OGC EDR API Overview 2021-02-09

Status, implementations, key issues, plans, demo

Chris Little, Technology Fellow, Met Office
Chair OGC EDR API Standard WG
Chair OGC Meteorology & Oceanography Domain WG
Member OGC Architecture Board, W3C Spatial Data on the Web WG


MADE IN
SWITZERLAND


MARKS & SPENCER LTD
YOUR FAVOURITE
FAMILY
BISCUIT
SELECTION
MARKS & SPENCER LTD
ESTABLISHED 1884
1d
ORIGINATORS OF THE PENNY BAZAAR
MARKS & SPENCER LTD
FAMILY
BISCUIT


BOURBON
© www.retro-en-design.com

Convenient Access

- Meteo interfaces & formats are not cross-domain

- Geospatial domain has some generic interfaces:
  - WFS Web Feature Service – get things to put on a map
  - WCS Web Coverage Service – get ‘fields’ to display on a map
    - tendency to equate a coverage with an image, of pixels
  - WMS Web Map Service – get a map or layers for a map
    - WMS has ‘broken out’ of geospatial domain

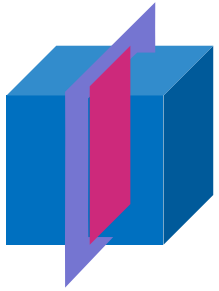
Maps & Layers ‘broken’

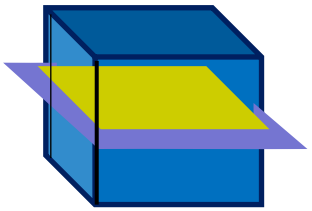
- 2D Maps and Layers long established
- 3D or 4D data breaks ‘Layers’ paradigm
  - Suppose data have 10 levels x 10 times: =>
  - 100 layers – choose from a menu?
- But
  - 100 levels x 100 times x 100 parameters x 100 ensembles: =>
  - 100 000 000 layers?
- Use WCS? Does not scale to lots of users
- Use Environmental Data Retrieval (EDR) API

WCS Coverage patterns

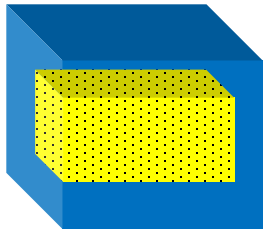
- GRID Trimming

- GRID Layers (Slicing)





- GRID Re-sizing and sampling



- GRID Time series (Slicing again)

Why a new API?

- Existing OGC Standards focus on powerful query capabilities for experts
- Generally a mapping between standards and specific data structures
- EDR aims to simplify blending data from a variety of sources / domains
- EDR is built around a set of separate but related query patterns
- ‘Measured’ values / variables / parameters considered attributes of location & time
- Structure of the returned data is based on the query type
- API publisher only has to support the queries that they need
- Each query type has well-defined, limited, functionality
- This should make it easier to optimize and secure server back-ends
- Defined using only HTTP(S)
- Which should reduce the barriers to implementation

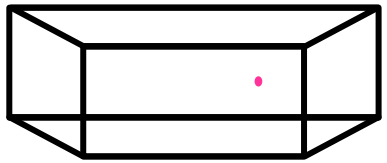
What does an EDR query look like?

https://somedomain.or.other/collections/{collectionId}/{queryType}?.....

- Position & Location
- Area & Radius & Cube
- Trajectory & Corridor
- Item
- …/{queryType}?keyword=value&keyword=value&…..

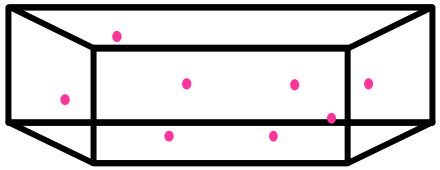
EDR API Queries:- position (and time)

- **Single Point:**



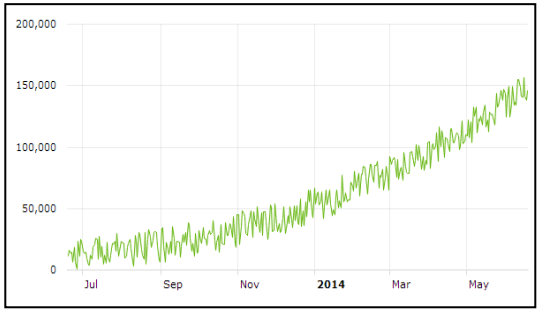
- coords=POINT(0 51.48)
- datetime=2018-02-12T23%3A20%3A52Z

