Marine Environmental Data and Information Network (MEDIN) Technical Report

A collaborative pilot project on the Open Geospatial Consortium (OGC) Environmental Data Retrieval Application Programming Interface (API) Standard, undertaken by the Centre for Environment Fisheries and Aquaculture Science (Cefas) and The Archive for Marine Species and Habitats Data (DASSH)

Authors: Oliver Williams[1], Kevin Paxman[2] & Dan Lear[2]

[1] Centre for Environment Fisheries and Aquaculture Science (Cefas), Lowestoft, UK

[2] DASSH at the Marine Biological Association (MBA), The Laboratory, Citadel Hill, Plymouth, PL1 2PB, UK

Date: April 2023

Contents

1. Executive Summary

The Marine Environmental Data and Information Network (MEDIN) have driven a pilot project undertaken in collaboration by the Centre for Environment Fisheries and Aquaculture Science (Cefas) and The Archive for Marine Species and Habitats Data (DASSH) to implement the Open Geospatial Consortium (OGC) Environmental Data Retrieval (EDR) Application Programming Interface (API) Standard (OGC-EDR-API). The implementation was successful although the interpretation of the standard as well as the number of datasets delivered via this method is different for each organisation. The amount of effort, resources and timespan required to deliver implementation are significant, but vary according to the suitability of existing infrastructure and availability of technical expertise. Any decisions to use limited resources to further adopt the OGC-EDR-API standard should be balanced against competing priorities. Potential use cases and user demand need assessing within the MEDIN community before wider rollout can be recommended.

2. Introduction and Background

Introduction
MEDIN's vision is that all UK marine data are Findable, Accessible, Interoperable and Reusable (FAIR). A key objective for MEDIN is to support the UK marine sector to implement globally and cross-domain interoperable marine data services e.g. machine-readable Application Programming Interfaces (APIs) for our Data Archive Centres (DACs) and others. This report describes the results from a pilot project undertaken by two DACs (Cefas and DASSH) that implemented and tested the recently published Open Geospatial Consortium (OGC) Environmental Data Retrieval (EDR) API standard.

Enabling direct access to data from disparate Data Archive Centres using the OGC-EDR-API standard
Since 2016, MEDIN has encouraged its data centres to make it easier for users to gain direct access to the datasets that they find on the MEDIN portal. Users of the portal are frustrated when they cannot go directly to data from a search. Several DACs have made technological changes to their systems to enable direct access but to date there has not been a consistent approach across the MEDIN DACs. MEDIN funding (matched by DACs themselves) was provided to Cefas and DASSH to implement the OGC-EDR-API standard for their data holdings to trial how it works for disparate marine data and provide feedback to MEDIN and the OGC in order to help the wider UK marine community understand the implications and benefits of adopting the OGC-EDR-API. The overall aim for this work was to streamline access to UK marine data, and this pilot helped improve understanding as to whether the OGC-EDR-API standard may be the appropriate means to do so.

What is OGC-EDR-API standard?
The OGC-EDR-API standard is part of the OGC API suite of standards and is documented on Github (https://github.com/opengeospatial/ogcapi-environmental-data-retrieval) . OGC API standards define modular API building blocks to spatially enable Web APIs in a consistent way. In practice this allows end user to use the same query-based approach to discover and explore spatial data from different data sources and for different data types.

Why does MEDIN want to implement the OGC-EDR-API standard?
MEDIN wants to recommend an API standard for the UK marine community to facilitate interoperability, thereby increasing access to the UK's marine data resources. That standard must meet the specific needs of the marine community and MEDIN wants to ensure that the recommended standard is interoperable internationally and cross domain, recognising that the oceans, seas, atmosphere, cryosphere and land are interconnected and interdependent. Moreover, we want to avoid duplicating work carried out elsewhere.

3. Infrastructure and Technologies Used

Cefas and DASSH have different platforms and infrastructures which were used to implement the OGC-EDR-API standard. These are described below.

Cefas

The location of the public facing Cefas API (published on 07/04/2022) is:

https://data-api.cefas.co.uk/index.html?urls.primaryName=Cefas%20OGC%20Environmental%20Data%20Retrieval%20API

Cefas' core supported data infrastructure, referred to as the Cefas Data Portal (CDH), is located on Microsoft Azure database infrastructure. The language used to convert the data from the CDH into the OGC-EDR-API standard was C# (https://learn.microsoft.com/en-us/dotnet/csharp/).

The Swagger platform (https://swagger.io/docs/specification/about/) has been used to describe the service using the Open API specification https://github.com/OAI/OpenAPI-Specification.

Prior to this pilot project the CDH already served all datasets via API in a non-standardised manner, these are described at the following address.

https://data-api.cefas.co.uk/index.html?urls.primaryName=Cefas%20Data%20Portal%20API

DASSH

The DASSH data infrastructure is built on a PostgreSQL (https://www.postgresql.org/) database with the PostGIS (https://postgis.net/) extension to enable spatial capabilities within the existing Relational Database Management System (RDBMS). Access is leveraged through an instance of Geoserver (https://geoserver.org), the open source application designed to share geospatial data using open standards including those developed and promoted by the Open Geospatial Consortium.

