Styles in general. e.g. to determine the impact on API resources when styles are used.
- The code sprint focused on the API aspects of the styles but not on the formats of the styles. More work is needed on the formats aspects of the styles (e.g. SymCore).
- While in the Tiles API we have developed a metadata model, in the Maps API there has been less interest in developing a specific metadata model.

The sprint participants considered what the APIs will do to help meet the needs of NMAs. The following is a summary:

- **Providing the public with access to geospatial data and maps**: The OGC APIs will make it easier for the general public to access maps through regular web browser technologies. For example, through OGC API - Maps it is now possible to access a complete map through a basic URL (i.e. no query parameters). OGC API - Tiles will make it easier to publish maps as vector tiles, which are becoming increasingly popular in the National Mapping Agency (NMA) community. The APIs are able to provide data in a way that 2.5D and 3D visualization clients are able to handle.
- **Facilitating analytics**: OGC API - Tiles is able to publish tiled coverage data in such a way that makes it easier to 'stream' coverages for analysis at the screen resolution. This makes it possible to create histograms, vegetation indices, and other analytical reports all at the screen resolution. The flexibility of specifying the origin of the tiles will make it easier to combine regular OGC tiles with other tiles.
- **Reducing barriers to accessing geospatial data**: All of the OGC APIs together make it easier to start with a dataset and then find a way to generate tiles and other resources. The OGC APIs are integrated in a very convenient way. The Styles API makes it possible for NMA’s to publish styles from a central location in a way that is consistent with how they publish data. The integrated environment makes it easier to manage things together.

The code sprint therefore successfully met all of its objectives and achieved its goal of progressing the development of the draft OGC API standards for Maps, Tiles and Styles.

2.1. Document contributor contact points

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

Contacts

Name	Organization	Role
Gobe Hobona	Open Geospatial Consortium	Editor
Adrian Alvarez	APCO	Contributor

Name	Organization	Role
Pelle Arvinder	Carmenta AB	Contributor
Nelio Matos	Connected Places Catapult	Contributor
Gaj Bala	CRTC	Contributor
Panagiotis (Peter) Vretanos	CubeWerx Inc.	Contributor
Keith Pomakis	CubeWerx Inc.	Contributor
Tino Kastbjerg Stigsen	Danish Defense	Contributor
Yasser Othman	EAD	Contributor
Joana Simoes	EarthPulse	Contributor
Antonio Cerciello	EarthPulse	Contributor
Diego Caraffini	Ecere Corporation	Contributor
Patrick Dion	Ecere Corporation	Contributor
Jerome St-Louis	Ecere Corporation	Contributor
James Munroe	Elemental Earth Data Ltd.	Contributor
Hisham Massih	Esri	Contributor
Anne Fitz	Esri	Contributor
Richie Carmichael	Esri	Contributor
Adewale Shittu	Federal University of Technology Akure	Contributor
Serge Lévesque	Fisheries and Oceans Canada	Contributor
Rowan Winsemius	FrontierSI	Contributor
Jeff McKenna	GatewayGeo	Contributor
Francesco Bartoli	Geobeyond Srl	Contributor
Paul van Genuchten	GeoCat BV	Contributor
Gérald Fenoy	GeoLabs	Contributor
Andrea Aime	GeoSolutions	Contributor
Oscar Díaz	GeoSolutions	Contributor
Susie Mielby	Geus	Contributor
Nazih Fino	Global Nomad GIS Services	Contributor
Sara Gholamian	Gozar_e No	Contributor
Charles Heazel	Heazeltech	Contributor

Name	Organization	Role
Clemens Portele	interactive instruments GmbH	Contributor
Muhammed Mete	İstanbul Technical University	Contributor
Berke Şentürk	ITU	Contributor
Zack Zhang	JLL	Contributor
Lorena Hernández	European Commission - Joint Research Centre	Contributor
Bryan Evans	Kinder Institute at Rice University	Contributor
Chris Gagnon	Kongsberg Geospatial	Contributor
Eric Tse	Lexco Limited	Contributor
Philippe Pinheiro	Luxembourg Institute of Science and Technology	Contributor
Rajveer Shekhawat	Manipal University Jaipur	Contributor
Tom Kralidis	Meteorological Service of Canada	Contributor
Gonzalo Nogueras	MetOffice	Contributor
Ali Chettih	Montefiore IT	Contributor
Cameron Wilson	Natural Resources Canada	Contributor
Ryan Ahola	Natural Resources Canada	Contributor
Ahmed Ragab	Natural Resources Canada	Contributor
Bruno Kinoshita	NIWA	Contributor
Azadeh Ashoori	Natural Resources Canada	Contributor
Pradeep Alva	National University of Singapore	Contributor
Scott Simmons	Open Geospatial Consortium	Contributor
Scott Serich	Open Geospatial Consortium	Contributor
Angelos Tzotsos	Open Source Geospatial Foundation	Contributor

Name	Organization	Role
Michael Gordon	Ordnance Survey	Contributor
Chris Holmes	Planet	Contributor
Tim Schaub	Planet	Contributor
Basile Goussard	Promethee	Contributor
Tarron Newman	Red Helmet Technology	Contributor
Senthil Rajrndran	RMSI Pvt Ltd	Contributor
Yohann Hazan	SDIS33	Contributor
Darrel Ronald	Spatiomatics	Contributor
Davince Koyo	Synergetic systems	Contributor
Núria Julià Selvas	UAB-CREAF	Contributor
Joan Maso	UAB-CREAF	Contributor
Ingrid Santana	UFMG	Contributor
Matthew Walker	UK Defence Science and Technology Laboratory	Contributor
Paul Walsh	UK Defence Science and Technology Laboratory	Contributor
Nick Bennett	UK Defence Science and Technology Laboratory	Contributor
Jonathan Lewis	UK Hydrographic Office	Contributor
Pablo Zader	UNC	Contributor
Andres Herrera	Univalle	Contributor
Joseph Olusina	University of Lagos	Contributor
Amy Youmans	US Army Geospatial Center	Contributor
Jeff Harrison	US Army Geospatial Center	Contributor
Huajun Zhang	US Census	Contributor
Ujjwal Yadav	Uttar Pradesh Remote Sensing Application Center	Contributor

2.2. Foreword

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

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

Chapter 3. References

The following normative documents are referenced in this document.

