OGC® DOCUMENT: 19-086R4

External identifier of this OGC® document: http://www.opengis.net/doc/IS/ogcapi-edr-1/1.0

OGC®
Making location count.

OGC API - ENVIRONMENTAL DATA RETRIEVAL STANDARD

STANDARD

APPROVED

Version: 1.0.0
Submission Date: 2020-12-11
Approval Date: 2021-03-30
Publication Date: 2021-08-13
Editor: Mark Burgoyne, Dave Blodgett, Chuck Heazel, Chris Little

License Agreement

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

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

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

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

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Suggested additions, changes and comments on this document are welcome and encouraged. Such suggestions may be submitted using the online change request form on OGC web site: http://portal.opengeospatial.org/public_ogc/change_request.php

Copyright notice

Note

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF RECOMMENDATIONS

ABSTRACT

The Environmental Data Retrieval (EDR) Application Programming Interface (API) provides a family of lightweight query interfaces to access spatio-temporal data resources by requesting data at a **Position**, within an **Area**, along a **Trajectory** or through a **Corridor**. A spatio-temporal data resource is a collection of spatio-temporal data that can be sampled using the EDR query pattern geometries. These patterns are described in the section describing the Core Requirements Class.

The goals of the EDR API are to make it easier to access a wide range of data through a uniform, well-defined simple Web interface, and to achieve data reduction to just the data needed by the user or client while hiding much of the data storage complexity. A major use case for the EDR API is to retrieve small subsets from large collections of environmental data, such as weather forecasts, though many other types of data can be accessed. The important aspect is that the data can be unambiguously specified by spatio-temporal coordinates.

The EDR API query patterns, such as Position, Area, Cube, Trajectory or Corridor, can be thought of as discrete sampling geometries, conceptually consistent with the feature of interest in the Sensor Observation Service (SOS) standard. A typical EDR data resource is a multidimensional dataset that could be accessed via an implementation of the Web Coverage Service (WCS) standard. In contrast to SOS and WCS, EDR implements the technical baseline of the OGC API family of standards and aims to provide a single set of simple-to-use query patterns. Use cases for EDR range from real or virtual time-series observation retrievals, to sub-setting 4-dimensional data cubes along user-supplied sampling geometries. These query patterns do not attempt to satisfy the full scope of either SOS or WCS, but provide useful building blocks to allow the composition of APIs that satisfy a wide range of geospatial data use cases. By defining a small set of query patterns (and no requirement to implement all of them), the EDR API should help to simplify the design of systems (as they can be performance tuned for the supported queries) making it easier to build robust and scalable infrastructure.

With the OGC API family of standards, the OGC community has extended its suite of standards to include Resource Oriented Architectures and Web Application Programming Interfaces (APIs). These standards are based on a shared foundation, specified in OGC API-Common, which defines the resources and access paths that are supported by all OGC APIs. The resources are listed in Table 1. This document extends that foundation to define the Environmental Data Retrieval API.

Table 1 — Overview of Resources

RESOURCE	PATH	HTTP METHOD	DOCUMENT REFERENCE
Landing page	/	GET	API Landing Page
API definition	/api	GET	API Definition
Conformance classes	/conformance	GET	Declaration of Conformance Classes

RESOURCE	PATH	HTTP METHOD	DOCUMENT REFERENCE
Collections metadata	`/collections`	GET	Collections Metadata
Collection instance metadata	`/collections/{collection_id}`	GET	Collection Metadata

The resources identified in Table 1 primarily support Discovery operations. Discovery operations allow clients to interrogate the API to determine its capabilities and obtain information (metadata) about a distribution of a resource. This includes the API definition of the server(s) as well as metadata about the resources provided by those servers.

This standard extends the Table 1 common query operations by defining simple, coordinate-based, queries which are applicable to many spatio-temporal, including geospatial, resource types. Other OGC API standards may define additional query capabilities specific to their resource type. EDR Query operations allow resources or values to be retrieved from the underlying spatio-temporal resource data store. The information returned is based upon the selection criteria (query string) provided by the client.

II KEYWORDS

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

ogcdoc, OGC document, property, geographic information, spatial data, spatial thing, spatio-temporal, dataset, distribution, API, JSON, geoJSON, coverageJSON, HTML, OpenAPI, AsyncAPI, REST, Common, position, area, trajectory, corridor, cube, time-series, radius, polygon, WKT

III PREFACE

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

IV SECURITY CONSIDERATIONS

No security considerations have been made for this document.

V SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- UK Met Office
- US Geological Service (USGS)
- US National Weather Service
- Wuhan University
- Meteorological Service of Canada
- Finnish Meteorological Institute
- Esri
- National Aeronautics and Space Administration (NASA)
- Météo-France

VI SUBMITTERS

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

Name	Affiliation
Mark Burgoyne *(editor)*	Met Office
Chris Little *(editor)*	Met Office
Chuck Heazel *(editor)*	HeazelTech LLC
Dave Blodgett *(editor)*	USGS
Tom Kralidis *(contributor)*	Meteorological Service of Canada

1 SCOPE

1 SCOPE

This specification identifies resources, captures compliance classes, and specifies requirements which are applicable to OGC Environmental Data Retrieval APIs.

This 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 a distribution of a resource. This includes the API definition of the server as well as metadata about the spatio-temporal data resources provided by the server.

A spatio-temporal data resource is a **collection** of spatio-temporal data that can be sampled using OGC-API Environmental Data Retrieval query patterns.

Query operations allow other data resources, such as environmental, to be sampled from the underlying spatio-temporal data resource, or data store, based upon EDR query geometry and other selection criteria, defined by this standard and selected by the client.

2

CONFORMANCE

2 CONFORMANCE

Conformance with this standard shall be checked using the tests specified in Annex B of this document. The framework, concepts, and methodology for testing, and the criteria to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site.

The one Standardization Target for this standard is Web APIs.

OGC API — Common — Part 1: Core defines an API module intended for re-use by other OGC Web API standards. This OGC API — EDR standard is an extension of OGC API — Common — Part 1: Core and OGC API — Common — Part 2: Geospatial Data. Conformance to the OGC API — EDR standard requires demonstrated conformance to the applicable Conformance Classes of OGC API — Common.

This OGC API — EDR standard identifies a set of Conformance Classes. The Conformance Classes implemented by an API are advertised through the `/conformance` path on the landing page. Each Conformance Class is defined by one or more Requirements Classes. The requirements in Annex A are organized by Requirements Class. The Requirements Classes therefore define the functional requirements which are tested through the associated Conformance Class.

2.1. Mandatory Requirements Classes

The mandatory requirements classes of OGC API-EDR include:

- Requirements Class “OGC API — Environmental Data Retrieval Core”: This requirements class inherits from the *Core Requirements Class* of OGC API — Common — Part 1: Core which specifies the minimal useful service interface for an OGC API. The requirements specified in the Requirements Class “OGC API — Environmental Data Retrieval Core” are mandatory for all implementations of the EDR API. The requirements are specified in Chapter 7 and in Annex A.2 in more detail.
- Requirements Class “Collections”: This requirements class inherits from the *Collections Requirements Class* of OGC API — Common — Part 2: Geospatial Data which extends the *Core Requirements class* of OGC API — Common — Part 1: Core to enable discovery and query access to collections of spatial resources.

The structure and organization of a collection of spatio-temporal data is very much dependent on the nature of that data and the expected access patterns. This is information which cannot be specified in a common manner. The OGC API — Common — Part 2: Geospatial Data Standard, specifies the requirements necessary to discover and understand a generic collection of spatio-temporal data.

The *Collections Requirements Class* of the EDR API extends the common requirements to those specific to the query and retrieval of collections of spatio-temporal data. The Requirements Class is specified in Chapter 8 and specified in more detail in Annex A.3.

2.2. Optional Requirements Classes

Neither the *Core* nor *Collections* requirements classes mandate specific encodings or formats for representing resources. The optional *HTML*, *GeoJSON* and *CoverageJSON* requirements classes specify representations for these resources in frequently used encodings for spatial data on the web.

- The **JSON** and **EDR GeoJSON** conformance classes ensure that basic discovery of Core and Collection resources for the EDR API can be performed. They have one Requirements Class each
- Encodings, three Requirements Classes
  - HTML
  - GeoJSON
  - CoverageJSON

The Requirements Classes are specified in Chapter 9 and specified in more detail in Annex A.6, A.8, and A.9.

None of these encodings are mandatory. An implementation of the EDR API may decide to implement another encoding instead of, or in addition to, those listed. However, a common format does simplify interoperability so support for *CoverageJSON* is highly recommended as an established, efficient and effective format for a variety of spatio-temporal data.

- OpenAPI 3.0, one Requirements Class

The OGC API — Common — Part 1: Core specification does not mandate any encoding or format for the formal definition of the API. However, the preferred option is the OpenAPI 3.0 specification. Therefore the EDR APIs will be defined using OpenAPI 3.0.

The OpenAPI 3.0 Requirements Class is specified in Chapter 9 and Annex A in more detail.