DASSH developed code is hosted on a GIT repository (https://gitserver.mba.ac.uk)

Our initial approach was therefore to build an API interface that utilises either the PostGIS spatial querying system, or the Geoserver Web Feature Service (WFS) system.

4. Data Selection – Methodology and Description

Cefas

The Cefas Data Portal contains over 6000 individual datasets which follow a variety of formats and standards and of which ~2500 are spatially referenced so data could only be selected from that subset.

The variety in both standard and quality of archived datasets led Cefas to decide to use only a limited number of datasets in this pilot project. However, the functionality has been developed to allow any selected dataset to be converted and served using the OGC-EDR-API standard as long as it is geospatially and temporally referenced. Cefas will await the final outcomes of this project and subsequent discussions within the marine data community regarding the potential adoption of the OGC-EDR-API standard before deciding whether to serve further datasets via this route. The selected datasets used for this pilot project are listed below alongside the MEDIN discovery metadata links.

Data Theme	MEDIN Discovery Metadata record	Data Link
Timeseries data of organic carbon in marine sediments	https://portal.medin.org.uk/portal/start.php#details?tpc=009_CEFASf5fe17ea-81d4-4ae9-b9a6-00943554c4c0	https://data.cefas.co.uk/view/18354
Timeseries data of seawater temperature records	https://portal.medin.org.uk/portal/start.php#details?tpc=009_CEFAS9d5852de-2498-4726-840e-cacb63161b07	https://data.cefas.co.uk/view/3232
Zooplankton abundance derived from plankton imagery	https://portal.medin.org.uk/portal/start.php#details?tpc=009_CEFAS6bd8041f-68bc-4b20-a2bc-645d9bda17f1	https://data.cefas.co.uk/view/20507
Sediment Particle Size Analysis (PSA) results from a fishing impact study	https://portal.medin.org.uk/portal/start.php#details?tpc=009_d1a9e99c-6c84-4f67-97d3-59260698e6d3	https://data.cefas.co.uk/view/19703
Timeseries data of marine seafloor litter	https://portal.medin.org.uk/portal/start.php#details?tpc=009_CEFAS021a5fad-4226-4f60-8e33-a15594894d4e	https://data.cefas.co.uk/view/3479
Fisheries spawning and nursery grounds sensitivity maps	https://portal.medin.org.uk/portal/start.php#details?tpc=009_CEFAS78edae85-c899-409b-ac05-1b5f6c1f68ae	https://data.cefas.co.uk/view/149

DASSH

DASSH holds biological and habitat data as point features, where points represent samples or stations, and individual records represent a species or habitat occurrence. DASSH operates a combined database which combines all our holdings in a standardised format. As previously described this is a PostgreSQL relational database with a PostGIS spatial extension. We are therefore well placed to expose the entirety of our standardised data holdings through a spatial query system.

The specific DASSH database structure does not need to be taken into account as we have created a single 'View' (the result set of a stored query on the data, which the database users can query just as they would in a persistent database collection object - https://en.wikipedia.org/wiki/View_(SQL) combining all the tables into one. Our methods and code are therefore replicable by others who use a PostgreSQL database, simply requiring the build of a single table based on their own database structure. To allow selection using meter units instead of the native coordinates (latitude-longitude), it was required to add an extra PostGIS Geography type column to our source table to go alongside to the pre-existing Geometry column.