- OGC: OGC 06-042, OpenGIS Web Map Service (WMS) Implementation Specification 1.3.0 (2006)
- OGC: OGC 05-078r4, Styled Layer Descriptor, Version 1.1 (2007)
- OGC: OGC 19-072, draft OGC API - Common - Part 1: Core candidate standard, http://docs.ogc.org/DRAFTS/19-072.html
- OGC: OGC 20-058, draft OGC API - Maps - Part 1: Core candidate standard, http://docs.ogc.org/DRAFTS/20-058.html
- OGC: OGC 20-057, draft OGC API - Tiles - Part 1: Core candidate standard, http://docs.ogc.org/DRAFTS/20-057.html
- OGC: OGC 20-009, draft OGC API - Styles - Part 1: Core candidate standard, http://docs.ogc.org/DRAFTS/20-009.html
- IETF: RFC-7946 The GeoJSON Format (2016)

Chapter 4. Terms and definitions

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

- **API**

  An Application Programming Interface (API) is a standard set of documented and supported functions and procedures that expose the capabilities or data of an operating system, application, or service to other applications (adapted from ISO/IEC TR 13066-2:2016).

- **coordinate reference system**

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

- **OpenAPI Document**

  A document (or set of documents) that defines or describes an API. An OpenAPI definition uses and conforms to the OpenAPI Specification (https://www.openapis.org)

- **Web API**

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

4.1. Abbreviated terms

- **API** Application Programming Interface
- **CRS** Coordinate Reference System
- **OGC** Open Geospatial Consortium
- **SLD** Styled Layer Descriptor
- **WMS** Web Map Service
- **WMTS** Web Map Tile Service

Chapter 5. Introduction

This Engineering Report (ER) summarizes the main achievements of the May 2021 OGC API Virtual Code Sprint, conducted between May 26 – 28, 2021. The sprint had been organized to advance the development of the draft OGC API - Maps, OGC API - Tiles and OGC API - Styles standards. 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.

The code sprint was sponsored by Ordnance Survey (OS) and Natural Resources Canada (NRCan).

5.1. User Needs and Use Cases

To help the sprint participants prioritise their efforts, the sprint organisers invited Natural Resources Canada (NRCan) to outline User Needs from NRCan's perspective as a National Mapping Agency (NMA). This section summarizes the user needs and relates them to use cases envisaged for the OGC APIs in focus for the sprint.

5.1.1. Introduction to Natural Resources Canada

Natural Resources Canada (NRCan) is a part of the Federal Government of Canada responsible for natural resources, energy, minerals and metals, forests, earth sciences, mapping, and remote sensing.

The broad mandate of NRCan is to Enhance responsible development and use of Canada's natural resources and improve the competitiveness of Canada's natural resources products within many areas such as Mapping Forestry, Mining/Geology, Energy and Energy Efficiency, Earth Observation. Geospatial data plays a key role in all of the aforementioned areas, hence NRCan's interest in the development of OGC APIs.

As the NMA of Canada, NRCan plays a critical nation-wide role in the distribution of authoritative geospatial data products, including cartographic products such as maps.

5.1.2. The Priorities that drive the Need for APIs

There are specific priorities that drive what NRCan would like to see from OGC APIs. The key priorities that drive what NRCan would like to see from OGC APIs include for example: Climate Change, Response to disasters/extreme events, the Arctic, Trade, Sovereignty, and Indigenous Reconciliation. The government has a strong desire to have collaboration and innovation within government processing benefiting Canadian society broadly. Innovation provides a bridge between the government's internal focus areas and how these will apply within Canada and its position in the world. So, indeed, all the OGC APIs that are being developed through this sprint will, in the future, help to benefit society.

5.1.3. Specific Needs

OGC APIs have a substantial role to play in future NMAs. At NRCan, this role is likely to involve the development and provision of microservices in order to support the delivery of geospatial data, maps and analytics. This role can be described in terms of the following needs:

- **Providing the public with access to geospatial data and maps**: This is a key function of an NMA. NRCan therefore sees OGC APIs as having the potential to help the NMA's to provide open data in a way that conforms to FAIR principles (Findable, Accessible, Interoperable, and Reusable). This enables the members of the public to make use of the geospatial data and maps as they see fit (e.g. in support of other parts of the community or economy).
- **Facilitating analytics**: Making geospatial a fundamental part of national decision making requires consideration of how to optimize the use of location information. So by focusing firstly on analytics, geospatial experts can be enabled to help others, then those experts could make better decisions through geospatial information analytics.
- **Reducing barriers to accessing geospatial data**: Geospatial data has gotten more accessible over the past decade. However, there has also been a significant increase in the demand for knowledge and expertise in all sorts of development to use geospatial information.

5.1.4. Sprint Area of Interest

For demonstration purposes, Sprint participants were encouraged to publish specific data and maps for the following Areas of Interest (AOI):

Europe: The area around Bournemouth, England, within the extent specified by this GeoJSON file [https://github.com/opengeospatial/ogcapi-code-sprint-2021-05/blob/main/BournemouthAOI.geojson] or this WKT string in EPSG:4326 coordinates `POLYGON -2.13384466616954 50.5343261657655,-2.14712951953212 50.822458640394,-1.77636133932212 50.8243659606517,-1.75884948716236 50.539699354356,-2.13384466616954 50.5343261657655`.

North America: Red River of the North, within the extent specified by this GeoJSON file [https://github.com/opengeospatial/ogcapi-code-sprint-2021-05/blob/main/RedRiverAOI.geojson] or this WKT string in EPSG:4326 coordinates `POLYGON -97.8656275465241 50.1994331527875,-97.8290574091464 48.9215621457706,-96.475962326173 48.9305725567791,-96.4851048605174 50.2082107872824,-97.8656275465241 50.1994331527875`.

The datasets that were recommended for the code sprint included:

5.1.5. Ordnance Survey datasets for the Sprint's Europe AOI

- OS Open Zoomstack data product [https://os.uk/business-government/products/open-zoomstack]: A comprehensive basemap of the United Kingdom showing coverage from national level right down to street detail.
- OS Open Zoomstack stylesheets [https://github.com/OrdnanceSurvey/OS-Open-Zoomstack-Stylesheets]: These are OS Open Zoomstack stylesheets encoded in OGC SLD, Esri LYR, QGIS QML and Mapbox GL Styles formats.

