MetOcean EDR Profile

2025-10-03

|  |
| --- |
|  |
| **The provision of Open Access to Public meteorological Data and Development of Shared federated Data Infrastructure for the Development of information Products and Services** |
| Date: 2025-10-03 |
| Version: 0.1.0 |
| Document location: TBD |
| Document status: DRAFT |
| RODEO project partners[[1]](#footnote-20) |
| Copyright © 2025 EUMETNET |

**i. Abstract**

The Open Geospatial Consortium (OGC) Environmental Data Retrieval (EDR) API provides a family of lightweight interfaces to access Environmental Data resources. Each resource addressed by an EDR API maps to a defined query pattern. OGC API - EDR identifies resources, captures compliance classes, and specifies requirements which are applicable to OGC Environmental Data Retrieval API’s. The specification addresses two fundamental operations; discovery and query. Discovery operations allow the API to be interrogated to determine its capabilities and retrieve information (metadata) about this distribution of a resource. This includes the API definition of the server as well as metadata about the Environmental Data resources provided by the server. Query operations allow Environmental Data resources to be retrieved from the underlying data store based upon simple selection criteria, defined by this standard and selected by the client.

The flexibility of EDR allows for implementation in various environmental domains and communities of practice.

This document defines an EDR profile in support of providing access to meteorological datasets in the MetOcean community of practice. This includes, but is not limited to, conventions and constraints on identifiers, metadata parameters and encodings/formats and their related definitions and extensions.

**ii. Keywords**

The following are keywords to be used by search engines and document catalogues.

metocean, ogc, api, edr, weather, meteorology, high value datasets

**iii. Security Considerations**

No additional security considerations have been made for this standard.

**iii. Submitters**

All questions regarding this submission should be directed to the editor or the submitters:

|  |  |
| --- | --- |
| Name | Affiliation |
| Håvard Futsæter (editor) | MET Norway |
| Tom Kralidis (editor) | Meteorological Service of Canada |

# Scope

This document defines the conventions, constraints and extensions of EDR in the context of the MetOcean community of practice.

The MetOcean EDR Profile defined herein is an extension of the International Standard *OGC API - Environmental Data Retrieval - Part 1: Core*.

This specification defines the conformance requirements for the MetOcean EDR Profile. Annex A defines the abstract test suite. Annex B provides normative information on schemas. Annex C provides informative examples.

# Conformance

Conformance with this standard shall be checked using the tests specified in Annex A (normative) of this document.

*OGC API - Environmental Data Retrieval* (EDR) provides a family of lightweight interfaces to access Environmental Data resources. Each resource addressed by an EDR API maps to a defined query pattern. This standard is a profile/extension of EDR. Conformance to this standard requires demonstrated conformance to the applicable Conformance Classes of *OGC API - Environmental Data Retrieval - Part 1: Core*.

Implementors of EDR API are required to comply with the MetOcean EDR Profile. A MetOcean EDR Profile shall therefore be compliant with OGC API - Environmental Data Retrieval - Part 1: Core.

This standard identifies the following Requirements Classes which define the functional requirements.

* "Core": baseline requirements class required by all other requirements classes in this document
* "Observations": TBD
* "Radar": TBD

Compliant implementation of this profile requires conformance to at least the "Core" Requirements Class.

# References

* OGC: OGC 19-086r6, OGC API - Environmental Data Retrieval Standard - Part 1: Core (2023) [[2]](#footnote-25)
* OGC: OGC 21-069r2, OGC CoverageJSON Community Standard (2023) [[3]](#footnote-28)
* IETF: RFC-7946 The GeoJSON Format (2016) [[4]](#footnote-31)
* IETF: RFC-8259 The JavaScript Object Notation (JSON) Data Interchange Format (2017) [[5]](#footnote-34)
* W3C/OGC: Spatial Data on the Web Best Practices, W3C Working Group Note (2017) [[6]](#footnote-36)
* W3C: Data on the Web Best Practices, W3C Recommendation (2017) [[7]](#footnote-38)
* IANA: Link Relation Types (2020) [[8]](#footnote-40)
* IANA: Media Types (2023) [[9]](#footnote-42)
* IETF: JSON Schema (2022) [[10]](#footnote-45)
* OpenAPI Specification 3.1.0 (2022) [[11]](#footnote-48)