- **Collection of Points:**



- coords=MULTIPOINT((38.9 -77),(48.85 2.35),(39.92 116.38),(-35.29 149.1),(51.5 -0.1))
- datetime=2018-02-12T23%3A20%3A52Z

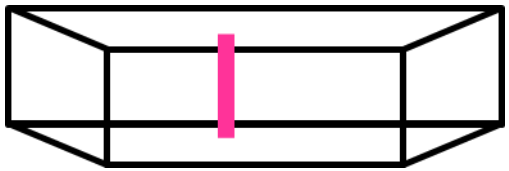
- **Time-series at a Point:**



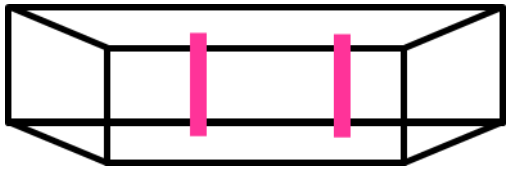
- **Time Series of a collection of Points:**

- datetime=2018-02-12T00%3A00%3A00Z%2F2018-03-18T12%3A31%3A12Z [%3A = :]

EDR API Queries: position (and z)



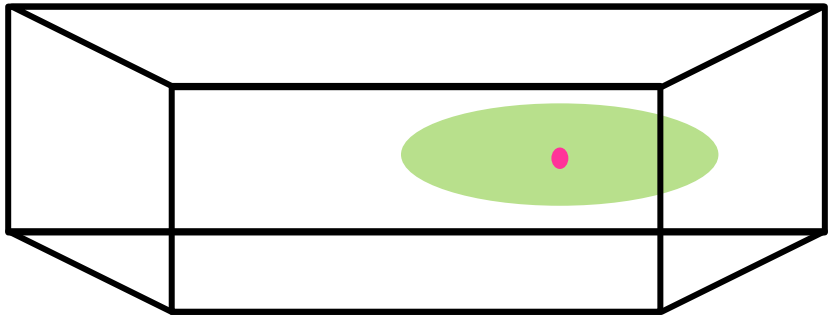
- **Single Profile**
  - coords=POINT(0 51.48)
  - z=2/100



- **Collection of Profiles**
  - coords=MULTIPOINT((38.9 -77),(48.85 2.35),(39.92 116.38),(-35.29 149.1),(51.5 -0.1))
  - z=2/100

- **Time-series of a Profile**
- **Time Series of a collection of Profiles**
  - datetime=2018-02-12T00%3A00%3A00Z%2F2018-03-18T12%3A31%3A12Z [%3A = :]

EDR API Queries: radius (& time or z)



- radius

/**radius**?
coords=POINT(-2.923 52.059)
&**width**=10
&**width-units** =km
&**parameter-name**=Visibility,Air Temperature
&***datetime***=2020-12-09T04:00Z/2020-12-09T16:00Z
&**crs**=CRS84
&**f**=CoverageJSON

EDR API Queries: area (& time or z)

area

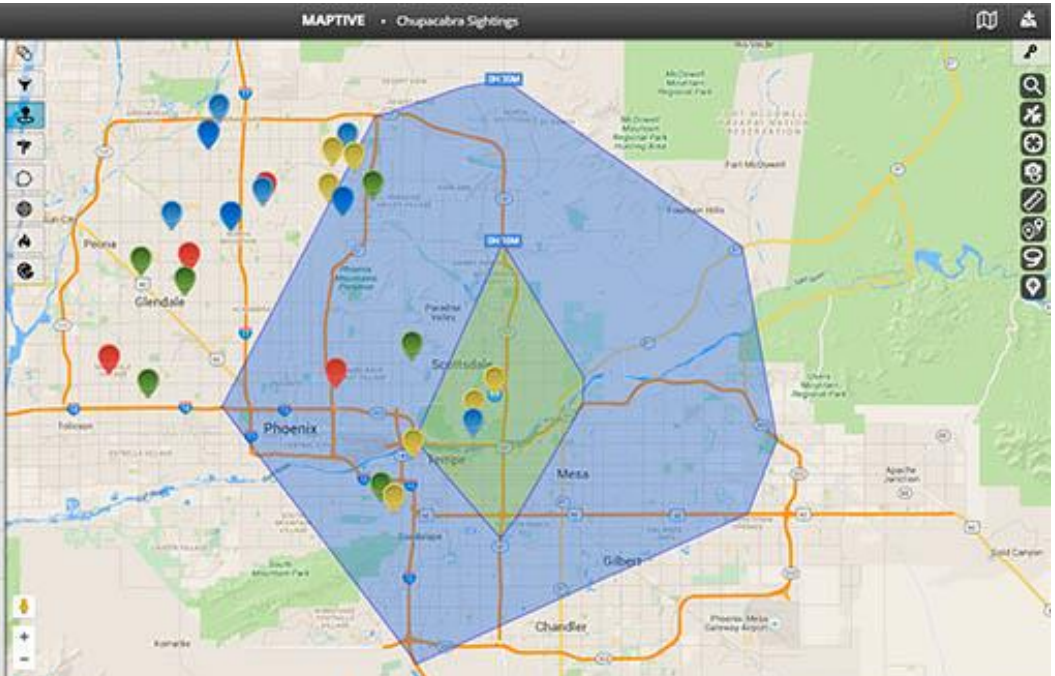
coords=POLYGON((-15 48.8,-15 60.95,5 60.85,5 48.8,-15 48.8))