NRCan datasets for the Sprint's North America AOI

- High Resolution Digital Elevation Model (HRDEM) [https://open.canada.ca/data/en/dataset/957782bf-847c-4644-a757-e383c0057995]: Complete coverage of the Canadian territory in a Digital Terrain Model (DTM), a Digital Surface Model (DSM) and other derived data.
- Canada Base Map Transportation (CBMT) [https://open.canada.ca/data/en/dataset/296de17c-001c-4435-8f9a-f5acab632e85]: Base map with a focus on transportation networks. Available as a tiled web map service.
- National Hydrographic Network (NHN) [https://www.nrcan.gc.ca/science-and-data/science-and-research/earth-sciences/geography/topographic-information/geobase-surface-water-program-geeau/national-hydrographic-network/21361]: Data about Canada's inland surface waters.
- RADARSAT-1 [https://www.asc-csa.gc.ca/eng/satellites/radarsat1/Default.asp]: An operational radar satellite system, equiped with a Synthetic Aperture Radar (SAR) instrument, capable of acquiring images of the Earth day or night, in all weather and through cloud cover, smoke and haze.
- Open Maps [http://open.canada.ca/en/open-maps]: Approximately 4600 open geospatial datasets for Canada.

5.2. Participants

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

- APCO
- Carmenta AB
- Connected places catapult
- CRTC
- CubeWerx Inc.
- Danish Defense
- EAD
- EarthPulse
- Ecere Corporation
- Elemental Earth Data Ltd.
- Esri
- Federal University of Technology Akure
- Fisheries and Oceans Canada
- FrontierSI
- GatewayGeo
- Geobeyond Srl
- GeoCat BV

- GeoLabs
- GeoSolutions
- Geus
- Global Nomad GIS Services
- Heazeltech
- interactive instruments GmbH
- İstanbul Technical University
- ITU
- JLL
- European Commission - Joint Research Centre
- Kinder Institute at Rice University
- Kongsberg Geospatial
- Lexco Limited
- Luxembourg Institute of Science and Technology
- Manipal University Jaipur
- Meteorological Service of Canada
- Met Office
- Montefiore IT
- Natural Resources Canada
- NIWA
- National University of Singapore
- Open Source Geospatial Foundation
- Ordnance Survey
- Planet
- Promethee
- Red Helmet Technology
- RMSI Pvt Ltd
- SDIS33
- Spatiomatics
- Synergetic systems
- UAB-CREAF
- UFMG
- UK Defence Science and Technology Laboratory
- UK Hydrographic Office
- Unc

- Univalle
- University of Lagos
- US Army Geospatial Center
- US Census
- uttar pradesh remote sensing application center

Chapter 6. Architecture

6.1. High Level Overview

The focus of the sprint was on support of the development of the draft OGC API - Maps [https://ogcapi.ogc.org/maps], OGC API - Tiles [https://ogcapi.ogc.org/tiles] and OGC API - Styles [https://ogcapi.ogc.org/styles] standards. Implementations of these draft standards were deployed in participants' own infrastructure in order to build a solution with the architecture shown below in Figure 1.

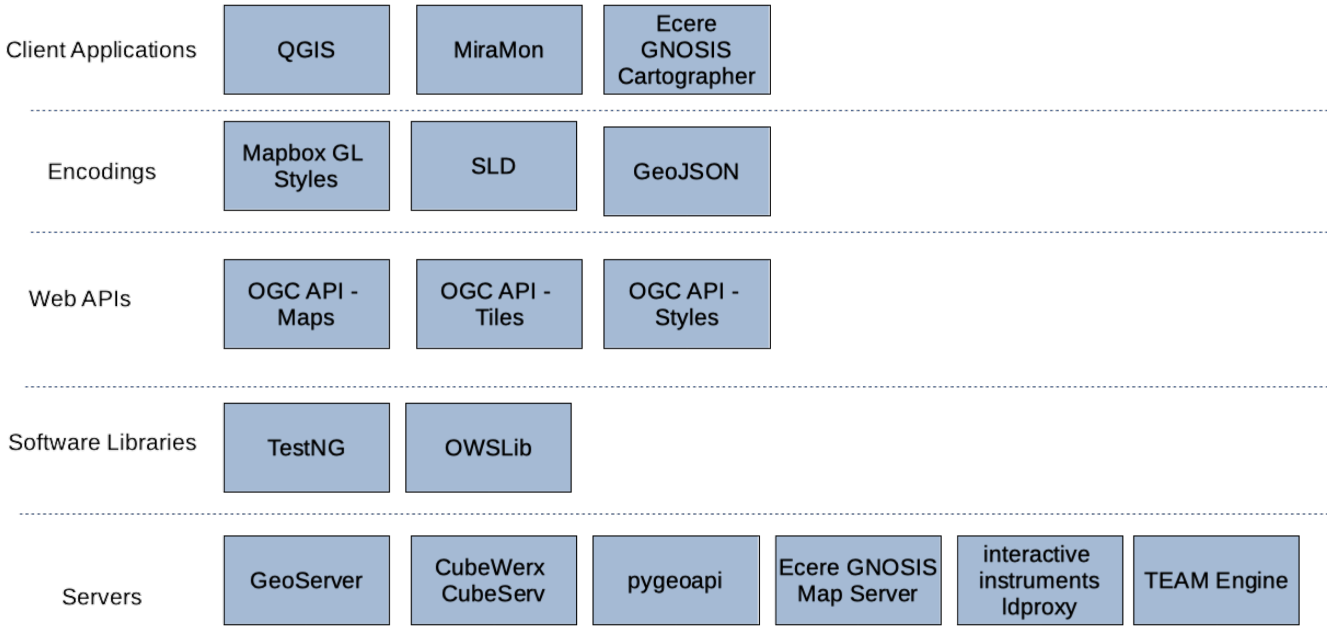


Figure 1. High level overview of the architecture implemented during the sprint

As illustrated, the sprint architecture was designed with the view of enabling client applications to connect to different servers that implement OGC APIs. The servers were provisioned with maps, tiled feature data (colloquially named vector tiles), map tiles, tiled coverage data, and styles.

6.2. Candidate Standards

6.2.1. OGC API - Maps

The draft OGC API - Maps standard describes an API that presents maps portraying data that has been rendered according to a style. The maps served by implementations of the draft OGC API - Maps standard are retrieved as images of any size, generated on-the-fly, and with the styling determined by the client application. The draft standard can be considered the successor to the widely implemented WMS standard. The draft OGC API – Maps standard is a multipart standard that includes a Core (Part 1) and extensions that are planned to be developed in the future.

