OGC API - Environmental Data Retrieval Sprint Engineering Report

Publication Date: 2020-10-22

Approval Date: 2020-06-22

Submission Date: 2020-05-28

Reference number of this document: OGC 20-032

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

Category: OGC Public Engineering Report

Editor(s): Chris Little, Peng Yue, Steve Olson

Title: OGC API - Environmental Data Retrieval Sprint Engineering Report

OGC Public Engineering Report

LICENSE AGREEMENT

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

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

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

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

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

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

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

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

Table of Contents

Chapter 1. Summary

1.1. Subject

The subject of this Engineering Report (ER) is a development Sprint that was held from March 18-20, 2020 to advance the Open Geospatial Consortium (OGC) Environmental Data Retrieval (EDR) Application Programming Interface (API) candidate standard. Due to the widespread of the virus, the Sprint was held virtually by using GoToMeeting teleconferencing facilities of OGC, email and GitHub.

1.2. Executive Summary

The idea of the EDR API is to enable end users and Web developers to conveniently and easily retrieve required data from big data stores, using current Web technologies and a significantly reduced learning curve, with unnecessary details initially hidden from the service endpoint. The API queries can be considered discrete sampling geometries into the non-sparse relatively persistent data store.

The EDR API Sprint was to provide feedback based on the current EDR API candidate specification. The specific objectives were discussed prior to and in the opening session of the Sprint, which includes:

- Verify and validate requirements and methods for the Query and Filter operations of the EDR API;
- Prototype rapidly other geometry types for the EDR API (partitioned grids or tiles, vertical profiles, and/or trajectories/corridors);
- Develop client-side value-added applications that consume data from UK Met Office and US NWS prototype EDR API implementations; and
- Make progress on conformance testing of EDR API parts based on queries.

There were about two dozen individuals that participated in the event. Significant progress was made advancing the specification in several key functional areas. A number of GitHub issues were created and discussed throughout the Sprint. The Sprint was considered very successful with clarifications and enhancements of the specification agreed. Commonality and differences with the other OGC APIs under development were identified.

Based on the results and findings of the Sprint, this engineering report makes the following recommendations:

- Expand metadata offerings, consider what Metadata frameworks should be used, recommended, or mandated;
- Test other sampling geometry types;
- Investigate how to integrate with other APIs (features/coordinates, maps);
- Incorporate the needed security aspects. This needs to be coordinated with the Security DWG and OWS Common - Security SWG;

- Incorporate a pub/sub mechanism with the EDR API specification; and
- Investigate how to align with decision impact groups like SmartCity, others.

1.3. Document contributor contact points

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

Contacts

Name	Organization	Role
Chris Little	UK Met Office	Editor
Steve Olson	US National Weather Service	Editor
Peng Yue	Wuhan University	Editor

1.4. 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 2. 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.

- Open API Initiative: OpenAPI Specification 3.0.3
- IETF RFC2616, HTTP/1.1
- IETF RFC 2818, HTTP Over TLS
- IETF RFC 3339, Date and Time on the Internet: Timestamps
- IETF RFC 3896, Uniform Resource Identifier (URI): Generic Syntax
- IETF RFC 7946, The GeoJSON Format
- IETF RFC 8288, Web Linking
- IETF RFC 2413, Dublin Core Metadata for Resource Discovery
- W3C: HTML5, W3C Recommendation
- Simple Feature Access - Part 1: Common Architecture
- Well-Known Text representation of Coordinate Reference Systems

Chapter 3. Terms

3.1. Terms and definitions

For the purposes of this report, the definitions specified in Sub-clause 5 of OGC API-Common Part 1 OGC 19-072 [http://docs.opengeospatial.org/DRAFTS/19-072.html] 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)

- **dataset**

  collection of data, published or curated by a single agent, and available for access or download in one or more formats

- **feature**

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

- **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)

- **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]

- **Web API**

  API using an architectural style that is founded on the technologies of the Web [derived from the W3C Data on the Web Best Practices]

- **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 19119:2016)

- **request**

  invocation of an operation by a client

- **response**

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

3.2. Abbreviated terms

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

- API Application Programming Interface
- CRS Coordinate Reference System
- HTML Hypertext Markup Language
- HTTP Hypertext Transfer Protocol
- JSON JavaScript Object Notation
- CSW Catalogue Service for the Web
- WCS Web Coverage Service
- WFS Web Feature Service
- OWS OGC Web Services
- REST Representational State Transfer
- ETS Executable Test Suite

Chapter 4. Overview

Section 5 introduces the Environmental Data Retrieval Sprint by describing the background, the EDR API approach, use cases, data resources, implementations, and the demonstrations. The section also presents a list of the organizations represented by the participants.