# Terms and definitions

This document uses the terms defined in [OGC Policy Directive 49](https://portal.ogc.org/public_ogc/directives/directives.php), which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “SHALL” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this Standard and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications ([OGC 08-131r3](https://portal.opengeospatial.org/files/?artifact_id=34762)), also known as the 'ModSpec'. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

The following additional terms and definitions also apply.

## Abbreviated terms

Symbols and abbreviated terms

| Abbreviation | Term |
| --- | --- |
| API | Application Programming Interface |
| EU | European Union |
| EUMETNET | European National Meteorological and Hydrological Services |
| FAIR | Findable, Accessible, Interoperable, Reusable |
| FEMDI | Federated European Meteo-hydrological Data Infrastructure |
| GIS | Geographic Information System |
| HVD | High Value Datasets |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| IANA | Internet Assigned Numbers Authority |
| IETF | Internet Engineering Task Force |
| ISO | International Organization for Standardization |
| JSON | JavaScript Object Notation |
| MIME | Multipurpose Internet Mail Extensions |
| NMHS | National Meteorological and Hydrological Service |
| NWP | Numerical Weather Prediction |
| OGC | Open Geospatial Consortium |
| REST | Representational State Transfer |
| ROA | Resource-oriented architecture |
| RODEO | The provision of open access to public meteorological data and development of shared federated data infrastructure for the development of information products and services |
| S3 | Simple Storage Service |
| SEO | Search engine optimization |
| SOA | Service-oriented architecture |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| W3C | World Wide Web Consortium |
| XML | eXtensible Markup Language |

# Conventions

This section provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of JSON and / or JSON schema, or special notes regarding how to read the document.

## Identifiers

The normative provisions in this Standard are denoted by the URI:

<https://eumetnet.github.io/spec/metocean-edr-profile/1>

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

## Examples

Examples provided in this specification are encoded as JSON, GeoJSON or CoverageJSON.

## Schema representation

JSON Schema [[12]](#footnote-58) objects are used throughout this standard to define the structure of metadata records. These schema objects are also typically represented using YAML [[13]](#footnote-59). YAML is a superset of JSON, and in this standard are regarded as equivalent.

## Use of HTTPS

For simplicity, this document only refers to the HTTP protocol. This is not meant to exclude the use of HTTPS and simply is a shorthand notation for "HTTP or HTTPS." In fact, most servers are expected to use HTTPS, not HTTP.

# Introduction

RODEO, the Provision of Open Access to Public Meteorological Data and Development of Shared Federated Data Infrastructure for the Development of Information Products and Services, is a joint effort by eleven European National Meteorological and Hydrological Services (NMHS), the European Centre for Medium-Range Weather Forecasts (ECMWF) and the network of 31 European National Meteorological and Hydrological Services (EUMETNET). The Project Partners are listed in the page Partners [[14]](#footnote-64).

The RODEO project develops a user interface and Application Programming Interfaces (API) for accessing meteorological datasets declared as High Value Datasets (HVD) by the EU Implementing Regulation (EU) 2023/138 under the EU Open Data Directive (EU) 2019/1024. The project also fosters the engagement between data providers and data users for enhancing the understanding of technical solutions being available for sharing and accessing the HVD datasets.

Surface weather observations, climate data, weather warnings, radar data and numerical weather prediction (NWP) data are defined as meteorological high value datasets. The distribution of these datasets is going to be made available under an open licence, in machine-readable formats using APIs and bulk downloads following the FAIR principles (usability, accessibility, interoperability and reusability).

The project is funded by the EU Digital Europe Program (DIGITAL) and EUMETNET.