coords=MULTIPOLYGON((-15 48.8,-15 60.95,5 60.85,5 48.8,-15 48.8),
(-6.1 50.3,-4.35 51.4,-2.6 51.6,-2.8 50.6,-5.3 49.9,-6.1,50.3))

datetime=2018-02-12T23%3A20%3A52Z

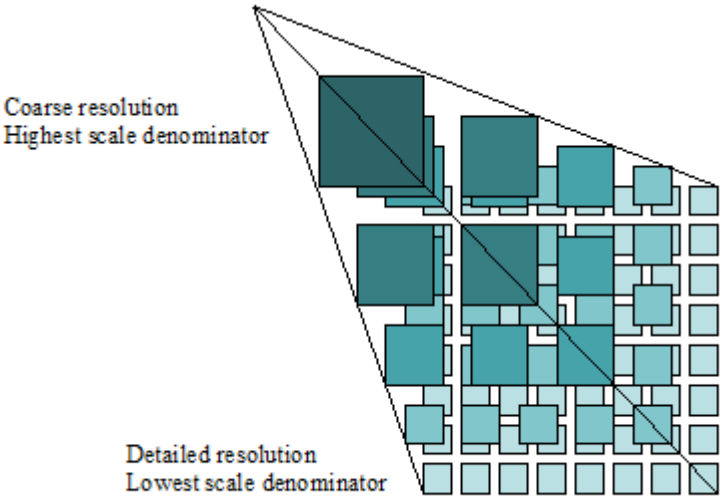
datetime=2018-02-12T00%3A00%3A00Z%2F2018-03-18T12%3A31%3A12Z

Z=50 z=2/100 z=ALL

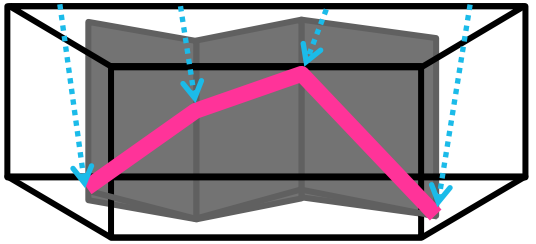


tile

coords=POLYGON((-6.0 50.0,-4.35 50.0,-4.35 52.0,,-6.0 52.0,-6.0 50.0))



EDR API Queries: trajectory



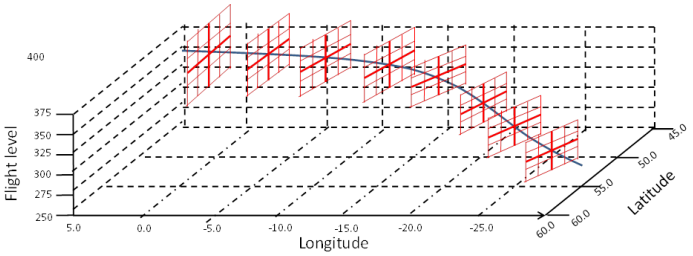
• Trajectory

- 2D: coords=LINESTRING(51.14 -2.98, 51.36 -2.87, 51.03 -3.15, 50.74 -3.48, 50.9 -3.36)
- coords=LINESTRING(51.14 -2.98, 51.36 -2.87, 51.03 -3.15, 50.74 -3.48, 50.9 -3.36) &datetime=2018-02-12T23:00:00Z
- coords=LINESTRING(51.14 -2.98, 51.36 -2.87, 51.03 -3.15, 50.74 -3.48, 50.9 -3.36) &z=850
- coords=LINESTRING(51.14 -2.98, 51.36 -2.87, 51.03 -3.15, 50.74 -3.48, 50.9 -3.36) &datetime=2018-02-12T23:00:00Z &z=850