Section 6 describes the working methods of the Sprint, and each of the specific objectives that were involved in the Sprint.

Section 7 presents the issues arising and resolved, as well as the issues outstanding.

Section 8 provides the recommended topics for further discussion and the next steps for the EDR API specification.

Chapter 5. Introduction

A Hackathon/Sprint was organized under the auspices of OGC to progress the development of the Environmental Data Retrieval API (EDR-API). It was to be hosted by the US National Weather Service near Washington DC, from 18-20 March 2020. Unfortunately, because of the global health crisis, the physical face-to-face meeting was canceled. It was decided to hold a virtual Hackathon/Sprint, more or less keeping to US Washington time (Eastern Daylight savings Time) and to communicate using GoToMeeting teleconferencing facilities of OGC, email, and GitHub.

The EDR-API is intended to allow a small amount of data, as required, to be retrieved, or "sampled", by coordinates from data stores that are too large or impractical to copy and transmit. The data queries are based on common but distinct geometry types (i.e., points/time series, polygons, trajectories, etc.).

The API is being developed by the OGC EDR API Standard Working Group against the following documents.

- Charter [https://github.com/opengeospatial/Environmental-Data-Retrieval-API/blob/master/EnvironmentalDataRetrievalAPI-SWG-Charter.adoc]
- Candidate standard [https://github.com/opengeospatial/Environmental-Data-Retrieval-API/blob/master/candidate-standard/EDR-candidate-specification.adoc]
- Background reading material [https://github.com/opengeospatial/Environmental-Data-Retrieval-API]

The first draft of the candidate standard was based on the OGC API - Features, Part 1: Core and what is now OGC API - Common, Part 1: Core following a hackathon in London during June 2019. Both of these standards follow OGC current policy on API development and are consistent with OpenAPI Version 3.

5.1. Background

Environmental data typically involves huge data stores which are difficult to duplicate and transfer, with the difficulty compounded by rapid information replacement, such as in weather forecasts. Such data stores are often generated by a multitude of data resources, with a large number of interested users from multiple domains.

Much of the data exist in operational environments, where service reliability, availability, and scalability, both in terms of volume and numbers of users, are essential requirements.

Many of the existing services have powerful query capabilities, however some are difficult to scale in volume and numbers of users, making it hard to guarantee service levels. They may be difficult to secure.

Because some of the existing services have relatively complex APIs, there is a steep learning curve and they may be difficult to integrate into production environments.

5.2. EDR API approach

By using simple query patterns “sampling” into a large data store where the data publisher is responsible for transforming and supplying the data, the user only defines area of interest in time and space, using Well Known Text (WKT) making the data simple to consume, handle and combine both “feature” and “coverage” data sources.

All the logic is in the “collection” itself (self-describing) and the self-describing queries.

Previous experiments, prototypes, experiences, and production implementations give confidence that this approach avoids large scale duplication of data, is easier to implement, with easier access control supplied by modern API management tools, and gives interoperability through simplicity.

The EDR API is intended to be part of a proposed family of OGC APIs that are complementary to existing OGC Web-based geospatial services. These new APIs are intended for a different audience, Web developers rather than geospatial specialists, and serve as basic introduction to OGC services thus lowering the bar for entry to using OGC Web services.

5.3. Purpose

The EDR API Sprint was to provide feedback based on the current EDR API candidate specification, and assess the compatibility with the OGC API - Features - Part 1: Core standard and the in-progress OGC API - Common - Part 1: Core standard.

5.4. Sprint Goals

These were developed in meetings of the EDR API Standard WG and the Met Ocean Domain WG, with other input from OGC staff and Members:

- Build implementations of the EDR API service based on existing data stores, both client and server-sides;
- Develop client-side value-added applications that consume data from prototype implementations of EDR API; and
- Develop prototype functionality for EDR API through developing and running code.

The goals were refined by the working groups into more specific objectives and deliverables.