5. Technical Implementation – Description & Links to Source Code

Cefas

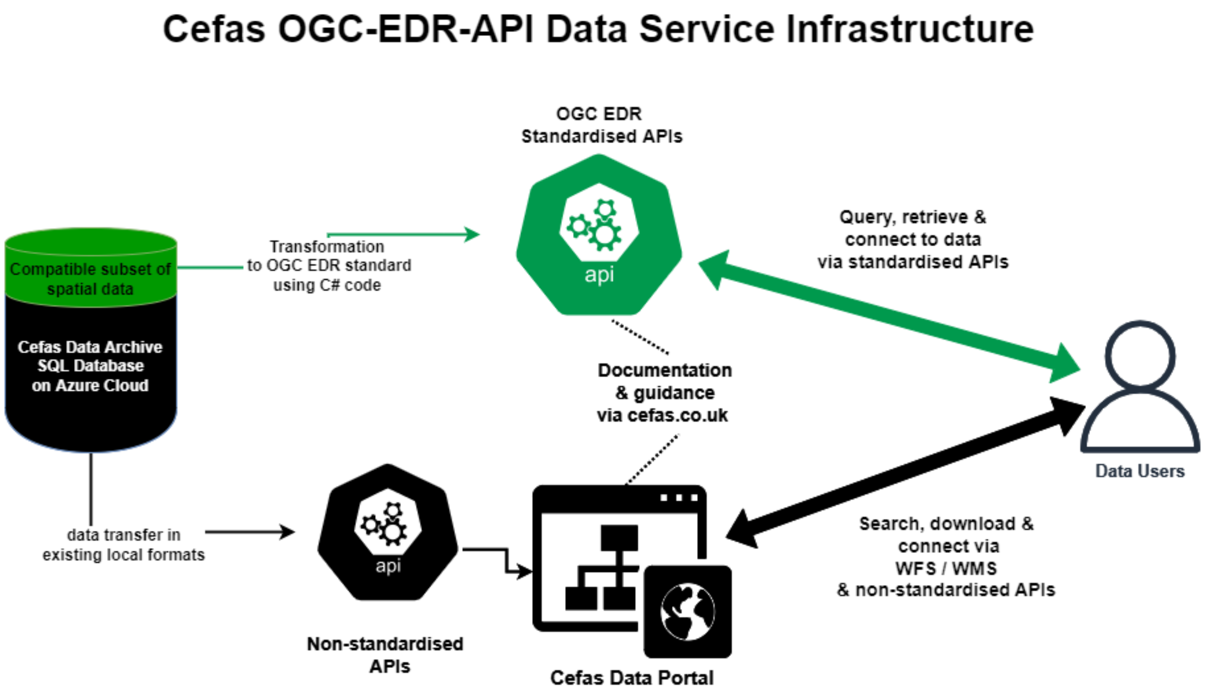

Cefas OGC-EDR-API Data Service Infrastructure
OGC EDR Standardised APIs
api
Compatible subset of spatial data
Cefas Data Archive SQL Database on Azure Cloud
Transformation to OGC EDR standard using C# code
Query, retrieve & connect to data via standardised APIs
Documentation & guidance via cefas.co.uk
Data Users
data transfer in existing local formats
api
Non-standardised APIs
Cefas Data Portal
Search, download & connect via WFS / WMS & non-standardised APIs

Figure 1 – technical diagram of Cefas infrastructure used to deliver OGC-EDR-API. See glossary for definition of terms.

The technical diagram above indicates how existing Cefas infrastructure was used to deliver selected datasets following the OGC-EDR-API standard.

The Swagger platform (https://swagger.io/docs/specification/about/) has been used to describe the service using the OpenAPI specification (https://github.com/OAI/OpenAPI-Specification/blob/main/versions/3.0.3.md). The Swagger endpoint groupings and the descriptions are taken directly from the OGC reference site (https://docs.ogc.org/is/19-086r4/19-086r4.html), however some references to features which have not been supported (as described below) were removed. The OGC term 'collection' is mapped to individual datasets (known as 'recordset' on the CDH). The OGC term 'instances' is mapped to 'versions' of recordsets on the CDH.

The OGC-EDR-API standard is very flexible and is designed to support querying many different types of dataset using different infrastructures and technologies. The limitations of both the CDH infrastructure and the practicalities of delivering a stable, usable system have led to some elements of the OGC-EDR-API standard being limited or not implemented for the Cefas API, as detailed below.

Elements not supported / limitations in the Cefas implementation

- As the CDH uses a geometry type based on actual distances in metres to store spatial data, the buffer function of the standard (which is based on fractions of degrees) is therefore unusable 'as is'. Hence the CDH converts the query inputs into metres to calculate the query and then reconverts back to the original type for the output.

- The underlying geometry of the CDH is 2-dimensional, hence although we include in our Swagger query types from the standard which support 3-dimensional querying (e.g. including depth as well as position), depth / height parameter inputs are ignored.
- The Well Known Text parser in Azure Database only supports the 'LINESTRING' query, hence the 'LINESTRINGM', 'LINESTRINGZ', and 'LINESTRINGZM' variants from the standard are not supported.
- While Cefas is satisfied that the JSON outputs of the metadata URLs match the specification, we did not find the naming and casing specification given in the standard connotation for the XML, YAML and HTML formats clear enough to ensure these are compliant with the standard.
- Geometry data is returned in GeoJSON format which is most appropriate for the data in the CDH. The standard mentions other formats that are grid based like NetCDF but as the CDH data is not grid based this was not appropriate to output.
- Only one date field from recordsets has been included for use in the queries as support start / end dates would not have been possible within the timeframe and resources of the pilot project.
- The CDH only uses data from the following spatial reference system (SRS) SRIDS (Spatial Reference System Ids), 4236 (WGS 84), 3857 (Web Mercator), 4322 (WGS 72), and 27700 (OSGB) so only these are used in outputs. While input queries can use any reference system from the CDH the geometry returned is always 4236 (WGS 84).
- The entire temporal range of the recordset is given in outputs, as opposed to listing each unique date because outputting large numbers of dates causes a JSON error.
- The named locations supported are limited to ICES Rectangles, ICES Areas and OSPAR Regions only as these are best suited to marine data.

DASSH

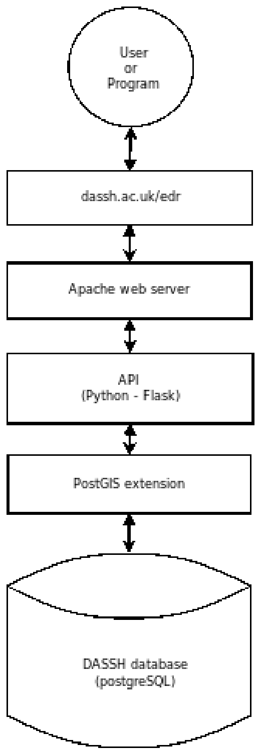


Figure 2 - technical diagram of DASSH infrastructure used to deliver OGC-EDR-API. See glossary for definition of terms.