- Queries, one Requirements Class
    - Position
    - Radius
    - Area
    - Cube
    - Trajectory
    - Corridor
    - Items
    - Locations
    - Instances

The EDR API Queries Conformance class does not mandate any specific query patterns for querying resources. The requirements class specifies query patterns for which there are ubiquitous use cases.

An implementation of the EDR API may decide to implement another query pattern instead of, or in addition to, those listed. However, a minimal query pattern of retrieving data at a position (with elevation and time) does simplify interoperability so support for the position query is highly recommended.

At least one of the 7 queries: Position, Radius, Area, Cube, Trajectory, Corridor, or Location shall be implemented.

The Queries Requirements Class is specified in Chapter 9 and specified in detail in Annex A.4.

3

NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Open API Initiative: OpenAPI Specification 3.0.3, https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: IETF RFC 2616, HTTP/1.1, https://tools.ietf.org/rfc/rfc2616.txt

Rescorla, E.: IETF RFC 2818, HTTP Over TLS, https://tools.ietf.org/rfc/rfc2818.txt

Klyne, G., Newman, C.: IETF RFC 3339, Date and Time on the Internet: Timestamps, https://tools.ietf.org/rfc/rfc3339.txt

Berners-Lee, T., Fielding, R., Masinter, L: IETF RFC 3896, Uniform Resource Identifier (URI): Generic Syntax, https://tools.ietf.org/rfc/rfc3896.txt

Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S., Schaub, T.: IETF RFC 7946, The GeoJSON Format, https://tools.ietf.org/rfc/rfc7946.txt

Nottingham, M.: IETF RFC 8288, Web Linking, https://tools.ietf.org/rfc/rfc8288.txt

Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., Orchard, D.: IETF RFC 6570, URI Template, https://datatracker.ietf.org/doc/html/rfc6570

W3C: HTML5, W3C Recommendation, https://www.w3.org/TR/html5/

Schema.org: https://schema.org/docs/schemas.html

Blower, J., Riechert, M., Roberts, B.: Overview of the CoverageJSON format, https://www.w3.org/TR/covjson-overview/

Weibel, S., Kunze, J., Lagoze, C., Wolf, M.: IETF RFC 2413, Dublin Core Metadata for Resource Discovery, https://tools.ietf.org/rfc/rfc2413.txt

John Herring: OGC 06-103r4, *OpenGIS Implementation Specification for Geographic information — Simple feature access — Part 1: Common architecture*. Open Geospatial Consortium (2011). https://portal.ogc.org/files/?artifact_id=25355

Roger Lot: OGC 18-010r7, *Geographic information — Well-known text representation of coordinate reference systems*. Open Geospatial Consortium (2019). http://docs.opengeospatial.org/is/18-010r7/18-010r7.html

Clemens Portele, Panagiotis (Peter) A. Vretanos, Charles Heazel: OGC 17-069r3, *OGC API — Features — Part 1: Core*. Open Geospatial Consortium (2019). http://docs.opengeospatial.org/is/17-069r3/17-069r3.html

Heazel, C.: OGC API — Common — Part 1: Core (Draft). OGC 19-072, Open Geospatial Consortium, http://docs.ogc.org/DRAFTS/19-072.html

Heazel, C.: OGC API — Common — Part 2: Geospatial Data (Draft). OGC 20-024, Open Geospatial Consortium, http://docs.ogc.org/DRAFTS/20-024.html

4

TERMS AND DEFINITIONS

4 TERMS AND DEFINITIONS

This document uses the terms defined in OGC Policy Directive 49, 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 document 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), also known as the ‘ModSpec’. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

For the purposes of this document, the following additional terms and definitions apply.

The Glossary includes terms from other standards and specifications that, while not normative, are critical to accurately understand this specification.

4.1. **area**

region specified with a geographic envelope that may have vertical dimension

4.2. **corridor**

two parameter set of points around a trajectory

4.3. **cube**

rectangular area, with a vertical extent

4.4. **location**

identifiable geographic place

(Source: ISO 19112:2019)

Note 1 to entry: A location is represented by one of a set of data types that describe a position, along with metadata about that data, including coordinates (from a coordinate reference system), a measure (from a linear referencing system), or an address (from an address system).

4.5. instance

version, release, or run of a given data collection

4.6. position

data type that describes a point or geometry potentially occupied by an object or person

(Source: ISO 19133:2005)

4.7. radius

region specified with a geographic position and radial distance

4.8. spatio-temporal data

data associated with a position in space-time

4.9. trajectory

path of a moving point described by a one parameter set of points

(Source: ISO 19141:2008)

5

CONVENTIONS

5 CONVENTIONS

5.1. Identifiers

The Architecture of the World Wide Web establishes the Uniform Resource Identifier (URI) as the single global identification system for the Web. Therefore, URIs or URI Templates are used in OGC Web API standards to identify key entities in those standards.