6.2.2. OGC API - Tiles

The draft OGC API - Tiles standard describes an API that implements the OGC Two Dimensional Tile Matrix Set (TMS) [http://docs.opengeospatial.org/is/17-083r2/17-083r2.html] standard to enable access to tiled resources on the Web. The TMS standard defines the rules and requirements for a tile matrix set as a way to index space based on a set of regular grids defining a domain (tile matrix) for a limited list of scales in a CRS. The draft OGC API – Tiles standard is a multipart standard that includes a Core (Part 1) and extensions that are planned to be developed in the future.

6.2.3. OGC API - Styles

OGC API - Styles describes the interface and exchange of styling parameters and instructions. The construction of symbology components of styles is addressed in the OGC Symbology Conceptual Model: Core Part [https://docs.ogc.org/is/18-067r3/18-067r3.html] standard and multiple OGC and other style encoding standards.

Chapter 7. Results

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

7.1. Implementations

7.1.1. CubeWerx Inc.

The CubeWerx server ("cubeserv") is an executable supports a wide variety of back ends including Oracle, MariaDB, SHAPE files, etc. It also supports a wide array of service-dependent output formats (e.g. GML, GeoJSON, Mapbox Vector Tiles, MapMP, etc.) and coordinate reference systems. At the time of publishing this engineering report, the CubeSERV OGC API - Features Server product is certified OGC compliant [https://www.ogc.org/resource/products/details/?pid=1601] to the OGC API - Features - Part 1: Core standard. The cubeserv product is implemented in C and currently implements the following OGC specifications:

- Multiple conformance classes and recommendations of the draft OGC API - Maps - Part 1: Core specification.
- Multiple conformance classes and recommendations of the draft OGC API - Tiles - Part 1: Core specification.
- Multiple conformance classes and recommendations of the draft OGC API - Styles - Part 1: Core specification.
- Multiple conformance classes and recommendations of the draft OGC API - Records - Part 1: Core specification.
- Multiple conformance classes and recommendations of the draft OGC API - Coverages - Part 1: Core specification.
- Multiple conformance classes and recommendations of the draft OGC API - Processes - Part 1: Core specification.
- All conformance classes and recommendations of the OGC API - Features - Part 1: Core standard.
- Multiple versions of the Web Map Service (WMS), Web Processing Service (WPS), Web Map Tile Service (WMTS) and Web Feature Service (WFS) standards
- A number of other "un-adopted" OGC web services including the Testbed-12 Web Integration Service, OWS-7 Engineering Report - GeoSynchronization Service, Web Object Service Implementation Specification.

7.1.2. Ecere Corporation

The GNOSIS Map Server is written in the eC programming language and supports multiple OGC API specifications. GNOSIS Map Server supports multiple encodings including GNOSIS Map Tiles (which can contain either vector data, gridded coverages, imagery, point clouds or 3D meshes), Mapbox Vector Tiles, GeoJSON, GeoECON, GML and MapML. An experimental server is available online at https://maps.ecere.com/ogcapi and has been used in multiple OGC Innovation Program initiatives. At the time of publishing this engineering report, the GNOSIS Map Server 1.0 product is certified

OGC compliant [https://www.ogc.org/resource/products/details/?pid=1670] to the OGC API - Features - Part 1: Core standard. For this code sprint, the server was configured to offer an endpoint supporting:

- Multiple conformance classes and recommendations of the draft OGC API - Maps - Part 1: Core specification.
- Multiple conformance classes and recommendations of the draft OGC API - Tiles - Part 1: Core specification.
- Multiple conformance classes and recommendations of the draft OGC API - Styles - Part 1: Core specification.
- Multiple conformance classes and recommendations of the draft OGC API - Coverages - Part 1: Core specification.
- Multiple conformance classes and recommendations of the draft OGC API - Processes - Part 1: Core specification.
- Multiple conformance classes and recommendations of the OGC API - Features - Part 1: Core Standard.

7.1.3. GeoSolutions

GeoSolutions deployed an instance of the GeoServer product. GeoServer is a Java-based software server that allows users to view and edit geospatial data. Using open standards by the OGC, GeoServer allows for great flexibility in map creation and data sharing. For this code sprint, the server was configured to offer an endpoint supporting:

- Multiple conformance classes and recommendations of the draft OGC API - Maps - Part 1: Core specification.
- Multiple conformance classes and recommendations of the draft OGC API - Tiles - Part 1: Core specification.
- Multiple conformance classes and recommendations of the draft OGC API - Styles - Part 1: Core specification.
- Multiple conformance classes and recommendations of the OGC API - Features - Part 1: Core Standard.

7.1.4. Meteorological Service of Canada

The Meteorological Service of Canada (MSC) is a federal deparment of the Government of Canada. MSC works in close collaboration with NRCan to publish weather, climate and water data to the general public and to help meet Canada's international commitments with the World Meteorological Organization (WMO).

The data is published using an instance of the pygeoapi product. The pygeoapi product is an open source Python server implementation of the OGC API suite of standards. The product supports the microservices approach and allows scalability. At the time of publishing this engineering report, the pygeoapi 0.9.0 product is certified OGC compliant [https://www.ogc.org/resource/products/details/?pid=1663] to the OGC API - Features - Part 1: Core standard. MSC deployed an instance of pygeoapi and worked with OSGeo colleagues from GeoCat BV and Geobeyond to implement an endpoint supporting:

- The draft OGC API - Maps - Part 1: Core specification.
- The draft OGC API - Tiles - Part 1: Core specification.
- The draft OGC API - Styles - Part 1: Core specification.

During the code sprint, MSC also extended the OWSLib library to enable the library to perform as a client application for OGC APIs for coverages, maps, features and styles.

7.1.5. Open Source Geospatial Foundation

The Open Source Geospatial Foundation (OSGeo) is a not-for-profit organization whose mission is to foster global adoption of open geospatial technology by being an inclusive software foundation devoted to an open philosophy and participatory community driven development. The foundation consists of projects that develop open source software products. Multiple OSGeo projects and community projects participated in the sprint, for example GeoServer, pygeoapi and OWSLib.