5.4.1. Objectives

- Verify and validate requirements and methods for the Query and Filter operations of the EDR API
- Prototype rapidly other geometry types for the EDR API (partitioned grids or tiles, vertical profiles and/or trajectories/corridors)
- Develop client-side value-added applications that consume data from UK Met Office and US NWS prototype EDR API implementations
- Make progress on conformance testing of EDR API parts based on queries

5.4.2. Proposed deliverables

- Software Code
- Recommended changes to the EDR API specification
- Report on level of compatibility with OGC API Features and Common
- Recommendations for other OGC APIs (Features, Common, etc.)
- Report on conformance class testing
- These goals, objectives, and deliverables were all discussed by the attendees prior to the meeting and in the opening session of the Sprint

5.5. Use Cases

The Sprint was introduced by a quick review of the EDR API use cases at https://github.com/opengeospatial/Environmental-Data-Retrieval-API/tree/master/use-cases. These included:

- Get Parameters for a Point across a Time series;
- Obtain or view a forecast time series of a parameter at a point;
- Get Unstructured Observations from within a Polygon;
- Show Weather Radar data time series; and
- Obtain or view Air Traffic Hazards and Restrictions for an Area.

5.6. Existing Data, Implementations and Demos

5.6.1. Implementations of EDR API

- US NWS: https://data-api.mdl.nws.noaa.gov/EDR-API Available data offered (METARs, TAFs, Global Forecast System(GFS), North American Mesoscale model (NAM), National Digital Forecast Database (NDFD))
- UK Met Office: http://labs.metoffice.gov.uk/edr/ Available data offered (METARs, UK Global Model, US GFS, US NDFD, Open StreetMap, Digital Elevation Models)

5.6.2. Data sources

- Amazon data sources: https://registry.opendata.aws/?search=tags:gis,earth%20observation,mapping,meteorological,environmental,transportation

5.6.3. Demonstrations

(1) Automated aggregation of metadata from collections (3D, 4D, 5D)

This creates collections of parameters (that have common dimensions) from operational data stores, and outputs in JSON which is used to convert the original dataset to zarr. Each zarr data store represents a specific collection. With the parameters grouped by common dimensions, more complex queries than EDR API can be made. Demo at https://github.com/ShaneMill1/edr-

automation.

(2) Demonstration of client side APIs (single and multi-domain feature extractions)

Uses the EDR API to access time series at a point for:

- Observations from a point cloud – latest airfield observations (METARs);
- Gridded forecast current data from US NWS GFS using https://data-api.mdl.nws.noaa.gov/EDR-CLIENT-API; and
- Gridded forecast data from 2 days old UK Met Office Global Unified Model using http://labs.metoffice.gov.uk/edr/ . Demo at http://labs.metoffice.gov.uk/map/wotwdemo/.

5.7. Participants

There were about two dozen attendees, from North America, the far East, and Europe. They represented OGC, government departments, universities, and private companies. Table 1 lists members that participated in the Sprint.

Table 1. Participating members

Name	Organization
Steve Olson	US NWS
Shane Mill	US NWS
Paul Hershberg	US NWS
Dave Blodgett	USGS
Jim Kreft	USGS
Chris Little	UK Met Office
Mark Burgoyne	UK Met Office
Pete Trevelyan	UK Met Office
Peng Yue	Wuhan University
Boyi ShangGuan	Wuhan University
Lei Hu	Wuhan University
Zhang Mingda	Wuhan University
Jeff Donze	ESRI
Sudhir Shrestha	ESRI
Keith Ryden	ESRI
George Percivall	OGC
Josh Liebermann	OGC
Chuck Heazel	HeazelTech LLC
Clemens Portele	Interactive Instruments

Name	Organization
Tom Kralidis	Meteorological Service of Canada
Stephan Siemen	ECMWF

Chapter 6. Working Methods

A GitHub repository was established at https://github.com/opengeospatial/EDR-API-Sprint . A branch of the EDR API Candidate standard was also created at https://opengeospatial.github.io/EDR-API-Sprint/edr-api.html so that changes could readily be made if required. Attendees also raised Issues in the Sprint repository to indicate their objectives, and any issues encountered. Discussions were then followed on the Issues tabs and tagged according to their content. Background information was supplied in a Slide presentation https://docs.google.com/presentation/d/1vU8O7dnkima9Vch_T0ebECV5RRjSw-8Kuox15gN2Z4k/edit?usp=sharing and wiki pages at https://github.com/opengeospatial/EDR-API-Sprint/wiki.

Also, at the beginning and end of each working day, briefing sessions were held on GoToMeeting to present work done, and to discuss in more depth any issues and their resolutions. Sessions were chosen to enable the different time zones to take part.