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

http://www.opengis.net/spec/ogcapi-edr-1/1.0

All Requirements and Conformance Tests that appear in this document are denoted by partial URIs which are relative to this base.

A key requirement of Web API standards is the unambiguous identification of the resources they address. In an implementation of such a standard, URIs would be used to identify those resources. A standard, however, is not an implementation. A standard can identify potential resources, but not the resources themselves. Therefore, OGC Web API standards use URI Templates to identify resource categories. These resource categories are instantiated in the implementation of the standard.

The scope of each URI Template is specified in the standard. In some cases, API implementations are required to implement the template as a path in their API. In most cases they are optional.

Implementation of the URI Templates is recommended in that they provide a common look and feel to implementations of OGC Web API standards.

5.2. Link relations

To express relationships between resources, RFC 8288 (Web Linking) and registered link relation types are used wherever possible and denoted below with [IANA]. Additional link relation types are registered with the OGC Link Relation Registry. These are denoted below with [OGC].

The following link-relations are in common use by OGC Web API Standards.

- **alternate**: Refers to a substitute for this context. [IANA]
- **collection**: The target IRI points to a resource which represents the collection resource for the context IRI. [IANA]

- **conformance**: Refers to a resource that identifies the specifications that the link's context conforms to. [OGC]
- **data**: refers to the root resource of a dataset in an API. [OGC]
- **describedby**: Refers to a resource providing information about the link's context. [IANA]
- **item**: The target IRI points to a resource that is a member of the collection represented by the context IRI. [IANA]
- **items**: Refers to a resource that comprises members of the collection represented by the link's context. [OGC]
- **license**: Refers to a license associated with this context. [IANA]
- **self**: Conveys an identifier for the link's context. [IANA]
- **service-desc**: Identifies service description for the context that is primarily intended for consumption by machines. [IANA]
- **service-doc**: Identifies service documentation for the context that is primarily intended for human consumption. [IANA]

NOTE 1: API definitions are considered service descriptions.

Each resource representation includes an array of links. Implementations are free to add additional links for all resources provided by the API. For example, an **enclosure** link could reference a bulk download of a collection. Or a **related** link on a feature could reference a related feature.

A **license** link could be used for constraints on the data retrieved. Multiple **license** links could be provided for different content types. Alternatively, if all data retrieved via the API is available under the same license, the link MAY instead be added to the top-level links property of the response.

NOTE 2: The query patterns of the EDR API use the link relation **data**. It is envisaged that, in the future, this link relation may be replaced by **position**, **area**, and **trajectory** which would all be specializations of the currently used **data**.

5.3. Media Types

JSON media types that would typically be used in an OGC API that supports JSON are:

- `application/prs.coverage+json` for resources that include coverage content encoded according to CoverageJSON
- `application/geo+json` for feature collections and features

- `application/json` for all other resource representations, as well as coverage content encoded according to the Coverage Implementation Schema (CIS)

XML media types that would typically occur in an OGC API that supports XML are:

- `application/gml+xml;version=3.2` for any Geography Markup Language (GML) 3.2 feature collections and features
- `application/gml+xml;version=3.2;profile=http://www.opengis.net/def/profile/ogc/2.0/gml-sf0` for GML 3.2 feature collections and features conforming to the GML Simple Feature Level 0 profile
- `application/gml+xml;version=3.2;profile=http://www.opengis.net/def/profile/ogc/2.0/gml-sf2` for GML 3.2 feature collections and features conforming to the GML Simple Feature Level 2 profile
- `application/xml` for all other resources

The typical HTML media type for all "web pages" in an OGC API would be `text/html`.

The media types for an OpenAPI definition are `vnd.oai.openapi+json;version=3.0` (JSON) and `application/vnd.oai.openapi;version=3.0` (YAML).

NOTE 1: The OpenAPI media type has not been registered yet with IANA and may change.

NOTE 2: The CoverageJSON media type has not been registered yet with IANA and may change.

5.4. Examples

Most of the examples provided in this standard are encoded in JSON. JSON was chosen because it is widely understood by implementors and easy to include in a text document. This convention should NOT be interpreted as a requirement that JSON must be used. Implementors are free to use any format they desire as long as there is a Conformance Class for that format and the API advertises its support for that Conformance Class.

5.5. Schema

JSON Schema is used throughout this standard to define the structure of resources. These schema are typically represented using YAML encoding. This convention is for the ease of the user. It does not prohibit the use of another schema language or encoding. Nor does it indicate that JSON schema is required. Implementations should use a schema language and encoding appropriate for the format of the resource.