The DASSH team specialises in application development using the Python language and we therefore envisaged building the tool in Flask, a straight-forward web development framework which allows the construction of APIs.
The code for this has been uploaded to an open Gitlab project (https://gitserver.mba.ac.uk/external/data_team/edr). Following trials, it was decided to access the database directly and translate the OGC-EDR-API queries into PostGIS. For example a radius can be selected using 'ST_DWithin', an area can be selected with 'ST_Intersects' and a corridor using 'ST_Buffer'. This removes an additional layer of abstraction of having to go through Geoserver and results in faster queries. In addition, the WFS/ECQL syntax used by Geoserver does not offer the same breadth and power of functions as the PostGIS library.
Basic metadata endpoints were created (e.g. '/', '/collections/') along with a sample of query endpoints ('/radius', '/area'), in order to test the concept. We used the `psycopg2` python library (https://pypi.org/project/psycopg2/) to connect to our database. Some of these interactions have been off-loaded to our own custom database library (dbossh_library), however these are only wrappers for psycopg2 functions. This serves to keep our database specific information, including passwords, out of the main body of code.

Using this approach we successfully demonstrated that features could be extracted from our database using OGC-EDR-API spatial queries.

Pygeoapi investigation

During the initial project meeting, 'pygeoapi' (https://pygeoapi.io/) was brought to our attention, which is a generic implementation of the OGC Core API modules, as well as some elements of the OGC-EDR-API standard. This therefore seemed the ideal and quickest way to implement OGC-EDR-API standard at DASSH. Work was therefore paused on our own custom implementation of OGC-EDR-API whilst we investigated whether this pre-developed package could be utilised instead.

Installation is recommended through a virtual environment, which did simplify the process, however there were still some issues installing, which required some research of the errors reported (for example '*error: invalid command 'bdist_wheel*' required the installation of the ''wheel' library for Python). Once installed, only basic configuration was required to get the implementation up and running, which was initially very positive.

Our first attempt to get our data connected to our pygeoapi instance, was to use the WFS Provider to connect to our Geoserver instance via WFS. Unfortunately, we ran into problems with this, with all queries returning a '400 Error'. After much trial and error, we discovered that specifying 'OGR_WFS_PAGING_ALLOWED: NO' allowed us to connect successfully. Unfortunately, no features were displayed under the 'items/' end point, and queries seemed to hang without returning any results. It is a possibility that our database is too large to be used in this way (currently containing ~5 million records). It is possible this may be resolved by adjusting TIMEOUT values, however this line of enquiry was abandoned at this point as such a solution would be unusable in practical terms.

During investigations it was discovered that the OGC-EDR-API part of pygeoapi has only been implemented for certain Providers (which do not include WFS). This meant that to use the OGC-EDR-API queries, our data would have to be provided as an array type. It is clear from this fact, and that the only export was a coverageJSON, and that the OGC-EDR-API implementation here at the time of writing is only intended for coverages and is not designed for features.

To ameliorate this shortcoming we investigated the possibility of exporting our database into a NetCDF file (https://www.unidata.ucar.edu/software/netcdf/) and connecting that to pygeoapi. We successfully converted a sample of our database into NetCDF using the python 'pandas' and 'xarray' libraries. We connected this file to pygeoapi and for the first time were able to query and return our data using this API. However, scaling up to access the whole database was found to be unfeasible. Given the vast range and irregularity of our sample locations, using these as dimensions for a standardised array was resulting in huge datasets. In addition, since our records are occurrence-based rather than sample-based, we have multiple features per position, which required the addition of an extra dimension to account for this. Our output arrays therefore had dimensions numUniqueLats x numUniqueLongs x numUniqueDateTimes x maxRecordsInSample which means a sample size of just 10,000 database records results in ~54 billion values.

Since the netCDF exporting plan had reached an impasse, our only option for using pygeoapi was to modify it to connect the OGC-EDR-API queries with the feature Providers. However, based on the perceived difficulty of doing so, including the fact that most of the OGC-EDR-API queryTypes (e.g. radius, corridor, cube) had not yet been implemented in pygeoapi, it was decided to go back to our original plan of building the API ourselves. We suspect that pygeoapi will be a very useful tool in the future, once it is more feature complete and specifically allows OGC-EDR-API queries on feature types, however it is currently not suitable for our purpose.

The OGC-EDR-API Specification

Unfortunately, we encountered a large number of errors in the specification, particularly concerning inconsistencies in the requirements specified in the Annex. This included missing requirements, requirements that contradict other requirements, and requirements that contradict the main body of text. Significant errors have been reported by the DASSH team via the GitHub project (https://github.com/opengeospatial/ogcapi-environmental-data-retrieval) during the lifespan of this project, as is to be expected with emerging standards. The result has been a specification that, in the form at the time, was very difficult to follow. However v1.0.1 has since been released which has addressed many of the problems described. The contributions of DASSH and other project partners have helped to improve the standard for future, more straightforward implementation by other parties.

Subsequent Stand-alone API development

We continued to develop our own API implementation using Flask. The instance is publicly accessible at: https://dassh.ac.uk/edr.
We have added parameter validation that we believe conforms to the standard. We have constructed the metadata through a combination of manually altering the example metadata from the official Swagger page (https://app.swaggerhub.com/apis/OGC/ogcapi-edr-1-example-1/), and automating the generation from our database. We systematically went through the requirements in the specification one-by-one and tested each in turn.