6.1. Specific Objectives

Most attendees, either individuals or teams, identified specific objectives that they would like to achieve.

6.1.1. Develop new EDR API server and client, Mark Burgoyne (Issue #20)

Develop a simple EDR server to demonstrate Point, Polygon, and Items queries against the same Collection of data. This is still work in progress, to track and implement the changes to the latest EDI API specification, and also to try out novel ideas. It will also demonstrate an alternative approach to the proposed application item_id as a unique identifier for a location (i.e., a METAR id, GeoHash key, WhatThreeWords, etc.) which has a set of data assigned to it. This addressed Issues #04 and #05: Point, timeseries at a point, vertical profile at a point, and 2D polygon.

6.1.2. Demonstrate automatic harvesting of metadata from aggregations of data stores to improve search capabilities for use with the EDR API, US NWS (Issue #19 and Issue #14))

Extend the automated aggregation of metadata from Collections (3D, 4D, 5D) by storing the output from a forecast model run as a concatenation of binary GRIB files, then process as xarray dataset with pynio and extract metadata (parameter ID determined by pynio, long name, level type, dimensions) and put in a pandas data frame. The data frame creates a map between dimension names and dimension values. Parameters that have the same dimensions are grouped. The metadata is output in JSON. The original dataset is converted to zarr using the collection JSON. This approach will enable more complex queries than just EDR API. The implementation is: https://data-api.mdl.nws.noaa.gov/EDR-API/groups/US_Models?outputFormat=application%2Fjson based on NAM and GFS GRIB data. The aggregation will be extended to other national weather service data sources, or other file types such as HDF5, NetCDF:

- France on AWS at https://registry.opendata.aws/meteo-france-models/
- Canada at https://weather.gc.ca/grib/index_e.html

- The Netherlands at https://data.knmi.nl/datasets?q=grib

Currently, one can search for a keyword, and Collections, which have parameters with a long name containing that keyword, will be shown as a URI, which will link to the EDR API query endpoint. The links returned need to be ordered in a dictionary. Additional work would be to:

- Match the keywords with other metadata attributes such as the dimension names (i.e., ISBL for isobaric);
- Add the ability to search by parameter name as well as dimension name so that a user can search more narrowly; and
- Utilize OpenSearch geo and time extensions. A pycsw module offers a python implementation of OGC CSW as well as the OpenSearch geo and time extensions. http://docs.pycsw.org/en/stable/introduction.html.

Looking at the documentation, pycsw was incorporated into the EDR-API implementation following this approach: https://docs.pycsw.org/en/latest/api.html

Then, following the steps provided at the link below, a compliant blank SQLite database was created to start from: http://docs.pycsw.org/en/stable/administration.html#metadata-repository-setup

Finally, the beginning of the implementation can be seen at the following endpoint: https://data-api.mdl.nws.noaa.gov/EDR-API/csw?service=CSW&version=2.0.2&request=GetCapabilities

The next steps will be to connect the dots from how metadata is created in the aggregation of collections software and incorporate that metadata into these services. The discussion needs to continue on what/how the formal metadata would be to serve as part of a CSW instance (Dublin Core, ISO, etc.). This will also give us an opportunity to investigate the OGC API - Records work (disclosure: Tom Kralidis is on this SWG and working on an implementation). This is now an issue in the EDR SWG GitHub at https://github.com/opengeospatial/Environmental-Data-Retrieval-API#40.

Outcomes

- Successfully used aggregation software to ingest GFS 1 Degree, 1/2 Degree, 1/4 Degree, NAM 32km, 12km, and Météo-France's Arpège 1/2 Degree data
- Identified memory problem when creating zarr data stores for higher resolution data

Observations

- Data should be WGS84 for this implementation, therefore NAM data would need to be converted to this CRS

Future work

- Test other national weather services' GRIB data
- Optimize the code to solve memory issue

6.1.3. Implement Trajectory queries against typhoon/hurricane data. Wuhan University (Issue #03)

This is the same as Issue #09, Objective: Trajectory, 2D, 3D, or 4D. The trajectory query was successfully implemented in all the 2D, 3D, and 4D cases, using the EDR API specification proposal to use Well Known Text (WKT) LineString formats.

6.1.4. Explore putting EDR API on ESRI REST API image server, UK Met Office, ESRI (Issue #02 and Issue #17)

This is ongoing work. There do not seem to be any fundamental architectural problems with possible approaches for the proposed use cases. It is expected that the queries supported would be for points/time series/vertical profiles, and cube/tiles.