5.6. Use of HTTPS

For simplicity, this document generally 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.

5.7. API definition

5.7.1. General remarks

Good documentation is essential for every API so that developers can more easily learn how to use the API. In the best case, documentation would be available both in HTML for human consumption and in a machine readable format that can be best processed by software for run-time binding.

This standard specifies requirements and recommendations for APIs that share spatio-temporal resources and want to follow a standard way of doing so. In general, APIs will go beyond the requirements and recommendations stated in this standard. They will support additional operations, parameters, etc. that are specific to the API or the software tool used to implement the API.

5.7.2. Role of OpenAPI

This document uses OpenAPI 3.0 fragments as examples and to formally state requirements. Using OpenAPI 3.0 is not required for implementing an OGC API. Other API definition languages may be used along with, or instead of OpenAPI. However, any API definition language used should have an associated conformance class advertised through the `/conformance` path.

This approach is used to avoid lock-in to a specific approach to defining an API. This standard includes a conformance class for API definitions that follow the OpenAPI specification 3.0. Conformance classes for additional API definition languages will be added as the API landscape continues to evolve.

In this document, fragments of OpenAPI definitions are shown in YAML since YAML is easier to format than JSON and is typically used in OpenAPI editors.

5.7.3. References to OpenAPI components in normative statements

Some normative statements (requirements, recommendations and permissions) use a phrase that a component in the API definition of the server must be “based upon” a schema or parameter component in the OGC schema repository.

In this case, the following changes to the pre-defined OpenAPI component are permitted:

- If the server supports an XML encoding, `xml` properties may be added to the relevant OpenAPI schema components.
- The range of values of a parameter or property may be extended (additional values) or constrained (if a subset of all possible values are applicable to the server). An example for a constrained range of values is to explicitly specify the supported values of a string parameter or property using an `enum`.
- Additional properties may be added to the schema definition of a Response Object.
- Informative text may be changed or added, like comments or description properties.

For API definitions that do not conform to the OpenAPI Specification 3.0 the normative statement should be interpreted in the context of the API definition language used.

5.7.4. Paths in OpenAPI definitions

All paths in an OpenAPI definition are relative to the base URL of a server. Unlike Web Services, an API is decoupled from the server(s). Some ramifications of this are:

- An API may be hosted (replicated) on more than one server.
- Parts of an API may be distributed across multiple servers.

Example — URL of the OpenAPI definition: If the OpenAPI Server Object looks like this:

```
servers:
  - url: https://dev.example.org/
    description: Development server
  - url: https://data.example.org/
    description: Production server
```

The path `/mypath` in the OpenAPI definition of the API would be the URL `https://data.example.org/mypath` for the production server.

5.7.5. Reusable OpenAPI components

Reusable components for OpenAPI definitions for an OGC API are referenced from this document.

6

OVERVIEW

6 OVERVIEW

6.1. General

The OGC API standards enable access to resources using the HTTP protocol and its associated operations (GET, PUT, POST, etc.). OGC API-Common defines a set of facilities which are applicable to all OGC APIs. Other OGC standards extend API-Common with facilities specific to a resource type.

This OGC API-EDR standard defines an API with the following goals:

a) to make it easier to access a wide range of data through a uniform, well-defined simple Web interface;

b) To allow clients to retrieve a subset of data created by the API in response to a standardized, coordinate orientated, query pattern;

c) To provide 'building blocks' allowing the construction of more complex applications.

The EDR API can be considered a 'Sampling API'. The query creates a discrete sampling geometry against the spatio-temporal data resource of a relatively persistent data store. The query and its response are transient resources, which could be made persistent for re-use if required.

The functionality provided by EDR query patterns could be realized through specific implementation of the SOS (and to some extent WCS) from the OGC Web Services family of (XML-based) standards. EDR introduces a streamlined JSON-based OGC API implementation of building blocks that could be used for many of the simple similar use cases addressed by SOS and WCS in the past.

The EDR API defines behaviour for the HTTP GET operation. Future versions may introduce additional methods as required, consistent with RFC 7231.

6.2. Resource Paths

Table 2 summarizes the access paths and relation types defined in this standard.

