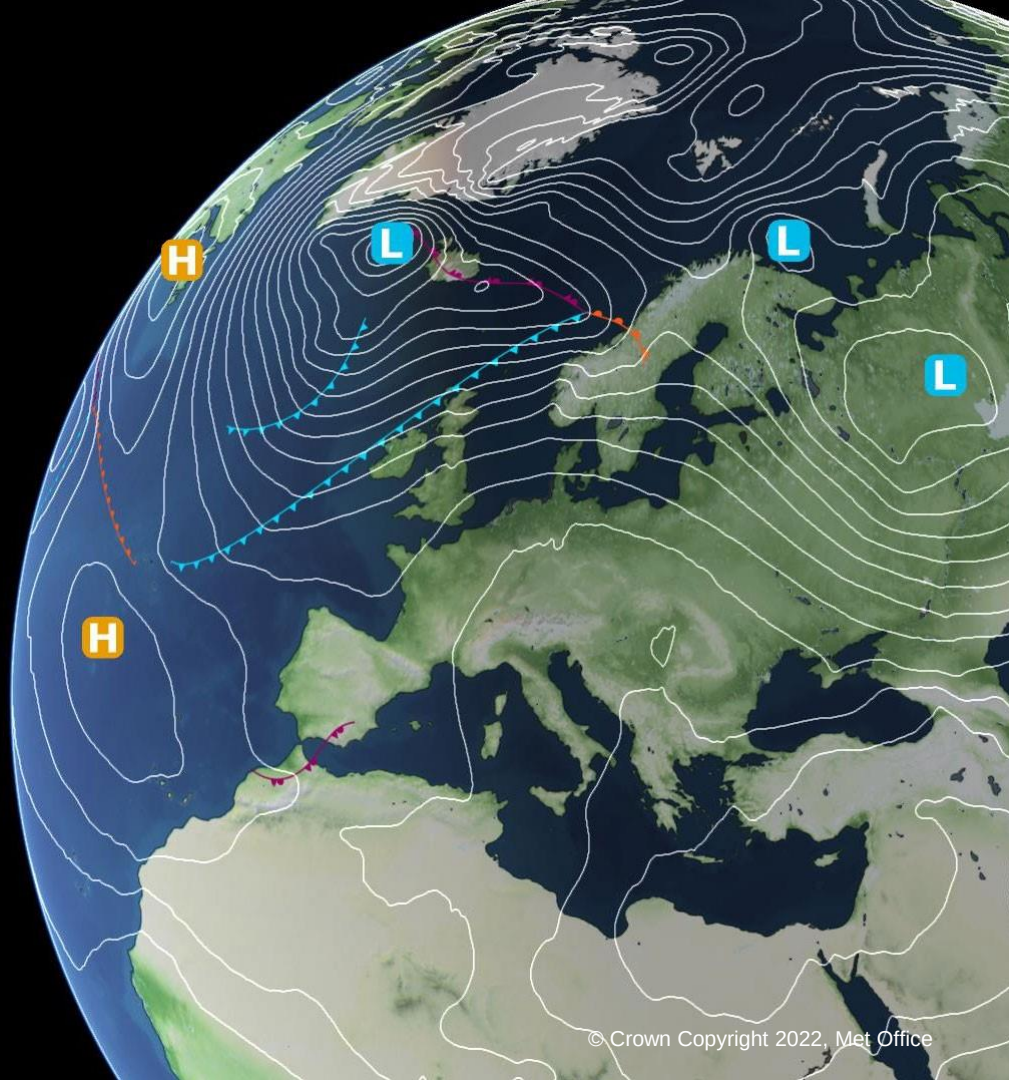
Met Office

OGC-API EDR

An introduction to the
Open Geospatial Consortium
Environmental Data Retrieval API
and how it differs from other OGC standards.



Met Office

OGC data retrieval web services

- Map Service
- Feature Service
- Coverage Service

Met Office

Feature Service

- Provides remote access to Feature datastores
- A Feature is a description of a real world object such as a road or river gauge observation
- Utilises the CQL language to support complex queries
  which provides support for operators such as: like, between, in -, +,*, /,=
- WFS, the latest iteration of the capability being OGC API Features

Met Office

Coverage Service

- Provides remote access to coverage datastores
- Coverage examples include raster images, digital elevation models and Meteorological data cubes
- Supports sub setting by Trimming and Slicing
- WCS and OGC API Coverages providing the latest iteration