The main aim of this work is to show how the EDR "Pattern" maybe "mapped" onto a proprietary RESTful API, such as ESRI's Arc Image Server. The output from this work will be a document outlining the issues that are brought to light and a very simple prototype consisting of a simple proxy server that uses the EDR API and translates into the Image Server REST API.

6.1.5. Use EDR API to retrieve feature orientated observations from hydrological network data stores, USGS (Issue #01)

6.1.6. Conformance testing, ECMWF (Issue #16)

There was general support for this topic, in particular, for the returned content of the payload from a specific query, but no work was done.

OGC API Features has a conformance test suite, written to be modular, and presumably, some of this will be shared with an OGC API Common test suite.

Conformance tests for the returned payload is a bit harder.

- XML has validating XML schemas and Schematron rules.
- GeoJSON has a clear standard to test against. For the payload, JSON schema is nowhere near as rigorous as XML schema.
- CoverageJSON is well defined in the original work done by Jon Blower and Maik Reichardt at Reading University, and the process to agree an OGC standard for the structure of CoverageJSON has started. See the repo at https://github.com/opengeospatial/CoverageJSON.

6.1.7. Implement and demonstrate a STAC (Spatio-Temporal Asset Catalogue) for accessing meteorological real-time data stores in pygeoapi, Meteorological Service of Canada (Issue #21)

This is ongoing work that started after the formal Sprint days. STAC provides a common language to describe a range of geospatial information, so it can more easily be indexed and discovered. A "spatiotemporal asset" is any file that represents information about the earth captured in a certain space and time. While STAC work has been primarily focused on space-based earth observation imagery, there is value in investigating STAC capability for real-time Numerical Weather Prediction

(NWP) data offerings. This would allow for an arbitrary hierarchy based on NWP workflow (model runs, forecast hours, forecast variables, etc.) as well as simple, file-based discovery/traversal of the same.

Questions

- How would NWP look as a static catalogue of (in this case) GRIB2 files?
- How would a given STAC implementation for MetOcean compare to OGC API - Records?
- Would a MetOcean STAC profile be valuable? Results?
- Implemented a STAC catalog for a (very very small subset) of our Global Deterministic Prediction System (GDPS)
- Code: https://github.com/geopython/pygeoapi/pull/389
- Deployment:
  - STAC root: http://52.170.144.218:8000/stac
  - root catalog: http://52.170.144.218:8000/stac/nwp
  - model run: http://52.170.144.218:8000/stac/nwp/00
  - model run/forecast hour: http://52.170.144.218:8000/stac/nwp/00/000
  - data description: http://52.170.144.218:8000/stac/nwp/00/000/CMC_glb_DEPR_ISBL_750_latlon.15x.15_2020040100_P000
  - raw asset download: - data description: http://52.170.144.218:8000/stac/nwp/00/000/CMC_glb_DEPR_ISBL_750_latlon.15x.15_2020040100_P000.grib2

Observations

- Successfully tested with STAC-validator
- Data properties are basically GRIB2 metadata
- Links are initially minimal: we could have links back to related EDR workflows (or OGC API - Coverages, OGC API - Processes)
- Search is not in scope for STAC Catalog (more OGC API - Records and STAC Catalog) Future Work
- Investigate integration with OGC API - Records and STAC Catalog
  - GDPS is a collection level discovery metadata in the scope of OGC API - Records
  - searching within GDPS would be a link from the OGC API - Records document/search result to the STAC API (essentially searching model runs/forecast hours of data)

6.1.8. The remaining Issues #04 - #10 were generic objectives covering the full scope of data query patterns of the EDR API, but divided up according to expected difficulty

- Point, timeseries at a point, and vertical profile at a point
- Polygon and tile (2D)
- Polygon in 3D or 4D

- Polygons in 3D and 4D
- Tile/Cube in 3D or 4D
- Trajectory, 2D, 3D or 4D
- Corridor, 3D or 4D

Chapter 7. Issues

7.1. Issues arising and resolved

7.1.1. Issue #03 and #09 Trajectories: which time specification?

The EDR API specification uses ISO8601 style of notation to specify dates and times for query parameters. This is very convenient and understandable for users. However, the Trajectory query specifies time as a number of seconds since the UNIX epoch. This is specified in the Well Known Text (WKT) standard for defining lines/trajectories by using LineString. This makes time into a proper coordinate, enabling easier software calculations, though the units are not easily comprehended by users. It was agreed to keep the two different approaches, as both had merits and disadvantages, and elicit wider feed back from the OGC Members, OAB, and the public. A mitigation may be to incorporate an easy to use service to convert between the two different time representations.