Table 2 — Environmental Data Retrieval API Paths

PATH TEMPLATE	RELATION	RESOURCE
	Common	
{root}/	none	Landing page
{root}/api	`service-desc` or `service-doc`	API Description (optional)
{root}/conformance	`conformance`	Conformance Classes
	Collections	
{root}/collections	`data`	Metadata describing the collections of data available from this API.
{root}/collections/{collectionId}		Metadata describing the collection of data which has the unique identifier `{collectionId}`
	Features	
{root}/collections/{collectionId}/items	`items`	Retrieve metadata about available items
	Queries	
{root}/collections/{collectionId}/{queryType}		Retrieve data according to the query pattern
{root}/collections/{collectionId}/instances		Retrieve metadata about instances of a collection
{root}/collections/{collectionId}/instances/{instanceId}		Retrieve metadata from a specific instance of a collection which has the unique identifier `{instanceId}`

Where:

- `{root}` = Base URI for the API server
- `{collectionId}` = an identifier for a specific collection of data
- `{instanceId}` = an identifier for a specific version or instance of a collection of data
- `{queryType}` = an identifier for a specific query pattern to retrieve data from a specific collection of data

7

DEPENDENCIES ON CORE AND COLLECTIONS REQUIREMENTS CLASSES OF OGC API — COMMON

7 DEPENDENCIES ON CORE AND COLLECTIONS REQUIREMENTS CLASSES OF OGC API — COMMON

The OGC API-EDR standard is an extension of OGC API — Common — Part 1: Core and OGC API — Common — Part 2: Geospatial Data. Therefore, an implementation of OGC API-EDR shall first satisfy the appropriate Requirements Classes from OGC API — Common, namely:

- Core, http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
- Collections, http://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections

7.1. Overview

The `Core` Requirements Class of OGC API — Environmental Data Retrieval defines the requirements for locating, understanding, and accessing spatio-temporal data resources.

See Requirements Class "OGC API — Environmental Data Retrieval Core" for a detailed specification of the `Core` Requirements Class.

The following five sections explain aspects of the `Core`, `Collections` and `Queries` Requirements Classes:

a) API Platform: a set of common capabilities

b) Collection Access: operations for accessing collections of spatio-temporal data.

c) Query Resources: operations for accessing spatio-temporal data resources through queries

d) Parameters: parameters for use in OGC API-EDR operations.

e) General: general principles for use with this standard.

Table 3 Identifies the OGC API — Common Requirements Classes which are applicable to each section of this Standard. Instructions on when and how to apply these Requirements Classes are provided in each section.

Table 3 — Mapping OGC API — EDR Sections to OGC API — Common Requirements Classes

API-EDR SECTION	API-EDR REQUIREMENTS CLASS	API-COMMON REQUIREMENTS CLASS
API Landing Page	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/core	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
API Definition	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/core	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
Declaration of Conformance Classes	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/core	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
Collections	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/collections	http://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections
OpenAPI 3.0	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/oas30	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/oas30
GeoJSON	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/geojson	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/json
CoverageJSON	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/covjson	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/json
HTML	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/html	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/html

7.2. Platform

OGC API — Common defines a set of common capabilities which are applicable to any OGC Web API. Those capabilities provide the platform upon which resource-specific APIs can be built. This section describes those capabilities and any modifications needed to better support spatio-temporal data resources.

7.2.1. API landing page

The landing page provides links to start exploration of the resources offered by an API. Its most important component is a list of links. OGC API — Common already requires some common links, sufficient for this standard, that are stated in the following Requirements Class of OGC API — Common:

- Core, http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core

7.2.1.1. Operation

The `Landing Page` operation is defined in the `Core` conformance class of OGC API — Common. No modifications are needed to support spatio-temporal data resources. The `Core` conformance class specifies only one way of performing this operation:

a) Issue a `GET` request on the `{root}/` path

Support for `GET` on the `{root}/` path is required by OGC API — Common.

7.2.1.2. Response

A successful response to the `Landing Page` operation is defined in OGC API — Common. The schema for this resource is provided in Figure 1.

```
type: object
required:
  - links
properties:
  title:
    type: string
    example: Meteorological data server
  description:
    type: string
    example: Access to Meteorological data via a Web API that conforms to the
 OGC
      API Environmental Data Retrieval specification.
  links:
    type: array
    items:
      $ref: link.yaml
    example:
      - href: http://www.example.org/edr/api
        hreflang: en
        rel: service
        type: application/openapi+json;version=3.0
        title: ""
      - href: http://www.example.org/edr/conformance
        hreflang: en
        rel: data
        type: application/json
        title: ""
      - href: http://www.example.org/edr/collections
        hreflang: en
        rel: data
        type: application/json
        title: ""
  keywords:
    type: array
    items:
      type: string
    example:
      - Temperature
      - Wind
      - Point
```

```
      - Trajectory
  provider:
    type: object
    properties:
      name:
        description: Name of organization providing the service
        type: string
      url:
        description: Link to service providers website
        type: string
  contact:
    type: object
    properties:
      email:
        description: Email address of service provider
        type: string
      phone:
        description: Phone number of service provider
        type: string
      fax:
        description: Fax number of service provider
        type: string
      hours:
        type: string
      instructions:
        type: string
      address:
        type: string
      postalCode:
        type: string
      city:
        type: string
      stateorprovince:
        type: string
      country:
        type: string
```