7.1.6. Universitat Autònoma de Barcelona

The Centre for Ecological Research and Forestry Applications (CREAF) is a public research institute created in 1987 and located in Catalonia. It is part of the Autonomous University of Barcelona (UAB). CREAF deployed an instance of the MiraMon Map Server. The MiraMon Map Server is a CGI application encoded in C language that is part of the MiraMon Geographic Information System (GIS) & Remote Sensing (RS) suite. The software originally started 10 years ago as a WMS server in support of the Catalan Administration and CREAF data services. Currently the server implements WMS, WMTS and partially implements WFS and WCS. It also partially implements the OGC Sensor Observation Service (SOS) standard. It also includes prototype support for the draft OGC API - Maps and OGC API - Tiles specifications. In order to perform efficiently, it requires a process preparing the data to be offered. The server can interoperate with other vendors' clients. When combined with the MiraMon Map Client, the server offers additional functionality, including functionality recently developed for the Catalan Data Cube. The MiraMon Map Client is built using client-side JavaScript and can therefore run on any web browser.

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 in Figure 2. The Gitter channel can be found at https://gitter.im/ogc-developer/Sprints

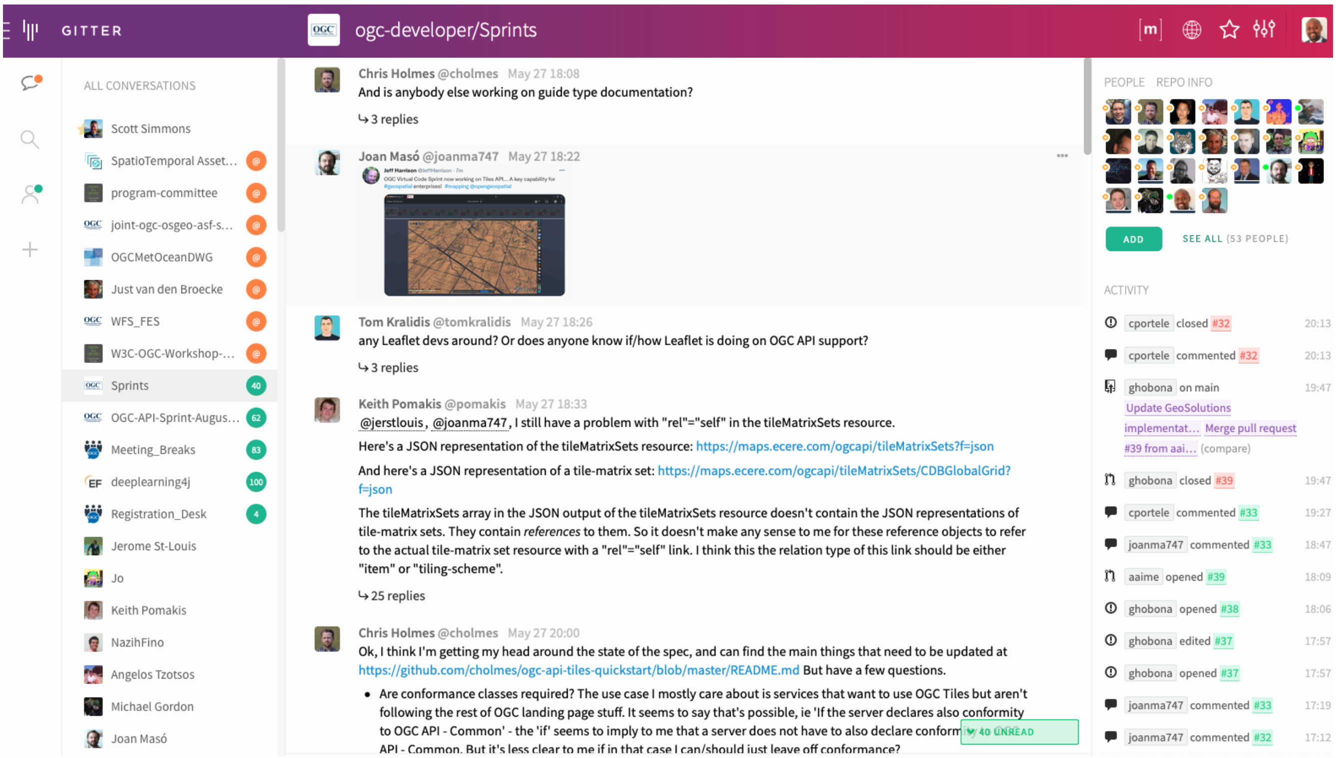
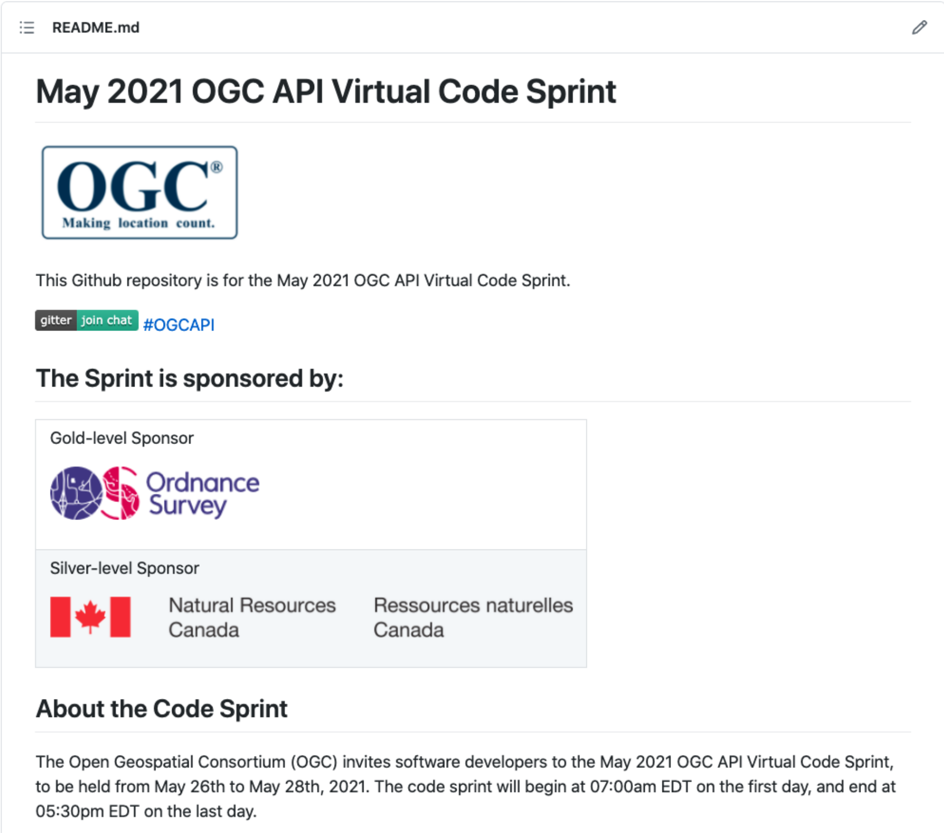


Figure 2. Screenshot of Gitter channel

A screenshot of the GitHub repository is shown below in Figure 3. The GitHub repository can be found at https://github.com/opengeospatial/ogcapi-code-sprint-2021-05


README.md

May 2021 OGC API Virtual Code Sprint

OGC® Making location count.