Instances

The OGC-EDR-API specification permits the use of instances/ which are a subdivision of a collection. The exact usage is left to the specific implementation, however from the name it can be reasonably guessed that these should be used for version control. However, in the case of the DASSH implementation, we have our entire data holdings as a single collection, with the goal of enabling our holdings to be queried as a whole. We therefore utilised the instances/ convention to represent individual datasets/surveys.

Work was invested in generating metadata for each individual instance from our database, both unique (e.g. title/description, spatial/temporal extent,) and standard (the OGC-EDR-API functions and field descriptors). We have over 5000 datasets so a script was written to generate these json files, populated from our database. API functions then had to be implemented to fetch this metadata from individual instances and also to combine them all together when the generic instances/ was requested. Additionally, all pre-existing implemented query functions can now be restricted to a specific dataset by specifying that instance.

Query Endpoints

Through this project we have implemented the following query endpoints:
- position
- radius
- area
- cube
- corridor

We believe these cover a wide range of use cases. Cube and corridor both notably add to our current options.

Depth

As a marine DAC, depth is a key component of our sample positions. We were therefore pleased to find that a vertical value ‘z’ is an integral part of all OGC-EDR-API queries. We elected to keep the value positive, but invert it to treat it as a depth. So, for example a z value of 5 will equate to a depth at 5m below a z value of 0 (the sea surface).

Output

All data queries are returned as geoJSON by default. This has been coordinated with our CEFAS partners. HTML format can also be requested if desired. We note that due to differing internal fieldnames of our respective databases, our outputs are not able to be combined immediately, without coercion of the resulting property names. In addition metadata are returned as JSON, optionally as HTML.

Swagger

A swagger implementation of the API is available at: https://app.swaggerhub.com/apis-docs/DAS365/dassh-implementation_of_ogc_api_environmental_data_retrieval/

This is linked from within the capabilities metadata document as its ‘service-doc’ link, and allows for testing and experimentation through a web interface.

6. Demonstrator Application

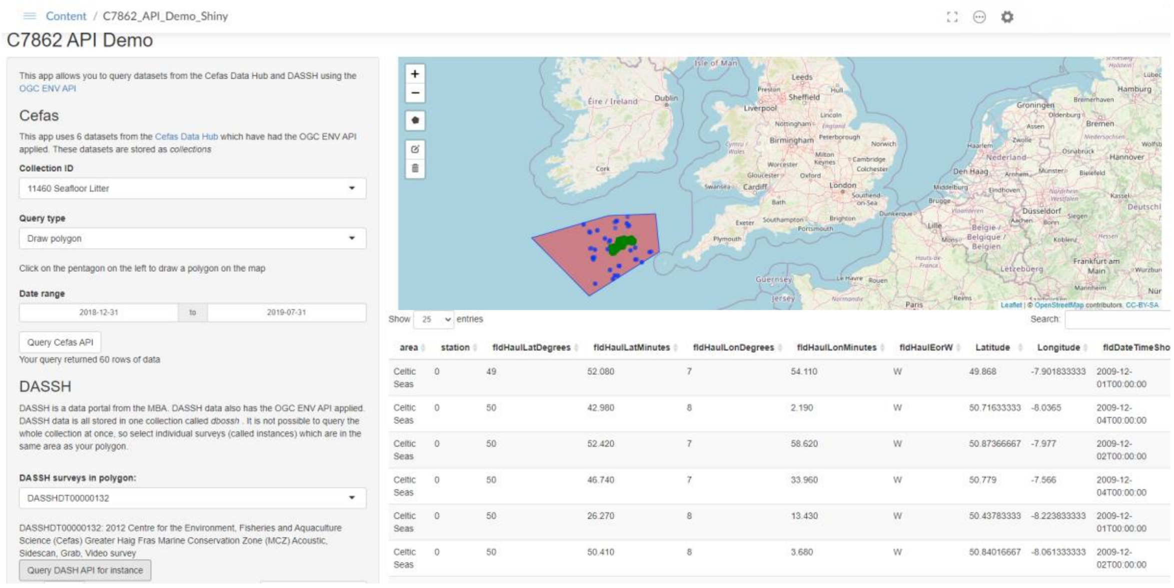
Overview

Beyond implementation of the standard, this pilot project also included the development of a demonstrator application to query different OGC-EDR-API implementations in a unified way, to showcase the interoperability improvements which can be achieved using the standard.

The application was not primarily designed for public use but rather process of its design was a simple test of the use of the standard and an exploration of the OGC-EDR-API implementation decisions and infrastructure delivered by the Cefas and DASSH API developers. For this reason, the demonstrator app and its base code had not been published at the time this report was produced but may be available on request to Cefas via data.manager@cefas.co.uk.

Cefas scientific developers utilised the shiny package in R, along with RStudioConnect infrastructure to create an interactive application that allowed users to run simple queries using the OGC-EDR-API standard to discover data that has been published via the standardised API for this pilot project from Cefas and/or DASSH.