## Project Aims & Goals

The project makes meteorological High Value Datasets easily available with an aim to bring new data to businesses, public administrations and citizens. It reinforces the European public meteorological data infrastructure and enhances the providers’ digital capacity to share data. Furthermore, the project also fosters the engagement between data providers and data users for enhancing the understanding of technical solutions to be available for sharing and accessing the datasets.

The project strengthens the capacity of the European meteorological data providers to comply with the HVD Implementing Regulation by:

* Developing a user interface
* Developing APIs for accessing weather observation data, climate data, weather radar data, warnings, and Artificial Intelligence (AI) datasets
* Developing a data catalogue for making data discoverable
* Engaging with the data owners and user communities
* Supporting the deployment of national data portals and APIs
* Making HVDs available from the project partners

## Impact on users

The increased availability of data boosts entrepreneurship and potentially results in the creation of new companies. New open datasets are an important resource for small and medium-sized enterprises to develop new digital products and services. Reuse of data opens business opportunities for several sectors ultimately leads attracting more investors. By making data much easier to use, reseach, academia, AI applications and many other application areas can utilize the data more efficiently.

Overall, better data availability leads to better warnings, forecasts, and services to the public and weather-critical industries, and contributes to the safe and efficient functioning of society with multiple benefits across the European economy, industry, and society.

# Core

## Requirements Class "Core"

|  |  |
| --- | --- |
| URI | <https://eumetnet.github.io/spec/metocean-edr-profile/1/req/core> |
| Target type | Web API |
| Dependency | OGC API - Environmental Data Retrieval Standard - Part 1: Core (2023) [[15]](#footnote-70) |
| Pre-conditions | The API conforms to OGC API - Environmental Data Retrieval - Part 1: Core, Requirements Class: Core and Requirements Class: Collections |

## OpenAPI

An OpenAPI specification provides a machine-readable description of the API interface.

|  |  |
| --- | --- |
| **Requirement 1** | **/req/core/openapi** |
| A | An API definition SHALL be described using an OpenAPI document (version 3.1 or higher). |
| B | The OpenAPI document SHALL be encoded as one of the following media types: application/openapi+json, application/vnd.oai.openapi+yaml, application/vnd.oai.openapi+json, application/vnd.oai.openapi+yaml. |
| C | The OpenAPI document SHALL be made available in the API Landing Page as a link object with a relation type of service-desc and one of the types given in B. |
| D | API documentation SHALL be made available in the API Landing Page as a link object with a relation type of service-doc and content type of text/html. |
| E | The media type MAY contain a suffix indicating the OpenAPI version used, e.g. ;version=3.1. |

## Collection identifier

A collection identifier provides a mechanism to uniquely identify a given collection in an OGC API.

|  |  |
| --- | --- |
| **Requirement 2** | **/req/core/collection\_identifier** |
| A | A collection identifier SHALL NOT be used to convey structured metadata. |
| B | A collection identifier SHOULD use the following list of values, each suitable for a specific data type: insitu-observations, climate\_data, radar\_observations, weather\_warnings, weather\_forecast. |
| C | The identifier string MAY if needed include a postfix to the values listed above. |

## Collection title

A collection title provides a human readable short description of the given collection.

|  |  |
| --- | --- |
| **Requirement 3** | **/req/core/collection\_title** |
| A | A title SHALL be set for all collections. |
| B | A title SHALL NOT have more than 50 characters. |
| C | A title SHOULD be written in English. |
| D | A title SHOULD have the most important information first, guarding against truncated presentation of the value. |
| E | A title SHOULD describe the collection in a way understandable also for non-experts. Usually mention both topic/domain and geographical area. |

## Collection license

A license describes the usage permissions for the data in a collection.

|  |  |
| --- | --- |
| **Requirement 4** | **/req/core/collection\_license** |
| A | The links property in a collection SHALL contain a link to a license of its data. |
| B | The license link SHALL set the rel property to license. |
| C | The type property SHALL be set to text/html. |
| D | If the data is open data, the license SHALL be CC BY 4.0 and the href property set to https://creativecommons.org/licenses/by/4.0/. |

## Collection temporal extent

Temporal extent of a collction.