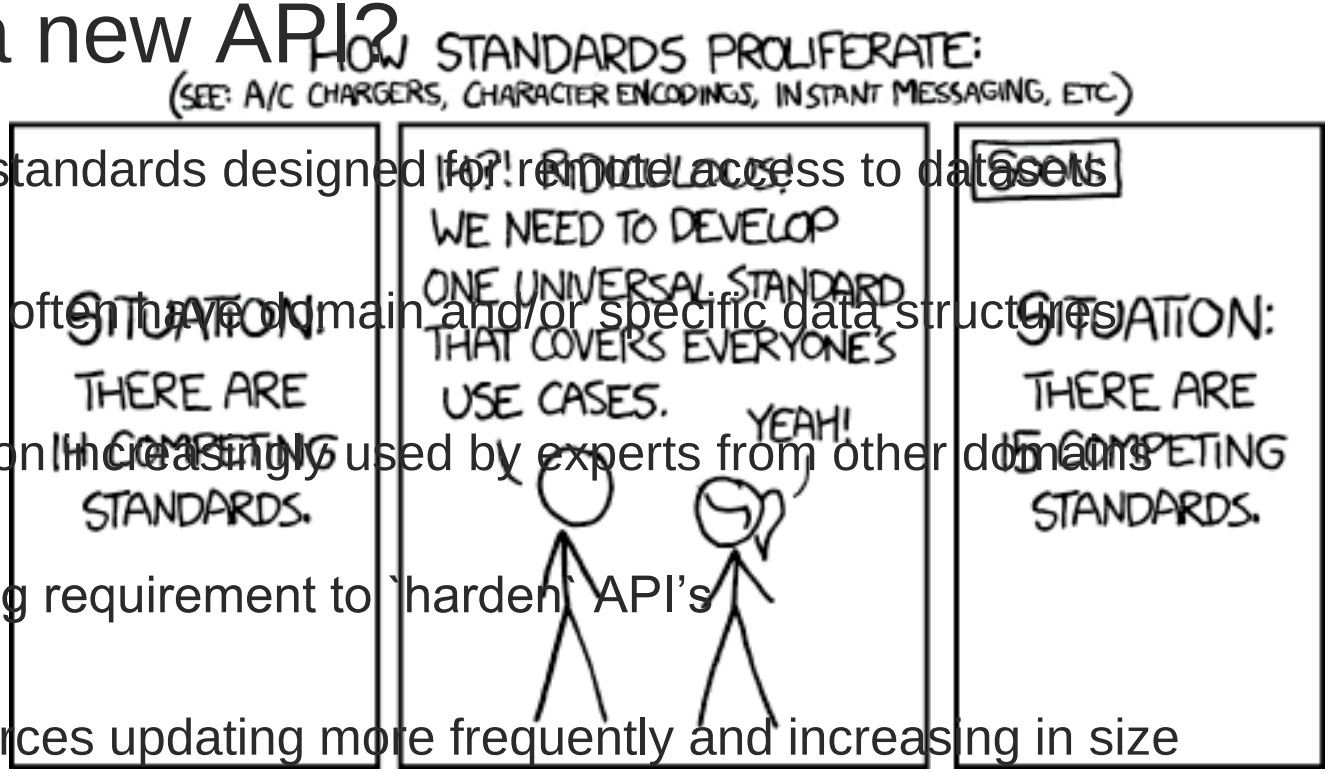
Met Office

Map Service

- Serves georeferenced images
- Query interface independent of the underlying datastore
- Basic data value access provided by GetFeatureInfo functionality
- WMS with next generation being delivered by OGC API Maps