This Github repository is for the May 2021 OGC API Virtual Code Sprint.

gitter join chat #OGCAPI

The Sprint is sponsored by:

Gold-level Sponsor

Ordnance Survey

Silver-level Sponsor

Natural Resources Canada Ressources naturelles Canada

About the Code Sprint

The Open Geospatial Consortium (OGC) invites software developers to the May 2021 OGC API Virtual Code Sprint, to be held from May 26th to May 28th, 2021. The code sprint will begin at 07:00am EDT on the first day, and end at 05:30pm EDT on the last day.

Figure 3. Screenshot of GitHub repository

The next subsections provide a summary of the discussion.

8.1. OGC API support for different approaches to organising styles, layers and data sources

There was a discussion [https://github.com/opengeospatial/ogcapi-code-sprint-2021-05/issues/15] on how OGC APIs could support different approaches to organising styles, layers and data sources. The sprint participants observed that there are two different approaches of how SLD is used by implementations. One approach involves having the SLD at the top level, whereas the other approach does not. The sprint participants were asked the question "Should we have two conformance classes for the different approaches?". Issue#5 on the OGC API - Styles repository [https://github.com/opengeospatial/ogcapi-styles/issues/5] includes a proposal how to change the specification to better support SLD as an encoding. After discussing the topic during the sprint, the sprint participants made the recommendation that: There is a need to have two separate conformance classes to support the different paradigms.

8.2. Support for legends

There was a discussion [https://github.com/opengeospatial/ogcapi-code-sprint-2021-05/issues/17] on the need for support for legends. Currently legends are not supported in OGC API - Maps nor OGC API - Styles, however it is expected that client applications will need or want to present a legend for the information on a map. To facilitate the discussion, CubeWerx provided the paths to a Cubeserv implementation of a legend capability. The paths are listed in Annex A. A screenshot from a prototype built using pygeoapi are shown in Figure 4.

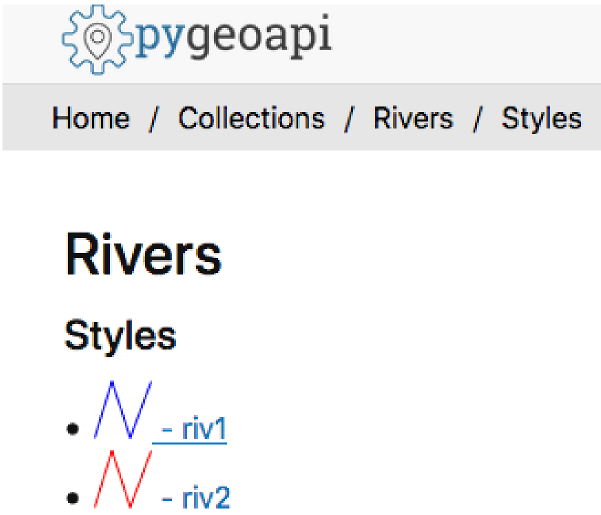


Figure 4. Screenshot from a legend generated using pygeoapi

The sprint participants also noted that if a legend is a resource in OGC API - Maps, it could be a bitmap graphic of a legend. However, if it is a resource in OGC API - Styles, then it should be a data resource, so that clients can render the legend as they wish and that supports combining information from multiple legends.

8.3. Changes to a style with multiple occurrences in an API

There was a discussion [https://github.com/opengeospatial/ogcapi-code-sprint-2021-05/issues/18] regarding how changes to a style with multiple occurrences in an API are executed. The sprint participants were asked the question "If a style `foo` is used in several places in an API (that is, there are several resources where the path includes `/styles/foo` somewhere), does a PUT/DELETE on that style affect all occurrences of the style or only the one on which the operation is executed? In other words, is it only one resource with multiple URIs or are these separate resources?". For example, for an implementation that has a "night" style at `/styles/night`, the participants considered whether changes to that style should be propagated to other instances of the same style e.g. `/collection/foo/styles/night`.

After discussing this topic, the sprint participants proposed adding the following statement to the OGC API - Styles specification:

> A service may implement HTTP PUT and/or HTTP DELETE methods for the `/collections/{collectionId}/styles/{styleId}` endpoints. If implemented, these methods shall have the effect of adding/replacing or deleting the definition of the specified style as it pertains to the specified collection. E.g., after a DELETE of `/collections/foo/styles/night`, a future GET of `/collections/foo/styles/night` should return a 404 Not Found, whereas a future GET of `/styles/night` should return a style definition that lacks any mention of collection foo.

8.4. Multiple dimensions in OGC API - Maps

In the MetOcean context, there is a need to consider addressing dimensions as part of rendering in OGC API - Maps. This need is similar, in part, to the need that triggered the development of the OGC Best Practice for using Web Map Services (WMS) with Time-Dependent or Elevation-Dependent Data. So the Best Practice document can be consulted for information on the approach taken by the previous generation of OGC web service standards. A given collection would need to advertise its dimensions and their relevant extents (e.g. in a MetOcean/NWP context there could be dimensions for the model run/reference time, forecast time, elevation/pressure level, etc.)

It was noted that the Maps API already supports `datetime` and `subset` parameters, as demonstrated by Ecere's implementation from this sprint and the Hexagon implementation from a previous sprint. Example queries from the Ecere implementation are below:

https://maps.ecere.com/ogcapi/collections/blueMarble/styles/default/map?datetime=2004-07

https://maps.ecere.com/ogcapi/collections/blueMarble/styles/default/map?subset=time(2004-11)

The sprint participants observed that the approach should ideally be the same for non-static tiles or maps as the approach taken for the underlying source data.