|  |  |
| --- | --- |
| **Requirement 5** | **/req/core/collection\_temporal\_extent** |
| A | The value of extent.temporal.trs SHALL be Gregorian. |

## Collection spatial extent

Spatial extent of a collction.

|  |  |
| --- | --- |
| **Requirement 6** | **/req/core/collection\_spatial\_extent** |
| A | The value of extent.spatial.crs SHALL be OGC:CRS84. |
| B | The value of crs SHALL include the element OGC:CRS84. |
| C | If crs\_details is specified for a data\_queries object, the crs\_details array SHALL include an object whose crs value is OGC:CRS84. |
| D | If a collection supports more CRSs than OGC:CRS84, these values SHOULD be presented on the syntax EPSG:<code> in crs and data\_queries.\*.crs\_details.crs if an epsg code is available. |

## Data query response metadata

Metadata in the response to a data query.

|  |  |
| --- | --- |
| **Recommendation 1** | **/rec/core/data\_query\_response\_metadata** |
| A | The metadata in the response to a data query, e.g crs, parameters and units, SHOULD mirror as closely as possible the terms and vocabularies used in the collection metadata. |

## Collection vertical extent

Vertical extent of a collection.

|  |  |
| --- | --- |
| **Recommendation 2** | **/rec/core/collection\_vertical\_extent** |
| A | If extent.vertical is defined, the value of extent.vertical.vrs SHOULD be one the following values: Pressure level in hPa, Geopotential height in gpm, Geometrical altitude above mean sea level in meters, Height above ground in meters, Flight level. |

## Collection parameter names

Parameter names is an object, containing parameter objects with metadata about all parameters the collection contains.

|  |  |
| --- | --- |
| **Requirement 7** | **/req/core/collection\_parameter\_names** |
| A | The keys in parameter\_names and id in the parameter object SHALL NOT be used to convey structured metadata. |
| B | A parameter object SHALL specify the following fields: label, description, unit |
| C | label SHALL NOT have more than 50 characters. |
| D | label SHALL be written in English. |
| E | unit SHALL use QUDT as vocabulary, and unit.symbol.type SHALL be on the form https://qudt.org/vocab/unit/<unit> and unit.symbol.value SHALL be set to the value of qudt:symbol for that unit. |
| F | observedProperty SHALL use the CF-convention if a suitable value is available and set observedProperty.id to value https://vocab.nerc.ac.uk/standard\_name/<standard\_name> If not, use observedProperty.description to describe it. |

## Collection radius data query

Metadata about the radius data query in a collection.

|  |  |
| --- | --- |
| **Requirement 8** | **/req/core/collection\_radius\_data\_query** |
| A | The within\_units array in a variables object for a radius data query SHALL at least contain m, interpreted as the SI unit of length. |
| B | within\_units can contain more values, but all values in the array SHALL be defined as a unit <https://qudt.org>, and SHALL use the symbol of the unit. |
| C | When performing a radius query which includes the limit query parameter, this SHOULD be interpreted as requesting the limited amount of data items that are closest to the specified center of the radius. |

## Locations query response format

Structure of the response document for a /locations query.

|  |  |
| --- | --- |
| **Requirement 9** | **/req/core/locations\_query\_response\_format** |
| A | The response document for a /locations query to a collection or an instance of a collection SHALL be a GeoJSON document with a FeatureCollection object. |
| B | Each Feature object in the FeatureCollection SHALL have the following properties: id of type string, properties.name of type string. |
| C | If the data for a location do not provide all parameters listed in the collection, it SHALL include properties.parameter-name which SHALL be a list of strings. This list SHALL be a subset of the list of keys from the parameter\_names object in the collection. |
| D | If there are links with more information about the location they SHALL be included in properties.links as a list of link objects. The link object SHALL have identical structure as link objects defined by OGC API - Common Part 1: Core. |
| E | A more detailed description of the location MAY also be provided via properties.description. |
| F | All strings SHALL be in english. |

## Error handling

HTTP status codes and response documents for responding to errors.