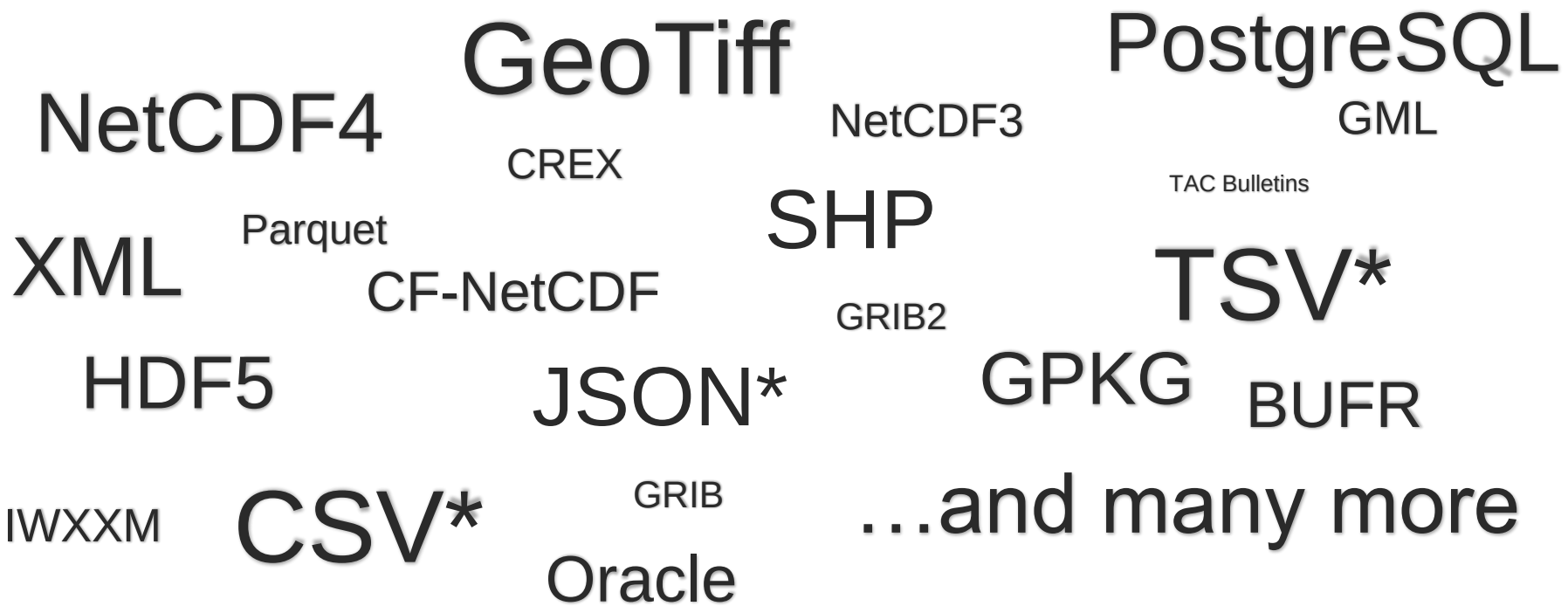
Why a new API?

- Existing standards designed for remote access to datasets
- Datasets often have domain and/or specific data structures
- Information increasingly used by experts from other domains
- Increasing requirement to `harden` API's
- Data sources updating more frequently and increasing in size



https://xkcd.com/927

Background



* many variations exist and additional information is required to understand a response in the format

Met Office

OGC API EDR

- Treat all data as information at a location and time
- API authors decide how best to respond to the request
- Create separate path resources for each query function
- Don’t require support for all the query patterns
- Limit query parameter functionality to defining range values

Data published as ‘Collections’

• A Collection is a set of data that shares the same coordinate dimensions (i.e. the same units of space, height and time)

- All data in a collection must support the advertised format and coordinate transformations
- An EDR collection behaves in much the same way as a database view

Met Office

Data discovery

- id - Unique identifier for the Collection used in the URL
- title - Name of the Collection
- description - Brief text description of the Collection
- links - URLs to information relevant to the Collection
- extent - Spatio-Temporal bounds of the Collection
- data_queries - Definition of the EDR queries supported by the Collection
- parameter_names - Definition of the data parameters in the Collection

Query (Sampling) types

- Position - /collection/{collectionid}/position?
- Radius - /collection/{collectionid}/radius?
- Area - /collection/{collectionid}/area?
- Cube - /collection/{collectionid}/cube?
- Trajectory - /collection/{collectionid}/trajectory?
- Corridor - /collection/{collectionid}/corridor?
- *Locations* - /collection/{collectionid}/locations/
- *Items* - /collection/{collectionid}/items/
- *Instances* - /collection/{collectionid}/instances/

Met Office

Query (Sampling) types (instances of collection)

- Position - /collection/{collectionid}/instances/{instanceid}/position?
- Radius - /collection/{collectionid}/instances/{instanceid}/radius?
- Area - /collection/{collectionid}/instances/{instanceid}/area?
- Cube - /collection/{collectionid}/instances/{instanceid}/cube?
- Trajectory - /collection/{collectionid}/instances/{instanceid}/trajectory?
- Corridor - /collection/{collectionid}/instances/{instanceid}/corridor?
- *Locations* - /collection/{collectionid}/instances/{instanceid}/locations/
- *Items* - /collection/{collectionid}/instances/{instanceid}/items/