If the source data is features, support parameters like datetime, filter, collections, properties, etc. like on the OGC API - Features resources to filter the features and reduce the properties that are returned. This approach was demonstrated by the ldproxy product and was part of the experiments in the OGC Vector Tiles Pilot 2. An example is below:

https://demo.ldproxy.net/daraa/tiles/WebMercatorQuad/11/827/1229?
collections=AgricultureSrf,TransportationGroundCrv&datetime=2012-02-12T00:00:00Z/..&
properties=F_CODE

8.5. Styles, Tiles: Metadata review

The sprint participants conducted a review of the metadata elements specified by OGC APIs for Maps, Tiles, and Styles. The review uncovered that there is significant overlap between the metadata for a style and for a tileset. At the same time, there are issues and differences that should be addressed. Harmonization of the metadata elements, should also be consistent with OGC API - Records. Specific observations and proposals are listed below:

- "title" and "version" are the same.
- "abstract" vs "description". Proposal: use "description".
- "keywords": Styles API uses strings, Tiles API uses a more complex model. Note that the Records API uses strings for "keywords", too. For controlled vocabularies, "themes" is used. Proposal: restrict "keywords" to strings.
- The Styles and Tiles API use "pointOfContact", whereas the Records API uses "contactpoint". Proposal: agree on a single name.
- The Styles and Tiles APIs use "accessConstraints" with a fixed list from the intelligence domain. At the same time more generally useful information like "license" is missing. Proposal: add

"license" and drop "accessConstraints". Communities that need the "accessConstraints" elements can always add it through an extension.

- The Tiles API has "publisher", whereas the Styles API does not. The Records API has it, too. Proposal: add "publisher" to the Styles API.
- The Styles API has a fixed "scope" (value "style"), whereas the Tiles API does not. The Records API instead has "type", a URI. Proposal: harmonize.
- The Styles API has various dates in "dates", whereas the Tiles API has them in "date". The Records API only has "created" and "updated", but not embedded in a data type. Proposal: Follow the approach from the Records API.
- The Styles and Tiles APIs both have "layers", but with different content.
- * "abstract" vs "description". Proposal: use "description. See #31.
- * The Styles API has "type" (point, line, polygon, geometry, raster), whereas the Tiles API has "dataType" (vector, coverage, map) and "geometryType" (points, lines, polygons). Proposal: use "dataType" and "geometryType".
- * The Styles API has "attributes" (the OpenAPI 3.0 schema for each attribute), whereas the Tiles API has "propertiesSchema" (a subset of JSON Schema describing an object where each attribute is a property plus some extensions to JSON Schema like "observedProperty" or "uom"). Proposal: Use standard JSON Schema without restrictions. Add a recommendation for a profile, similar to the approach taken by Features for Queryables.
- There is "mediaType" as a string, but the description implies that there can be multiple media types. Proposal: Either change the element to an array, or revise the description to state a maximum occurrence of 1.

There was also an observation made regarding the Tiles API, that the use of scaleDenominator, cellSize and/or the tileMatrix could lead to confusion. This is because every client would have to be able to handle all of them and convert them to the internal mechanism that the client uses. There was a suggestion to pick one to make it easier for clients.

It was acknowledged across the sprint that there is a need to request feedback from Client implementors regarding whether to keep scaleDenominator, cellSize, tileMatrix elements, and that there are several benefits to keeping all of the elements. The sprint participants recommended that, for the ETS, if the server provides the scaleDenominator, cellSize, tileMatrix elements, the ETS should check if they are consistent to a significant number of digits (e.g. at least 12 digits).

After discussing the results of the metadata review, the sprint participants noted that there is a lot of value in dropping the 'accessConstraints' field and going with 'license'. Whereas 'accessConstraints' is used more in implementations of ISO 19115, the term 'license' is used in Dublin Core and DCAT implementations. Therefore, the term 'license' may be the more general term to use - between the two terms.

The sprint participants recommended that a review of the metadata in OGC API - Common, - Maps , -Tiles be carried out and similarities/differences should be discussed in a future multi-SWG meeting.

8.6. Suggested styleId when creating a style

The Styles API extends on the generic Create/Replace/Delete requirements class, i.e., styles are always created with POST and the server assigns the styleId. The server is free to parse the submitted stylesheet to determine a meaningful `styleId`, if it wants, but that is not possible in style encodings that do not include an identifier.

The Sprint participants noted that this potentially could be addressed by adding an HTTP header that clients could use to suggest a `styleId` with a HTTP POST request. The server would be free to ignore the suggestion. It was also noted that HTTP PUT could also be used for a similar purpose, in accordance with RFC 2616, as demonstrated by the CubeWerx implementation's acceptance of an HTTP PUT `/styles/{styleId}` request to create (or replace) a style with a specific ID. This issue was highlighted for further discussion in the SWGs because HTTP PUT is the typical way of allowing a client to create a resource with a client-defined URI, whereas HTTP POST is for situations where the server should assign a URI.

8.7. Summary of Code Sprint Outcomes

This section presents a summary of the outcomes of the sprint.

8.7.1. Immediate Lessons

- The Tiles API is reasonably stable. We have seen different interpretations of how to apply styles to collections maps and the dataset maps.
- Evolution of WKSS into common TMS (the ones that are going to be registered). The evolution has taken us to a conclusion that WKSS may no longer be necessary.
- The concept of buildings blocks has been completely demonstrated. The three APIs have been successfully demonstrated together.
- The sprint has shown that a lot that is common can be shared across the APIs i.e. how much OGC API - Common - Part 2 facilitates the client implementation.
- The interaction between Maps, Tiles, Styles worked well. No major issues came up that could not be verified.
- More work needs to be done on Styles in general. e.g. to determine the impact on API resources when styles are used.
- The code sprint focused on the API aspects of the styles but not on the formats of the styles. More work is needed on the formats aspects of the styles (e.g. SymCore).
- While in the Tiles API we have developed a metadata model, in the Maps API there has been less interest in developing a specific metadata model.

8.7.2. Implications for NMAs

The sprint participants considered what the APIs will do to help meet the needs of NMAs. The following is a summary.

- **Providing the public with access to geospatial data and maps**: The OGC APIs will make it easier for the general public to access maps through regular web browser technologies. For example, through OGC API - Maps it is now possible to access a complete map through a basic URL (i.e. no query parameters). OGC API - Tiles will make it easier to publish maps as vector tiles, which are becoming increasingly popular in the NMA community. The APIs are able to provide data in a way that 2.5D and 3D visualization clients are able to handle.