In practice the application was developed to allow users to define simple details such as date range, dataset ID and spatial area extents, and then translate these into an API query. The query would then return results as required and display them within the application in a usable format. This allows non-technical users to make of use of the API without requiring coding skills.


Content / C7862_API_Demo_Shiny

C7862 API Demo

This app allows you to query datasets from the Cefas Data Hub and DASSH using the OGC ENV API

Cefas

This app uses 6 datasets from the Cefas Data Hub which have had the OGC ENV API applied. These datasets are stored as *collections*

Collection ID: 11460 Seafloor Litter

Query type: Draw polygon

Click on the pentagon on the left to draw a polygon on the map

Date range: 2018-12-31 to 2019-07-31

Query Cefas API

Your query returned 60 rows of data

DASSH

DASSH is a data portal from the MBA. DASSH data also has the OGC ENV API applied. DASSH data is all stored in one collection called *dbossh* . It is not possible to query the whole collection at once, so select individual surveys (called instances) which are in the same area as your polygon.

DASSH surveys in polygon: DASSHDT00000132

DASSHDT00000132: 2012 Centre for the Environment, Fisheries and Aquaculture Science (Cefas) Greater Haig Fras Marine Conservation Zone (MCZ) Acoustic, Sidescan, Grab, Video survey

Query DASH API for instance

Show 25 entries Search:

area	station	fldHaulLatDegrees	fldHaulLatMinutes	fldHaulLonDegrees	fldHaulLonMinutes	fldHaulEorW	Latitude	Longitude	fldDateTimeSho
Celtic Seas	0	49	52.080	7	54.110	W	49.868	-7.901833333	2009-12-01T00:00:00
Celtic Seas	0	50	42.980	8	2.190	W	50.71633333	-8.0365	2009-12-04T00:00:00
Celtic Seas	0	50	52.420	7	58.620	W	50.87366667	-7.977	2009-12-02T00:00:00
Celtic Seas	0	50	46.740	7	33.960	W	50.779	-7.566	2009-12-04T00:00:00
Celtic Seas	0	50	26.270	8	13.430	W	50.43783333	-8.223833333	2009-12-01T00:00:00
Celtic Seas	0	50	50.410	8	3.680	W	50.84016667	-8.061333333	2009-12-02T00:00:00

Figure 3 – a screenshot taken from the Cefas pilot demonstrator application

Lessons Learned

Key lessons learned from the development of the demonstrator app are listed below.

- The developers found it useful to use each DACs Swagger implementation to test queries, then use the results to construct query URLs for the application.
- The visualisation of returned results for the application was kept very simple, partly due to time constraints for this pilot and partly because it would not be feasible to support complex

visualisations that would be appropriate for such a wide range of data types available across both DACS.

- The DASSH OGC-EDR-API implementation translated the standard term 'collection' as being all current published data across all datasets, whereas the Cefas implementation translated the term 'collection' as each discrete dataset. The standard itself is very flexible, allowing such different implementation decisions to be made, according to both the setup of existing infrastructure and user need. One downside of this flexibility is that spatial queries with a wide date or spatial range run on a specific collection for the Cefas API will return results within a reasonable time frame, however the same extends run on the DASSH API will often result of server timeouts or take an unfeasible amount of time to complete.
- The flexibility of the OGC-ENV-API standard allows implementors to determine their own terms for spatial information in queries, for example the Cefas query used the term 'boundingBox' whereas the DASSH query used the term 'bbox'. While being relatively minor, such differences did make it harder to develop standardised code to query both APIs at the same time.
- The application developers found it difficult to write queries to explore and return the data from each API, without first determining some information on the contents of the datasets available. While the OGC-EDR-API standard does facilitate the use of some metadata queries to discover some high level dataset information, in practice future development of applications would be most efficiently delivered by using both the discovery metadata available in parallel with information derived using the OGC-EDR-API standard. Hence the utility of the OGC-EDR-API as a stand-alone discovery tool for science developers is limited.
- The scienctific developers could not determine a simple method to return a complete dataset, without having to specify spatial or temporal parameters. A workaround was found by using the metadata query to return the bounding box of the whole dataset and these values were subsequently used to run a polygon query to return all results. It would require far less effort if such a 'return all' functionality was including within the OGC-EDR-API standard.

Conclusions

The OGC-EDR-API standard did facilitate some basic functions which made it easier to query and return data from both the Cefas and DASSH implementations, but in practice the limitations of the standard and the flexibility by which it can be interpreted, made its use as a way of exploring and returning data without further reference to specific metadata impractical.

7. Resources Used

Resources used are given below as indicators for other organisations which may want to estimate effort required to implement and use the OGC-EDR-API standard. They do not include other costs associated with this pilot which would not be as relevant for other organisations (e.g. project management, report writing and delivery of final workshop).

There are several key factors which will heavily affect the amount of effort required including:

- The number of different datasets selected for implementation and the level of standardisation within them (as a greater diversity and overall number of datasets will require greater effort).
- The degree of functionality implemented within the API service (there are a number of functions which may be selected for implementation within the OGC-EDR-API standard according to user need, resource availability and status of available datasets).
- The overall stability of the systems used, as less stable infrastructure will likely require more effort to ensure services remain live.
- The maturity of support for systems. If existing infrastructure is already in place prior to implementation of the OGC-EDR-API standard and is already subject to technical support, then this will likely take less additional effort to ensure long term support for the service.