Parameter metadata

- **Description** - A text label for the parameter
- **Units**
  - **Label** – A text label for the units
  - **Symbol**
    - **value** - The symbol used to represent the units
    - **type** – Unique identifier for the units (ideally an URI to a common shared registry)
- **ObservedProperty**
  - **id** – Unique identifier for the parameter (ideally an URI to a common shared registry)
  - **Label** – the formal name for the parameter
- MeasurementType
  - Method – The statistical process involved in deriving the value
  - Period – The time period that the process occurs over (defined as an ISO8601 period)


```
"air_temperature": {
    "type": "Parameter",
    "description": "Air temperature is the bulk temperature of the air, not the surface (skin) temperature.",
    "unit": {
        "symbol": {
            "type": "https://qudt.org/vocab/unit/K",
            "value": "K"
        },
        "label": "Kelvin"
    },
    "observedProperty": {
        "id": "http://vocab.nerc.ac.uk/standard_name/air_temperature/",
        "label": "Air Temperature"
    }
}
```

Queries built around core query parameters

- **COORDS** – Spatial coordinates defined as Well Known Text (WKT)
- **DATETIME*** – time range based on the ISO8601 standards
- **Z*** – vertical range selection
- **PARAMETER_NAMES** – comma delimited list of the parameters
- **F** – Data format to return the data in
- **CRS** – Coordinate reference system to return the data in

(and also defines the CRS that COORDS values are defined in)

* only when data has the dimensional component

Position: extract data for a point location

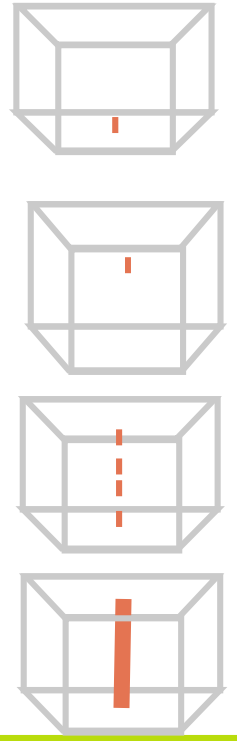
(Can also be combined with datetime)

coords=POINT(*X Y*)

coords=POINT(*X Y*)&**z**=Z_1

coords=POINT(*X Y*)&**z**=Z_1,Z_2,Z_3,*Z4*

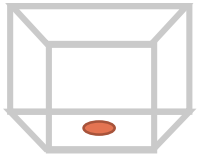
coords=POINT(*X Y*)&**z**=Z_1/Z_4



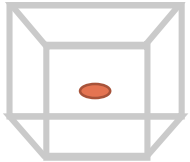
Radius: extract data within a defined radius

(Can also be combined with datetime)

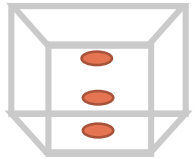
coords=POINT(*x y*)&**within**=*r*&**within-units**=*u*



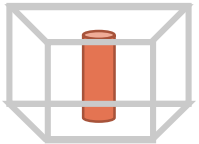
coords=POINT(x *y*)&**z**=Z_1&**within**=r&**within-units**=u



coords=POINT(x *y*)&**z**=Z_1,Z_2,Z_3,*Z4*&**within**=r&**within-units**=u



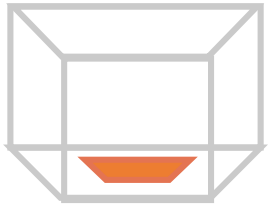
coords=POINT(x *y*)&**z**=Z_1/Z_4&**within**=r&**within-units**=u



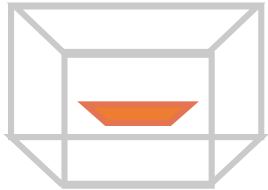
Area: extract data for a 2D geospatial domain

(Can also be combined with datetime)

coords= POLYGON(x_1 y_1, x_2 y_2, x_3 y_3, x_4 y_4, x_1 y_1))



coords= POLYGON(x_1 y_1, x_2 y_2, x_3 y_3, x_4 y_4, x_1 y_1))&**z**=Z_1



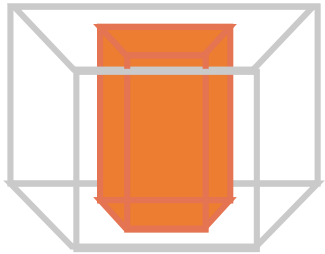
coords= POLYGON(x_1 y_1, x_2 y_2, x_3 y_3, x_4 y_4, x_1 y_1))&**z**=Z_1,Z_2,Z_3,$Z4$