- **Facilitating analytics**: OGC API - Tiles is able to publish tiled coverage data in such a way that makes it easier to 'stream' coverages for analysis at the screen resolution. This makes it possible to create histograms, vegetation indices, and other analytical reports all at the screen resolution. The flexibility of specifying the origin of the tiles will make it easier to combine regular OGC tiles with other tiles.
- **Reducing barriers to accessing geospatial data**: All of the OGC APIs together make it easier to start with a dataset and then find a way to generate tiles and other resources. The OGC APIs are integrated in a very convenient way. The Styles API makes it possible for NMA's to publish styles from a central location in a way that is consistent with how they publish data. The integrated environment makes it easier to manage things together.

Chapter 9. Conclusions

The code sprint facilitated the development and testing of prototype implementations of the OGC API - Maps draft standard, OGC API - Tiles draft standard, and the OGC API – Styles draft standard. Further, the code sprint also enabled the participating developers to provide feedback to the editors of OGC standards. The code sprint was also able to identify ways through which the OGC APIs would help to meet the needs of NMAs. The code sprint therefore met all of its objectives and achieved its goal of progressing the development of the draft OGC API standards for Maps, Tiles and Styles.

9.1. Future Work

The sprint participants considered how the sprint’s outcomes could be incorporated into future OGC Standards Program and Innovation Program activities.

9.2. Next Steps for the Innovation Program

There is a need to:

- experiment with multidimensional data support in OGC APIs.
- explore how to turn legends into real data(objects) that can be combined by the client (e.g. asking a client to provide the elements that are in a legend).
- research how simple a structure needs to be to meet the needs for a legend while also being easily implementable.
- experiment with coverage tiles, as they are becoming increasingly important (e.g. in support of rendering DSM’s and DDIL environments). Strategies for identifying suitable sizes of the tiles needs to tested/researched.
- experiment with non-grid coverages (e.g. point clouds).
- explore the possibility of an 'info' capability that supports different data sources and query options (not just retrieval of the value at a point).

9.3. Next Steps on the Standards Program

There is a need to:

- specify a legend conformance class for the OGC API - Maps and OGC API - Tiles draft specifications.
- specify an 'info' conformance class for the OGC API - Maps and OGC API - Tiles draft specifications.
- implement an OGC API - Maps conformance class/extension to support time dependent maps (in a way similar to the WMS Best Practice for Time Dependent) e.g. the subset and datetime parameters.

Appendix A: Prototype Legend Support

The map-level endpoints that the CubeWerx OGC API demo server at https://test.cubewerx.com/cubewerx/cubeserv/demo/ogcapi/Daraa implemented for this are:

`/map/legend` - A legend image showing a graphical representation of one or more collections as they would appear in the corresponding map. e.g.: https://test.cubewerx.com/cubewerx/cubeserv/demo/ogcapi/Daraa/map/legend?transparent=false https://test.cubewerx.com/cubewerx/cubeserv/demo/ogcapi/Daraa/map/legend?scale=150000&transparent=false

`/styles/{styleId}/map/legend` - A legend image showing a graphical representation of one or more collections as they would appear in the corresponding map in the specified style. e.g.: https://test.cubewerx.com/cubewerx/cubeserv/demo/ogcapi/Daraa/styles/Night/map/legend?transparent=false

`/collections{collectionId}/map/legend` - A legend image showing a graphical representation of the specified collection as it would appear in the corresponding map. e.g.: https://test.cubewerx.com/cubewerx/cubeserv/demo/ogcapi/Daraa/collections/TransportationGroundCrv/map/legend?transparent=false

`/collections/{collectionId}/styles/{styleId}/map/legend` - A legend image showing a graphical representation of the specified collection as it would appear in the corresponding map in the specified style. E.g.: https://test.cubewerx.com/cubewerx/cubeserv/demo/ogcapi/Daraa/collections/TransportationGroundCrv/styles/Night/map/legend?transparent=false

Each of these endpoints takes the following optional parameters:

- **transparent** - Whether or not the background of the image should be transparent (when supported by the requested image format).
- **bgcolor** - Hexadecimal red-green-blue color value for the background color. If not specified, the background color specified by the style (0xFFFFFF by default) will be used.
- **textcolor** - Hexadecimal red-green-blue color value for the text color. If not specified, a color that contrasts the background color will be used.
- **scale** - If specified, a legend graphic specific to this scale (expressed as a scale denominator) will be returned. Otherwise, a possibly-composite image providing a legend graphic for each of the scale ranges defined by the style will be returned.
- **pixelSize** - The physical linear size of a display pixel in millimetres. If the display device has non-square pixels, then sqrt(width * height) should be provided. The pixel units and scale rules in a style definition are with respect to a standardized rendering pixel size of 0.28mm. Knowledge of the actual pixel size of the display device will allow the renderer to produce a map with the intended look even if the actual pixel size is significantly different from the standardized rendering pixel size.
- **f** - A token indicating the content type to return. Overrides the HTTP "Accept" header if present. A value of "jop" (content type "image/x-jpegorpng") indicates that either JPEG or PNG should be returned, whichever the server deems to be most appropriate for this particular image.

The CubeWerx implementation also supports legends at the tile-level endpoints (which indicate their zoom level via the {tileMatrixSetId} and {tileMatrix} pathe elements):

`/collections/{collectionId}/map/tiles/{tileMatrixSetId}/{tileMatrix}/legend` - A legend image showing a graphical representation of the specified collection as it would appear in the corresponding map tiles of the specified zoom level. e.g:

https://test.cubewerx.com/cubewerx/cubeserv/demo/ogcapi/Daraa/collections/CultureSrf/map/tiles/smerc/12/legend?transparent=false

`/collections/{collectionId}/styles/{styleId}/map/tiles/{tileMatrixSetId}/{tileMatrix}/legend` - A legend image showing a graphical representation of the specified collection as it would appear in the corresponding map tiles of the specified zoom level in the specified style. E.g.:

https://test.cubewerx.com/cubewerx/cubeserv/demo/ogcapi/Daraa/collections/CultureSrf/styles/Night/map/tiles/smerc/12/legend?transparent=false

CubeWerx also experimented with the following legend endpoint:

`/styles/{styleId}/legend` - A legend image showing a graphical representation of the specified style, broken down by each of the collections that it's capable of rendering.

https://test.cubewerx.com/cubewerx/cubeserv/demo/ogcapi/Daraa/styles/Night/legend?transparent=false

However, this returns practically the same legend as `` `/styles/{styleId}/map/legend `` so it is unclear whether a separate legend endpoint here is warranted.

Appendix B: Revision History

Table 1. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
2021-05-26	G. Hobona	.1	all	initial version
TBA	TBA	TBA	TBA	TBA

Appendix C: Bibliography