Cefas

Initial review of datasets and implementation of OGC-EDR-API standard

Data Governance Specialist Time - 10 days

Senior Software Developer Time - 20 days

Testing of API Endpoints – 5 days

Total: **35 days**

Familiarisation with standard and development of demonstrator in Shiny R

Data Governance Specialist Time – 5 days

Scientific Developer Time - 10 days

Total: **15 days**

DASSH

Research (mostly making analysis and understanding of specification) - 5 days

Investigating options (including aborted Pygeo API development) - 12 days

Building the API in Flask - 18 days

Populating metadata - 2 days

Testing, fixes and further improvements - 8 days

Total: **45 days**

8. Lessons Learnt

The challenges to implementation of the OGC-EDR-API standard can be broken down as system status, existing data readiness and resource availability. The overall amount of resource required to complete the task is dependent on those factors.

- **System Status**

  Cefas

  Prior to this pilot commencing, the Cefas Data Portal was already serving all published data via API, using a bespoke system developed in-house and documented via the Swagger interface (https://swagger.io/docs/specification/about/). This allowed developers to translate the OGC-EDR-API standard within pre-existing infrastructure, limiting the amount of time needed to complete the implementation.

  DASSH

  The existing DASSH database structure, database systems and publishing platforms ensured that DASSH was well placed to deliver the new API standard relatively easily. By far the biggest utilisation was, as previously mentioned, related to the interpretation and understanding of the standard itself, including the mitigations required for omissions or inaccuracies in the documentation.

- **Data Readiness**

  Cefas

  As described in the data selection section of this report, the variety of Cefas datasets in terms of theme, data structure and quality led to the implementation of the standard to only six datasets for the pilot. This is not a limitation of the OGC-EDR-API standard itself but rather a limitation of the readiness of Cefas' data for implementation.

  DASSH

  As previously mentioned, within DASSH we took the approach of delivering access to our entire standardised data holdings. Whilst increasing the breadth of data accessible via the API, it does result in lengthy query times and can cause occasional timeouts which would need to be correctly handled by the querying service.

- **Resource Availability**

  Cefas

  The availability of in-house developer time was ringfenced for the last quarter of FY2021-2022 in order to deliver what we consider a reasonable viable product for our OGC-EDR-API standard implementation. Arranging for a consistent level of focused development over a shorter period allowed for a more efficient delivery.

  DASSH

  The MEDIN funding allowed for allocated developer time to be utilised and ensure the OGC-EDR-API development work was prioritised. Without MEDIN funding the work would not have taken place.

9. Conclusions and Recommendations

Cefas specific conclusions

The implementation of the OGC-API-EDR standard has provided our users with an additional way to query and retrieve a small number of spatial datasets within our data portal. The variety of data types and standards published by Cefas (both spatial and non-spatial) means that we would never be able to serve all data via this method, hence the applicability of this method as a way of exploring all datasets within our portal is limited.

Our demonstrator app has shown that the standard could be useful in future, as a way of querying datasets across multiple archive centres. If hurdles such as processing power are overcome, this might enable technical end users of our portal to access and use data from across the MEDIN DAC network and beyond.

DASSH specific conclusions

We have shown that without a prohibitive amount of work we were able to expose our data holdings via the OGC-EDR-API standard, and indeed have done so for the entirety of our database. Our API instance will continue to be maintained in the future. It will therefore be available to use and, along with the public codebase, can be used as an example to follow by other institutes. It will be of particular use to those who utilise a postGRES database and are familiar with Python/Flask.

However, in our view the new standard confers no obvious benefit within the DASSH archive over our existing WFS service, provided by our Geoserver instance. The only queries that confer added value are the cube and corridor queries, especially their ability to select sampling heights. However, given the irregular nature of our samples, it is unlikely to be utilised by our uses in the near future.

Joint Conclusions and Recommendations

The OGC-EDR-API offers a relatively simple way for users to access data using common queries, but it is not necessarily as simple for developers to implement it on existing systems. In addition, the flexible way in which the standard can be interpreted and implemented can make it more difficult for users to query across different datasets and platforms. If the marine data community where to adopt this as a standard for APIs, there would likely need to be further agreement across organisations to better standardise implementation and MEDIN would be well placed to steer this potential initiative. Beyond implementation, the lessons learned from the development of the demonstrator application indicate that agreeing to standardise the details of documentation (for example including spatial terms within swagger) would also be beneficial for end users.

Any methods which simplify direct access to multiple data sources risk separating users from the discovery metadata records which sit alongside each distinct dataset. Such metadata contains potential crucial contextual information such as collection methodology, provenance, and conditions for use. Hence there is a risk that end users may misuse or misinterpret data which is accessed only via the OGC-EDR-API standard, although it is likely that this is a risk for any methods which facilitate programmatic access to data and not just for the OGC-EDR-API.

It must be recognised that not all marine data is spatial in nature and would therefore not be able to be served by the OGC-EDR-API standard, however this is likely to be a limitation of most API standards as without temporal / spatial components finding common elements across many types of dataset would likely be very difficult.