|  |  |
| --- | --- |
| **Requirement 10** | **/req/core/error\_handling** |
| A | A query for data inside the extent of a collection that results in no data SHALL have a HTTP status code of 204, and return no content. |
| B | A data query that is partially within the extent SHALL respond as if it is within the extent, but SHOULD only respond with data for the part of the geographical area in the query that is inside the extent. |
| C | Any http request that results in a response with a 4xx HTTP status code SHALL give a response which complies with [RFC 9457 Problem Details for HTTP API](https://www.rfc-editor.org/rfc/rfc9457.html). |

# Insitu observations

## Requirements Class "Insitu observations"

|  |  |
| --- | --- |
| URI | <https://eumetnet.github.io/spec/metocean-edr-profile/1/req/insitu-observations> |
| Target type | Web API |
| Dependency | [Core](#rc_core) |

## Collection data queries

The types of data queries implemented in a collection.

|  |  |
| --- | --- |
| **Requirement 11** | **/req/insitu-observations/collection\_data\_queries** |
| A | A collection SHALL support locations, area and radius. |

## Collection parameter names

Parameter names is an object, containing parameter objects with metadata about all parameters the collection contains.

|  |  |
| --- | --- |
| **Requirement 12** | **/req/insitu-observations/collection\_parameter\_names** |
| A | Each object in parameter\_names SHALL specify a property metocean:standard\_name with value set to one of the values listed in values array for custom dimension with id with value standard\_names. |
| B | Each object in parameter\_names SHALL specify a property metocean:level with value set to one of the values listed in values array for custom dimension with id with value levels. |
| C | Each object in parameter\_names SHALL specify measurementType and the measurementType.method SHOULD follow the CF-convention cell methods <https://cfconventions.org/Data/cf-conventions/cf-conventions-1.7/build/ch07s03.html> and use a value from the list in <https://cfconventions.org/Data/cf-conventions/cf-conventions-1.7/build/ape.html>. |

## Collection custom dimensions

The custom dimensions for which you can query the collection.

|  |  |
| --- | --- |
| **Requirement 13** | **/req/insitu-observations/collection\_custom\_dimensions** |
| A | A collection can contain multiple parameters who fit under a standard name as defined by the CF metadata conventions. A collection SHALL have an object in extent.custom with id of value standard\_name. |
| B | Custom dimension object standard\_name SHALL specify a list of CF-conventions standard names in values. |
| C | Custom dimension standard\_name SHALL specify reference with value https://vocab.nerc.ac.uk/standard\_name/. |
| D | A collection can contain parameters that are identical except for having performed measurements on different heights. A collection SHALL have an object in extent.custom with id of value level. |
| E | Custom dimension level SHALL specify reference with value Height of measurement above ground level in meters. |

## Data query response format

The response format for data queries.

|  |  |
| --- | --- |
| **Requirement 14** | **/req/insitu-observations/data\_query\_response\_format** |
| A | All data queries SHALL support at least CoverageJSON as response format. |

## CoverageJSON parameters

The metadata about parameters in CoverageJSON.

|  |  |
| --- | --- |
| **Requirement 15** | **/req/insitu-observations/coveragejson\_parameters** |
| A | Each object in parameters in a coverage object SHALL have a property metocean:measurementType. This object SHALL have the same properties as measurementType in an object in parameter\_names in a collection. |
| B | Each object in parameters SHALL have a property metocean:standard\_name. |
| C | Each object in parameters SHALL have a property metocean:level. |

## CoverageJSON referencing

Coordinate referencing metadata in CoverageJSON.

|  |  |
| --- | --- |
| **Requirement 16** | **/req/insitu-observations/coveragejson\_referencing** |
| A | When requesting data and crs query param is not specified the system property for the spatial coordinates in domain.referencing.\* SHALL have the following values: "type": "GeographicCRS" and "id": "OGC:CRS84". |
| B | When requesting data and the crs query param is specified, the domain.referencing.\*.system object with "type": "GeographicCRS" SHALL have its id property set to the same value as the crs query param. |

# Conformance Class Abstract Test Suite (Normative)

## Conformance Class: Core

label

<https://eumetnet.github.io/spec/metocean-edr-profile/1/conf/core>

subject

Requirements Class "core"

classification

Target Type:Web API

### OpenAPI

label

/conf/core/openapi

subject

/req/core/openapi

test-purpose

Validate that a MetOcean EDR Profile provides an API definition using an OpenAPI document

Construct a path for a landing page.