Figure 1 — Landing Page Response Schema

The following JSON fragment is an example of a response to an OGC API-EDR Landing Page operation.

```
{
  "title": "string",
  "description": "string",
  "links": [
    {
      "href": "http://data.example.org/",
      "rel": "self",
      "type": "application/json",
      "title": "this document"
    },
    {
      "href": "http://data.example.org/api",
      "rel": "service-desc",
      "type": "application/openapi+json;version=3.0",
      "title": "the API definition"
    },
    {
      "href": "http://data.example.org/conformance",
      "rel": "conformance",
      "type": "application/json",
      "title": "OGC conformance classes implemented by this API"
```

```
    },
    {
      "href": "http://data.example.org/collections",
      "title": "Metadata about the resource collections"
    }
  ]
}
```

Figure 2 — Landing Page Example

7.2.1.3. Error Handling

The requirements for handling unsuccessful requests are provided in Recommendation http://www.opengis.net/spec/ogcapi-common-1/1.0/rec/core/problem-details of OGC API — Common. General guidance on HTTP status codes and how they should be handled is provided in Clause 9.2 — HTTP Status Codes.

7.2.2. API definition

Every API is required to provide a definition document that describes the capabilities of that API. This definition document can be used by developers to understand the API, by software clients to connect to the server, or by development tools to support the implementation of servers and clients.

Support for an API definition is specified in the following Requirements Class of OGC API — Common:

- Core, http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core

7.2.2.1. Operation

This operation is defined in the `Core` conformance class of OGC API — Common. No modifications are needed to support spatio-temporal data resources. The `Core` conformance class describes two ways of performing this operation:

a) Issue a `GET` request on the `{root}/api` path

b) Follow the `service-desc` or `service-doc` link on the landing page

Only the link is required by OGC API — Common.

7.2.2.2. Response

A successful response to the API Definition request is a resource which documents the design of the API. OGC API — Common leaves the selection of format for the API Definition response to

the API implementor. However, the options are limited to those which have been defined in the API-Common standard. At this time OpenAPI 3.0 is the only option provided.

7.2.2.3. Error Handling

The requirements for handling unsuccessful requests are provided in Recommendation http://www.opengis.net/spec/ogcapi-common-1/1.0/rec/core/problem-details of OGC API — Common. General guidance on HTTP status codes and how they should be handled is provided in Clause 9.2 — HTTP Status Codes.

7.2.3. Declaration of conformance classes

To support "generic" clients that want to access multiple OGC API standards and extensions — and not "just" a specific API server, the API has to declare the conformance classes it claims to have implemented.

Support for the declaration of conformance classes is specified in the following Requirements Class of OGC API — Common:

- Core, http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core

7.2.3.1. Operation

This operation is defined in the `Core` conformance class of OGC API — Common. No modifications are needed to support spatio-temporal data resources. The `Core` conformance class describes two ways of performing this operation:

a) Issue a `GET` request on the `{root}/conformance` path

b) Follow the `conformance` link on the landing page

Both techniques are required by OGC API — Common.

7.2.3.2. Response

A successful response to the Conformance operation is a list of URLs. Each URL identifies an OGC Conformance Class for which this API claims conformance. The schema for this resource is defined in OGC API — Common and provided for reference in Figure 3.

Apply Requirement /req/core/conformance on declaration of Core conformance classes.

```
type: object
required:
  - conformsTo
properties:
  conformsTo:
    type: array
```

```
    items:
      type: string
```

Figure 3 — Conformance Response Schema

The following JSON fragment is an example of a response to an OGC API-EDR conformance operation.

```
{
  "conformsTo": [
    "http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core",
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/core",
    "http://www.opengis.net/spec/ogcapi-common-2/1.0/conf/collections",
    "http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/oas30",
    "http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/html",
    "http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/geojson"
  ]
}
```

Figure 4 — Conformance Information Example

8

QUERY, SPATIO-TEMPORAL AND INFORMATION RESOURCES

8 QUERY, SPATIO-TEMPORAL AND INFORMATION RESOURCES

Query resources are spatio-temporal queries which support operation of the API for the access and use of the spatio-temporal data resources. The OGC API-EDR standard has identified an initial set of common `queryTypes` to implement, described in Clause 8.2. This list may change as the standard is used and experience gained.