- **3D:** coords=LINESTRINGZ(51.14 -2.98 0.1, 51.36 -2.87 0.2, 51.03 -3.15 0.3, 50.74 -3.48 0.4, 50.9 -3.36 0.5)
- coords=LINESTRINGM(51.14 -2.98 1560507, 51.36 -2.87 15605076, 51.03 -3.15 15605082, 50.74 -3.48 15605085, 50.9 -3.36 1560510240)
- coords=LINESTRINGM(51.14 -2.98 15605, 51.36 -2.87 1560507, 51.03 -3.15 1560508, 50.74 -3.48 1560508, 50.9 -3.36 15605102) &z=200
- coords=LINESTRINGZ(51.14 -2.98 0.1, 51.36 -2.87 0.2, 51.03 -3.15 0.3, 50.74 -3.48 0.4, 50.9 -3.36 0.5) &datetime=2018-02-12T23:00:00Z

- **4D:** coords=LINESTRINGZM(51.14 -2.98 0.1 1560507000,51.36 -2.87 0.2 1560507600, 51.03 -3.15 0.3 1560508200, 50.74 -3.48 0.4 1560508500, 50.9 -3.36 0.5 1560510240)

• Corridor (‘fat trajectory’)



- **Ditto, as above plus** corridor-width = width "/" units
- corridor-z = height "/" units

EDR API Queries: shared parameters

- **coords** – spatial coordinates defined as Well Known Text (WKT), including time
- **time** – time range based on the ISO8601 standards
- **parameter-name** – comma delimited list of the parameters
- **f** – data format used to return the data
- **crs** – Coordinate reference system used to return the data

- Others parameters specific to each query type

EDR API: Other Queries

- Locations
  - Named location identifiers for predefined geospatial coordinates
  - Lists named locations and the definition of coordinates they represent
  - Supports the other common query parameters ***parametername***, ***time***, ***f*** and ***crs***
- Items
  - Lists available pre-existing, pre-defined objects by their identifier
  - The list of available items can be sub-set by area and time
  - Allows user to request required objects by their identifier
  - OGC API – Features Part 1: Core compatible

Environmental Data Retrieval API

- Early Proof of Concept Hackathon Washington 2018-12,
  - based on long standing meteorological production systems
- WeatherOnTheWeb API renamed EDR API – hydrology, defence, space,…
  - Github https://github.com/opengeospatial/Environmental-Data-Retrieval-API
  - Mailing list EDR-API.SWG@lists.opengeospatial.org
- Other hackathons: London Geovation 2019-06 https://www.opengeospatial.org/projects/initiatives/oapihackathon19
- Charter established 2019-11-25
- 1st EDR API Hackathon 2020-03-18/20
- OGC Architecture Board Review passed 2020-07-28
- Released for Public Comment 2020-08-28
- Public Comments finished 2020-09-28
- 2nd EDR API Hackathon 2020-11-09/10
- Request publication 2020-12-07/11 OGC E-Vote ongoing

Reuses common practice:

- WKT for geometry (2D, 3D, or 4D)
- ISO 8601/RFC 3339 for other time ranges
- Compliant to API Common Core & Collections
- Compatible with API Features Core

Public comments:

- Why recommend CoverageJSON, not GeoJSON, CIS JSON, NetCDF, GeoTIFF, etc?
  - CoverageJSON has significant use in communities, supports features and coverages, and is performant
  - NOT mandated - other payload formats can be used

EDR API SWG Charter Members

Name	Organization
Chris Little	UK Met Office
Steve Olson	US National Weather Service
Frédéric Guillaud	Météo-France
Dave Blodgett	US Geological Survey
Tom Kralidis	Meteorological Service of Canada
Roope Tervo	Finnish Meteorological Institute
Bruce Bannerman	Individual
Chris Lynnes	NASA
Ethan Davis	UCAR
Cristiano Lopes	ESA
Iain Burnell	DSTL
Keith Ryden	ESRI

EDR API SWG Plan

Initial Deliverables by Dec 2020: APIs

- retrieve data values at a specified point location altitude and time (x,y,z,t)
- retrieve a time series of values at a specified location and height (x,y,z), whether elevation or altitude with a specific vertical CRS
- retrieve a vertical profile of values at a specified location and time (x,y,t)
- retrieve an array of values across a rectangular area (tile)
- retrieve a set of values across a polygonal area (optional z or t extents)
- retrieve a series of values along a specified trajectory, whether 2, 3, or 4 dimensions
- retrieve a set of values within a 'corridor', a trajectory with a surrounding buffer region along its length

EDR API SWG Next Plans

Currently retrieve data values at a specified position, area, trajectory, corridor.