coords= POLYGON(x_1 y_1, x_2 y_2, x_3 y_3, x_4 y_4, x_1 y_1))&**z**=Z_1/$Z4$

Cube: extracts data for a 3D domain geospatial domain

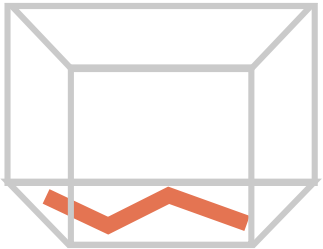
(Can also be combined with datetime)

bbox=minX,minY,maxX,maxY&**z**=Z_1/Z_2

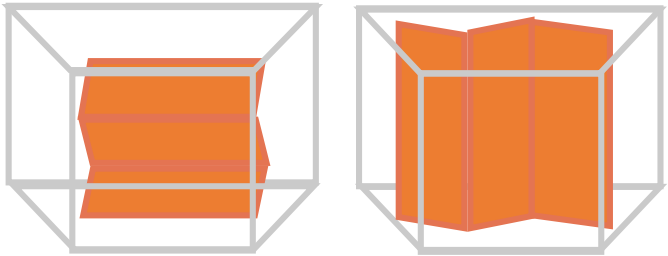


Trajectory: extracts data along a defined path

coords=LINESTRING(X_1 Y_1 , … X_n Y_n)



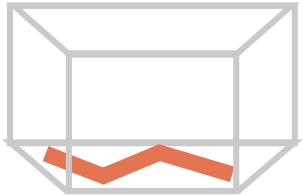
coords=LINESTRING(X_1 Y_1 , … X_n Y_n)
(can be combined with the z and datetime)



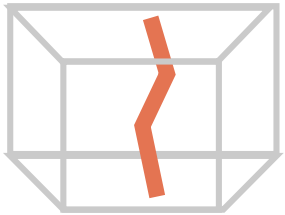
Met Office

Trajectories can have more complex height and time coordinates

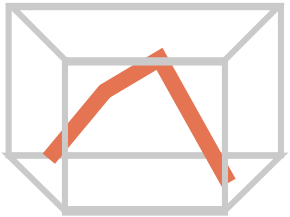
coords=LINESTRINGM(X_1 Y_1 T_1, … X_n Y_n T_n)
(can be combined with the z query parameter)



coords=LINESTRINGZ(X_1 Y_1 Z_1, … X_n Y_n Z_n)
(can be combined with the datetime query parameter)

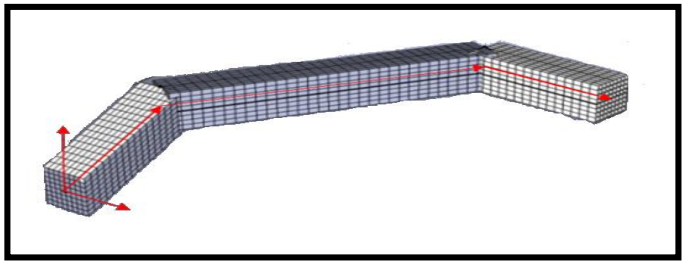


coords=LINESTRINGZM(X_1 Y_1 Z_1 T_1, … X_n Y_n Z_n T_n)



Corridor

- The same capabilities as the trajectory query but with extra values to define a corridor
    - corridor-width
    - width-units
    - corridor-height
    - height-units

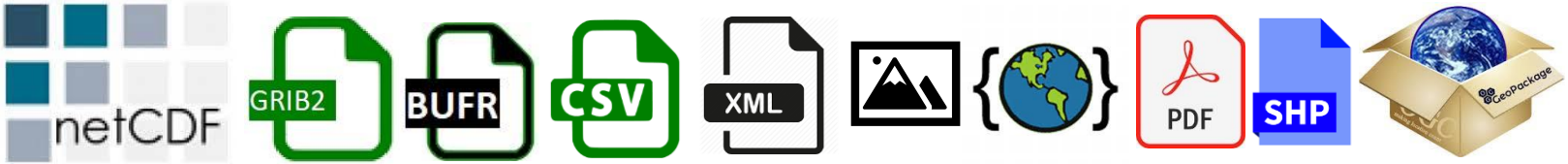


Locations

- Named location identifiers for predefined Geospatial coordinates
- API provides capability to get list of identifiers and the definition of the coordinates they represent
- Supports the other common query parameters i.e. ***parameter-name***, ***datetime***, ***f***

Met Office

Items



- There will always be a need to return predefined data objects
- /items lists the available data objects (each with a unique identifier id)
- /items lists can be filtered by bbox and datetime (with paging support)
- Objects requested by their identifier (/items/{identifier_id})

Met Office

Any questions?