In summary we have found that with sufficient investment of time and resources, it is likely possible for many marine data organisations to implement the OGC-EDR-API and this may be of benefit to a range of users who have the skills and resources to programmatically retrieve data. However, the amount of investment will differ for each case and where resources are limited, it must be balanced against efforts towards other improvements such as publishing more data and improving quality. We believe that the UK marine data community will be better placed to determine both potential use cases as well as overall user demand as result of the publications of this report and the MEDIN community offers excellent fora to cover engagement on this topic going forward.

10. Glossary of Terms

Term	Definition	Relevant Links
API	Application Programming Interface - a type of software interface that allows communication between computer programmes.	https://en.wikipedia.org/wiki/API
Cefas	Centre for Environment, Fisheries and Aquaculture Science, which is a MEDIN DAC.	https://www.cefas.co.uk/
DASSH	The Archive for Marine Species and Habitats Data (UK) which is a MEDIN DAC.	https://www.dassh.ac.uk/
Discovery metadata	Information related to a dataset to facilitate discovery and use.	https://medin.org.uk/medin-discovery-metadata-standard
FAIR data	Principles which ensure data are Findable, Accessible, Interoperable, and Reusable.	https://en.wikipedia.org/wiki/FAIR_data
GeoJSON	A open standard format designed to represent simple geographical features, based on the JSON format.	https://en.wikipedia.org/wiki/GeoJSON
HTML	HyperText Markup Language, a standard markup language for web based documents.	https://en.wikipedia.org/wiki/HTML
ICES	The International Council for the Exploration of the Sea which is an intergovernmental marine science organisation.	https://www.ices.dk/
JSON	JavaScript Object Notation, an open standard format which uses human readable text to store and transmit data objects.	https://en.wikipedia.org/wiki/JSON
MBA	The Marine Biological Association, a learned society based in the UK which hosts DASSH.	https://www.mba.ac.uk/
MEDIN	The Marine Environmental Data and Information Network, a partnership of UK organisations committed to improving access to marine data.	https://medin.org.uk/
MEDIN DAC	Data Archive Centres which are accredited as part of the MEDIN network.	https://medin.org.uk/data-archive-centres
Microsoft Azure	A cloud computing service delivered by Microsoft.	https://azure.microsoft.com/
NetCDF	Network Common Data Form, a common data format which supports access, creation and sharing of scientific data.	https://en.wikipedia.org/wiki/NetCDF
OGC	The Open Geospatial Consortium, a consortium of experts committed to improve access to geospatial or location information.	https://www.ogc.org/

OGC-EDR-API	The OGC Environmental Data Retrieval API is a standard which provides lightweight interfaces to access spatial environmental data resources.	https://ogcapi.ogc.org/edr/
Open API specification	A publicly available API specification.	https://en.wikipedia.org/wiki/OpenAPI_Specification
OSPAR	An international mechanism by with 15 Governments & the EU cooperate to protect the marine environment of the North-East Atlantic.	https://www.ospar.org/
PostgreSQL	An open-source relational database management system.	https://en.wikipedia.org/wiki/PostgreSQL
Python	A high level, general purpose programming language.	https://www.python.org/ https://en.wikipedia.org/wiki/Python_(programming_language)
R	R is a programming language primarily aimed as statistical computing and graphics.	https://www.r-project.org/
SRID	Spatial Reference System, a framework used to measure locations on the surface of the Earth.	https://en.wikipedia.org/wiki/Spatial_reference_system
WFS	Web Feature Service, an interface standard which allows requests for geographical features across the web.	https://www.ogc.org/standard/wfs/
WGS	World Geodetic System, a standard used define a coordinate system which is published and maintaining by the US National Geospatial-Intelligence Agency.	https://en.wikipedia.org/wiki/World_Geodetic_System
WMS	Web Map Service, a standard protocol developed by the OGC to serve map images.	https://www.ogc.org/standard/wms/

11. Acknowledgements

The authors of this report would like to thank the following organisations and individuals for their contribution to this pilot project and to the preparation of this report.

The MEDIN coordinator, Clare Postlethwaite, for organising the core team efforts which underlay preparation and review of bids for this work, as well the review of the final outcomes, including this report. Also, the MEDIN DAC working group, the MEDIN Standards working group and the wider MEDIN community for all discussions and feedback which were prompted by this pilot project, especially to Edd Lewis of the British Geographic Survey (BGS) who undertook a parallel pilot project and has contributed heavily to technical discussions and wider efforts to communicate the results of the MEDIN pilots.

Mark Burgoyne and Chris Little of the MET office, UK who are part of the OGC standards community and have provided advice both MEDIN and the DACs who undertook the pilot projects.

The Cefas AT development team, especially David Lees who undertook the development of the OGC-EDR-API standard to Cefas datasets, as well as the IT and AT teams who provide support for the underlying infrastructure.

The Cefas project managers, Gary Saggers and Joshua Ward who supported the delivery this project.

The science developers, Kate Collingridge, and Rebecca Millard who designed and produced the demonstrator app.

The members of the Data Governance, Strategy and Support Team who have supported Cefas efforts, especially Michelle Dann and Laura Hanley who have helped with the preparation of this report.