0. Outreach and implementations
1. Interoperability and consistency with OGC API Records for metadata for datasets/distributions/service endpoints
2. CoverageJSON standardization in WCS SWG or community standard ?
3. Example of EDR compared to WCS and SOS/SWE/O&M
4. Incremental improvements: postponed work, interpolation, plumes, etc
5. 1st order stats and thresholding from collections/instances/ensembles
6. Asynchronous approaches – now tasked to API-Common SWG
7. Security approaches
8. Interoperability and consistency with TestBed16 DAPA and WPS ?

Demo – Burgoyne, Met Office

Questions and answers

Some slides with more background

Weather on the Web (WotW) Hackathon

Washington 2018-12

- Demonstrated initial WotW capability, gridded data focus
- Completed OpenAPI demo endpoint (Canadian CMEP)
  - Successfully conducted test against generic WFS 3.0 client library (OWSLib - https://github.com/geopython/OWSLib)
  - Recommendations to align closer to WFS 3.0 patterns in GitHub: https://github.com/OGCMetOceanDWG/MetOceanAPI-Workshop/blob/master/api-review-comments.md

WotW Hackathon continued

Completed initial cut at WotW API conformance classes:

1. Core: Use simple feature spec (point, line string, polygon)
2. Spatial: Required for start/stop/interval, enumeration, elevation
3. Weather Params: Use WFS3 as basis for query parameters. Create filter
4. Temporal: define time range/units using MetOcean spec, GRIB2, netCDF
5. Measurements: Define specific observations to return
6. Interpolation: Define how to map observations to return locations
    - Specify interpolation method per axis/axes, use well-known methods
    - Specify default as nearest-neighbour.
    - Work on other interpolation methods with vendors
7. Return: Inform JSON, CovJSON, XML, GRIB, netCDF
8. Vendor Extras: Out of scope

WotW Hackathon continued

- Completed initial support of WotW API geometry query types
  - Point, Point Timeseries, Polygon, Profile, Corridor, Trajectory
- Next steps
  - Adoption of MetOcean Profile as OGC standard (Done)
  - Better alignment of MetOcean API with WFS3 patterns (Done)
  - Agree conformance with WFS3 geometry classes (OBE)
  - Assess vendor implementations of MetOcean Profile (OBE)
  - Work towards plugfest (Done)
  - Establish formal Standards WG (Done)

More OGC API hackathons:

London Geovation June 2019
https://www.opengeospatial.org/projects/initiatives/oapihackathon19

- OGC API – Features
- OGC API – Common
- OGC API – WotW / EDR
- OGC API – Maps
- OGC API - Tiles

ESIP, Bethesda, MD, Jan 2020
https://2020esipwintermeeting.sched.com/event/Vabd/esip-and-ogc-api-coverage-analytics-sprint-day-1

- OGC API – Coverages, grids, analytics, …

Summary - Data

Cross-domain information becoming the norm, but

- Domain specific (big) data formats here to stay:
  - (NetCDF, HDF, GRIB, BUFR, FITS, BAM, VCF, GF3, …)
  - Binary, efficient, optimised
  - Established eco-systems of access software & tools
  - Established domain expertise, controlled vocabularies
- Generic data formats have no traction, but
- Web friendly transfer formats for browsers and apps
  - (CSV, JSON, XML)
  - More verbose – small amounts OK

Summary - Metadata

- Metadata should be cross-domain for discovery
  - Metadata fixed format/containers a good start
  - Only for discovery, not usage and management
  - Metadata is open-ended, not in containers/catalogues/portals
- Controlled Vocabularies -> Taxonomies -> Ontologies
- Semantic Web / Web 2.0: standards action in W3C
- Highly scalable
- Highly flexible
- Resolvable registries of controlled vocabularies happening
- Conceptual models can be stored as ontologies
- Ontologies allow valid machine reasoning
- Semantic formats too verbose for data but metadata OK