Issue an HTTP GET request on that path.

Check that the value of the returned HTTP status header is 200.

In the links array in the landing page response, find the object with rel set to service-desc and type matching one of the following strings:

* application/openapi+json
* application/openapi+yaml
* application/vnd.oai.openapi+json
* application/vnd.oai.openapi+yaml

Optionally, the string may be suffixed with a version number like ;version=3.1. In that case, check that the version is >= 3.1.

Issue an HTTP GET request using the value of href for that object.

Check that the value of the returned HTTP status header is 200 and check that the response body is a valid OpenAPI document of version 3.1 or higher.

In the links array in the landing page response, find the object with rel set to service-doc.

Issue an HTTP GET request using the value of href for that object.

Check that the value of the returned HTTP status header is 200 and that the HTTP response header Content-Type contains text/html.

### Collection identifier

label

/conf/core/collection\_identifier

subject

/req/core/collection\_identifier

test-purpose

Validate that a MetOcean EDR Profile API provides a valid collection identifier

This requirement is not applicable to ATS testing.

### Collection title

label

/conf/core/collection\_title

subject

/req/core/collection\_title

test-purpose

Validate that a MetOcean EDR Profile provides a valid collection title

Issue an HTTP GET request to path /collections.

Check that the value of the returned HTTP status header is 200.

In the collections array, check that the title property is present for all array elements.

For each title value, check that the value is less than or equal to 50 characters.

### Collection license

label

/conf/core/collection\_license

subject

/req/core/collection\_license

test-purpose

Validate that a MetOcean EDR Profile API provides a collection links array with a license link.

Issue an HTTP GET request to path /collections.

Check that the value of the returned HTTP status header is 200.

In the links array, check that there exists one link object with rel set to license.

Issue a HTTP GET request to the url in href for link with rel license.

Check that the value of the returned HTTP status header is 200 and the HTTP response header Content-Type is text/html.

### Collection spatial extent

label

/conf/core/collection\_spatial\_extent

subject

/req/core/collection\_spatial\_extent

test-purpose

Validate that a MetOcean EDR Profile API provides spatial extent in the correct crs.

Issue an HTTP GET request to path /collections.

Check that the value of the returned HTTP status header is 200.

In the response document, for each object in the collections array, check that the extent.spatial.crs property is set to OGC:CRS84.

In the collections array, check that the crs property is set to OGC:CRS84 for each object.

For each object in collections array, for each object in the data\_queries array, if crs\_details is specified, check that one of the objects in the crs\_details array has a crs property set to OGC:CRS84.

### Collection temporal extent

label

/conf/core/collection\_temporal\_extent

subject

/req/core/collection\_temporal\_extent

test-purpose

Validate that a MetOcean EDR Profile API provides temporal extent in the correct trs.

Issue an HTTP GET request to path /collections.

Check that the value of the returned HTTP status header is 200.

In the response document, for each object in the collections array, check that the extent.temporal.trs property is set to Gregorian.

### CoverageJSON referencing

label

/conf/core/coveragejson\_referencing

subject

/req/core/coveragejson\_referencing

test-purpose

Validate that a MetOcean EDR Profile API CoverageJSON response document provides correct domain referencing values.

Issue an HTTP GET request to path /collections.

Check that the value of the returned HTTP status header is 200.

For each object in the collections array, for each object in data\_queries, issue a GET request with no crs query parameter specified.

Check that the response is a CoverageJSON document where each object of type Coverage contains a domain.referencing array which includes an object where system.type is set to GeographicCRS and system.id set to OGC:CRS84.