7.1.2. Issue #04 Point, time series at a point, and vertical profile at a point: which combinations?

Would the current EDR API specification allow simultaneously both a time series and a vertical profile at a point? I.e., A 2D array of values would be returned. At present, the Point query only allows a time series at a point or a vertical profile at a point, but not both. It was agreed to stay with these minimal cases. The general 2D vertical time series is not widely used outside of meteorology, where it is known as a Hovmöller diagram.

Another consequence of these discussions was that Point was re-named Position, and Polygon re-named as Area.

7.1.3. Issues #08 and #06 and #05 Cube/Tile, Polygon in 3D or 4D, Polygon/Tile in 2D: which bounding box styles?

Should bounding boxes for a query, in 4D, be specified by ranges of values or coordinates of the corners? The consensus was that ranges are more natural for time and vertical coordinates. This is then the same as specified 2D Polygons in WKT.

There was no disagreement that there will be no polyhedra or polytopes (complex multi-dimensional polygons, rather than just “extruded” 2D polygons. An extruded 2D polygon can be called a “prism.”

7.1.4. Issue #0? What are the EDR API resource types?

This was the usual semantic question. It was agreed that the initial resource was a persistent, dense data store. The queries against it were Discrete Sampling Geometries sampling the data store, and were transient, but could be made a persistent resource if required by another service. [The Research Data Alliance, RDA, recommends a query store in its Best Practice Recommendations for citing dynamic data.] the returned data payload is also a transient resource, which also could be made persistent. It was agreed to add words to this effect in the EDR API candidate standard.

7.1.5. Issue #18 Streaming of EDR API response media types

If the data returned in response to a query is small, there is no need for streaming of the results. It may be necessary for large responses, but that is starting to be outside the scope of the EDR API. It was agreed that the EDR API specification would not mention streaming, and it would be an implementation decision as to whether streaming is supported. This decision would obviously be influenced by the choice of supported media type, which may or may not support streaming.

7.1.6. Issue #12 Items view of EDR resources and Issue #15 JSON-Schema for /collections/{collectionID}/items

The EDR API provides no mechanism for the user to discover available location identifiers (such as ICAO ids) in the metadata. The identifiers are available in the query results, but not in any of the available metadata outputs. What would the JSON-Schema be for these items? E.g: collections/metar/raw/items?id=KIAD&parametername=icao_id&time=2020-01-31T00:00:00Z/2020-02-01T04:00:00Z. If consistent with Features, then collections/metar/raw/items returns the list of items (paged, if there are many) and that the query would look like: collections/metar/raw/items/KIAD?parametername=icao_id&datetime=2020-01-31T00:00:00Z/2020-02-01T04:00:00Z.

Currently, the EDR API “Items” specifies CoverageJSON rather than a GeoJSON Feature Collection with valid query parameters for each item in the collection. Can lists of available parameters in GeoJSON be exposed?

```
{
  "type": "FeatureCollection",
  "crs": {
    "type": "EPSG",
    "properties": {
      "code": 4326,
      "coordinate_order": [ 1, 0 ]
    }
  },
  "features": [
    {
      "type": "Feature",;
      "id": "123",
      "geometry": {
        "type": "Point",
       e. "coordinates": [ -105.683442, 36.740017 ]
      },
      "properties": {
        "datetime": "2018-02-12T00:00:00Z/2018-03-18T12:31:12Z",
        "parametername": [ "param object 1", "param object 2" ],
        "label": "Something like a site name to use on a link",
        "uri": "https://feature_identifier"
      }
    }
  ]
}
```

This validates against the GeoJSON schema: id is what goes in /collection/{collectionID}/items/{itemID}; datetime follows features; parametername follows the draft spec naming convention; label needed because there has to be a list/link label in the client; uri which could be @id, like JSON-LD and it should really be a linked data feature ID which 303 redirects if a real-world sampling feature. If these extensions are put in the GeoJSON FeatureCollection schema, the software SF/GDAL outputs:

```
> sf::read_sf("~/Documents/active_code/EDR-API-Sprint/items/test.geojson")
Simple feature collection with 1 feature and 5 fields
geometry type:  POINT
dimension:      XY
bbox:           xmin: -105.6834 ymin: 36.74002 xmax: -105.6834 ymax: 36.74002
epsg (SRID):    4326
proj4string:    +proj=longlat +datum=WGS84 +no_defs

A tibble: 1 x 6
  id    datetime            label                uri           parametername
geometry
  <chr> <dttm>              <chr>                <chr>         <list>
<POINT [°]>
1 123   2018-02-11 18:00:00 Something like a si… https://feat… <chr [2]>        (-
105.6834 36.74002)
```

The properties schema is here:

```
 properties:
      type: object
      title: The Properties Schema
      description: An explanation about the purpose of this instance.
      default: {}
      example:
      -   datetime: 2018-02-12T00:00:00Z/2018-03-18T12:31:12Z
          label: Something like a site name to use on a link
          parametername:
          - param object 1
          - param object 2
          uri: https://feature_identifier
      required:
        - datetime
        - parametername
        - label
        - uri
      properties:
        datetime:
            type: string
            title: The Datetime Schema
            description: An explanation about the purpose of this instance.
            default: ''
            example:
            - 2018-02-12T00:00:00Z/2018-03-18T12:31:12Z
        parametername:
            type: array
            title: The Parametername Schema
            description: An explanation about the purpose of this instance.
            default: []
            items:
```

```
                type: string
                title: The Items Schema
                description: An explanation about the purpose of this instance.
                default: ''
                example:
                - param object 1
                - param object 2
        label:
            type: string
            title: The Label Schema
            description: An explanation about the purpose of this instance.
            default: ''
            example:
            - Something like a site name to use on a link
        uri:
            type: string
            title: The Uri Schema
            description: An explanation about the purpose of this instance.
            default: ''
            example:
            - https://feature_identifier
```

7.2. Issues outstanding

7.2.1. Issue #11 Groups versus Collections

The EDR API specification found a need to have Groups of Collections in the API path. In the wider OGC, there is now a discussion of whether APIs could have Collection of Collections. The Sprint agreed to stay with Groups until the wider issue is resolved.

7.2.2. Issue #10 Corridors, 3D or 4D

Corridors were originally envisaged as a volume defined by a surface of constant distance from a line trajectory. The idea of the bottom of a corridor volume being delineated by the earth's surface (or some other surface) was raised. It was agreed to tackle this interesting, practical, and difficult problem later.

7.2.3. Other Issues

Observers, from outside the Sprint, have raised some substantive questions, including about interpretation and implementation of vertical coordinates in the EDR API. These will be raised and addressed in the EDR API Standard WG.

Chapter 8. Recommendations and Next Steps

- WKT LineString format to be used for multi-dimensional geometries
- Some query names were changed (Polygon → Area, Point → Position)
- The specification will not mention streaming, because it is an implementation detail
- Agreed that we would allow named "locations" (now Positions)
- Agreed that named “times” seem to be a good analogy to named “locations,” such as “Latest”
- Groups vs Collections were explored and the current working definition of Collections remains

8.1. Recommended Topics for further discussion

- Contribute EDR API definition of a Collection to the wider OGC debate
- Decide what Metadata frameworks should be used, recommended or mandated and expand metadata offerings
- Determine workflows and search metadata to integrate with OGC API - Records
- Security considerations. This needs to be coordinated with other OGC API groups and the Security Domain Working Group
- Test other sampling geometry types.
- Continue to investigate how to integrate with other APIs (features/coordinates, maps)
- Incorporate a pub/sub mechanism with a later version of the EDR API specification
- Investigate how to align with decision impact groups like SmartCity, others

8.2. Next Steps for EDR API Specification

- Generate Engineering Report on Sprint
- Finalize updates to current specification
- Update documentation
- Start wider consultation processes inside and outside OGC

Appendix A: Bibliography

1. W3C/OGC: Spatial Data on the Web Best Practices, W3C Working Group Note 28 September 2017, https://www.w3.org/TR/sdw-bp/
2. W3C: Data on the Web Best Practices, W3C Recommendation 31 January 2017, https://www.w3.org/TR/dwbp/
3. W3C: Data Catalog Vocabulary, W3C Recommendation 16 January 2014, https://www.w3.org/TR/vocab-dcat/
4. IANA: Link Relation Types, https://www.iana.org/assignments/link-relations/link-relations.xml