For each object in the collections array, for each object in data\_queries, issue a GET request for each crs\_details object where crs query parameter set to the value in the crs\_details object.

Check that the response is a CoverageJSON object where domain.referencing includes an object where system.type is set to GeographicCRS and system.id is equal to the crs query parameter.

### Collection parameter names

label

/conf/core/collection\_parameter\_names

subject

/req/core/collection\_parameter\_names

test-purpose

Validate that a MetOcean EDR Profile API provides a valid collection parameter\_names

Issue an HTTP GET request to path /collections.

Check that the value of the returned HTTP status header is 200.

In the response document, for each object in the collections array, check that each object in the parameter\_names array has the properties label, unit and description.

The value of a label property in a parameter object is less than or equal to 50 characters.

The value of unit.symbol.type in a parameter object is a url on the form https://qudt.org/vocab/unit/{unit}.

The value of unit.symbol.value in a parameter object is specified.

### Collection radius data query

label

/conf/core/collection\_radius\_data\_query

subject

/req/core/collection\_radius\_data\_query

test-purpose

Validate that a MetOcean EDR Profile API with a radius data query in a collection has correct metadata.

Issue an HTTP GET request to path /collections.

Check that the value of the returned HTTP status header is 200.

In the response document, for each object in the collections array, if the data\_queries property contains radius, check that the link.variables object for radius has a within\_units array that contains at least m.

### Locations query response format

label

/conf/core/locations\_query\_response\_format

subject

/req/core/locations\_query\_response\_format

test-purpose

Validate that a MetOcean EDR Profile API has a correctly formated response to a locations query.

For every collection and instance of a collection, check if locations exists as a key in the data\_queries object. If so, get the href value from the link object with "rel":"self" and append /locations to the value and issue an HTTP GET request.

Check that the value of the returned HTTP status header is 200.

Check that the response http header Content-Type has value application/geo+json.

In the response document, check that it has the property "type": "FeatureCollection". For each object in the features array, check that the properties id and properties.name are present.

### Error handling

label

/conf/core/error\_handling

subject

/req/core/error\_handling

test-purpose

Validate that a MetOcean EDR Profile API handles errors correctly.

Issue an HTTP GET request to an non-existent path /collections/nonexistentcollection.

Check that the value of the returned HTTP status code is 404 and the HTTP response header Content-Type is application/problem+json.

Check that the returned JSON document includes the properties type and title. Check that title is a string and that type is either an URI or the value about:blank.

## Conformance Class: Insitu observations

label

<https://eumetnet.github.io/spec/metocean-edr-profile/1/conf/insitu-observations>

subject

Requirements Class "Insitu observations"

classification

Target Type:Web API

### Collection data queries

label

/conf/insitu-observations/collection\_data\_queries

subject

/req/insitu-observations/collection\_data\_queries

test-purpose

Validate that a MetOcean EDR Insitu observations Profile API has implemented the mandatory data queries.

Issue an HTTP GET request to path /collections.

Check that the value of the returned HTTP status header is 200.

In the collections array in the returned document, check that each collection has a data\_queries array containing at least area, locations and radius.

### Collection parameter names

label

/conf/insitu-observations/collection\_parameter\_names

subject

/req/insitu-observations/collection\_parameter\_names

test-purpose

Validate that a MetOcean EDR Insitu observations Profile API has the required properties for parameter\_names

Issue an HTTP GET request to path /collections.

Check that the value of the returned HTTP status header is 200.

For each object in the collections array and for each object in parameter\_names check that it has the properties metocean:standard\_name with the value of type string.

For each object in the collections array and for each object in parameter\_names check that it has the property metocean:level with a value of type number.

For each object in the collections array and for each object in parameter\_names check that is has the property measurementType.

### Collection custom dimensions

label

/conf/insitu-observations/collection\_custom\_dimensions

subject

/req/insitu-observations/collection\_custom\_dimensions

test-purpose

Validate that a MetOcean EDR Insitu observations Profile API has required custom dimensions.

Issue an HTTP GET request to path /collections.

Check that the value of the returned HTTP status header is 200.

Check that the extent.custom array has an object with property id set to standard\_name and property reference set to https://vocab.nerc.ac.uk/standard\_name/.

Check that the extent.custom array has an object with property id set to level and property reference set to Height of measurement above ground level in meters.

### Coveragejson parameters

label

/conf/insitu-observations/coveragejson\_parameters

subject

/req/insitu-observations/coveragejson\_parameters

test-purpose

Validate that a MetOcean EDR Insitu observations Profile API has required metadata in CoverageJSON parameters.

Issue an HTTP GET request to path /collections.

Check that the value of the returned HTTP status header is 200.

For each object in the collections array and for each object in data\_queries, issue a GET request with no crs query parameter specified.

Check that each object in parameters array of the response CoverageJSON document contains a property metocean:standard\_name with a string type.

Check that each parameter object in parameters in the response CoverageJSON document contains a property metocean:level with a number type.

Check that each parameter object in parameters in the response CoverageJSON document contains a property metocean:measurementType which has an object type. Check that this object has the property method of type string and property duration of type string.

### Data query response format

label

/conf/insitu-observations/data\_query\_response\_format

subject

/req/insitu-observations/data\_query\_response\_format

test-purpose

Validate that a MetOcean EDR Insitu observations Profile API has correct response format for data queries.

Issue an HTTP GET request to path /collections.

Check that the value of the returned HTTP status header is 200.

In the returned document, for each object in data\_queries for each collection in collections make a corresponding data query request and check that the returned response has status 200 and has a http response header Content-Type with value application/vnd.cov+json.

# Schemas (Normative)

# Examples (Informative)

## Examples

# Bibliography

* W3C/OGC: Spatial Data on the Web Best Practices, W3C Working Group Note 28 September 2017, <https://www.w3.org/TR/sdw-bp>
* W3C: Data on the Web Best Practices, W3C Recommendation 31 January 2017, <https://www.w3.org/TR/dwbp>
* W3C: Data Catalog Vocabulary, W3C Recommendation 16 January 2014, <https://www.w3.org/TR/vocab-dcat>
* IANA: Link Relation Types, <https://www.iana.org/assignments/link-relations/link-relations.xml>
* Linux Foundation: SPDX License List, <https://spdx.org/licenses>

# Revision History

| Date | Release | Editor | Primary clauses modified | Description |
| --- | --- | --- | --- | --- |
| 2024-09-16 | Template | Tom Kralidis | all | initial template |

1. <https://rodeo-project.eu/partners/#project-partners> [↑](#footnote-ref-20)
2. <https://docs.ogc.org/is/19-086r6/19-086r6.html> [↑](#footnote-ref-25)
3. <https://docs.ogc.org/cs/21-069r2/21-069r2.html> [↑](#footnote-ref-28)
4. <https://datatracker.ietf.org/doc/html/rfc7946> [↑](#footnote-ref-31)
5. <https://datatracker.ietf.org/doc/html/rfc8259> [↑](#footnote-ref-34)
6. <https://www.w3.org/TR/sdw-bp> [↑](#footnote-ref-36)
7. <https://www.w3.org/TR/dwbp> [↑](#footnote-ref-38)
8. <https://www.iana.org/assignments/link-relations/link-relations.xml> [↑](#footnote-ref-40)
9. <https://www.iana.org/assignments/media-types/media-types.xhtml> [↑](#footnote-ref-42)
10. <https://json-schema.org> [↑](#footnote-ref-45)
11. <https://github.com/OAI/OpenAPI-Specification/blob/3.1.0/versions/3.1.0.md> [↑](#footnote-ref-48)
12. <https://json-schema.org> [↑](#footnote-ref-58)
13. <https://en.wikipedia.org/wiki/YAML> [↑](#footnote-ref-59)
14. <https://rodeo-project.eu/partners> [↑](#footnote-ref-64)
15. <https://docs.ogc.org/is/19-086r6/19-086r6.html> [↑](#footnote-ref-70)