A spatio-temporal data resource is a collection of spatio-temporal data that can be sampled using the OGC API-EDR query patterns.

This clause specifies the “Collections” Requirements Class. The detailed specification of the Requirements Class is in Annex A.3.1.

Query resources related to spatio-temporal data resources (collections of spatio-temporal data) can be exposed using the path templates:

- `/collections/{collectionId}/{queryType}`
- `/collections/{collectionId}/instances/{instanceId}/{queryType}`

Where

`{collectionId}` = a unique identifier for a collection of spatio-temporal data.

`{instanceId}` = a text string identifying the version or instance of the chosen collection.

`{queryType}` = a text string identifying the query pattern performed by the API.

The `instanceId` parameter allows support for multiple instances or versions of the same underlying data source to be accessed by the API. This is applicable when the entire data source has been regenerated rather than individual values in the data source being changed. If only one instance of the data source exists a value of `default` or `latest` could be used.

Information resources associated with a specific collection should be accessed through the `/collections` path. Information resources not associated with a specific collection should be accessed via the `/{instanceId}/{queryType}` path template.

The resources returned from each node in these templates are described in Table 4.

8.1. Information Resources

Table 4 — Information Resource Paths

PATH TEMPLATE	RESOURCE
/collections	The root resource describing the collections of spatio-temporal data available from this API.
/collections/{collectionId}	Identifies a collection of spatio-temporal data with the unique identifier `{collectionId}`
/collections/{collectionId}/{queryType}	Identifies an Information Resource of type {queryType} associated with the `{collectionId}` collection.

The OGC API — Common specification does not define any information resource types. However Table 5 provides a mapping of the initial query types proposed for the EDR API.

8.2. Query Resources

Table 5 — Query Types

PATH TEMPLATE	QUERY TYPE	DESCRIPTION
/collections/{collectionId}/position	Position	Return data for the requested position
/collections/{collectionId}/radius	Radius	Return data within a given radius of a position
/collections/{collectionId}/area	Area	Return data for the requested area
/collections/{collectionId}/cube	Cube	Return data for a spatial cube
/collections/{collectionId}/trajectory	Trajectory	Return data along a defined trajectory
/collections/{collectionId}/corridor	Corridor	Return data within a spatio-temporal corridor
/collections/{collectionId}/items	Items	Items associated with the `{collectionId}` collection.
/collections/{collectionId}/locations	Locations	Location identifers associated with the `{collectionId}` collection.
/collections/{collectionId}/instances	Instances	List the available instances of the collection

8.2.1. Shared query parameters

Query parameters are used in URLs to define the resources which are returned on a GET request. The following are defined as standard shared parameters for use.

8.2.1.1. Parameter coords

Apply Requirement /req/edr/coords-definition for Parameter coords definition.

Apply Requirement /req/edr/coords-response for Parameter coords response.

8.2.1.2. Parameter datetime

Apply Requirement /req/core/datetime-definition for datetime definition.

Apply Requirement /req/core/datetime-response for datetime response.

The `datetime` parameter is defined in OGC API — Common. The following information is provided here as a convenience.

"Intersects" means that the time (instant or duration) specified in the parameter `datetime` includes a timestamp that is part of the temporal geometry of the resource (again, a time instant or duration). Time durations include the start and end times.

Example 1 — A datetime: February 12, 2018, 23:20:52 GMT:
`datetime=2018-02-12T23%3A20%3A52Z`

For resources with a temporal property that is a timestamp (like `lastUpdate`), a `datetime` value would match all resources where the temporal property is identical.

For resources with a temporal property that is a date or a time interval, a `datetime` value would match all resources where the timestamp is on that day or within the time interval.

Example 2 — Intervals: February 12, 2018, 00:00:00 GMT to March 18, 2018, 12:31:12 GMT: `datetime=2018-02-12T00%3A00%3A00Z%2F2018-03-18T12%3A31%3A12Z` February 12, 2018, 00:00:00 UTC or later: `datetime=2018-02-12T00%3A00%3A00Z%2F..` March 18, 2018, 12:31:12 UTC or earlier: `datetime=..%2F2018-03-18T12%3A31%3A12Z`

A template for the definition of the parameter in YAML according to OpenAPI 3.0 is available at datetime.yaml.

8.2.1.3. Parameter parameter-name

Apply Requirement /req/edr/REQ_rc-parameter-name-definition for Parameter parameter-name definition.

Apply Requirement /req/edr/parameter-name-response for Parameter parameter-name response.

Example 1 — A single parameter: Only return values for the Maximum_temperature `parameter-name=Maximum_temperature`

Example 2 — Return multiple parameters: Values for the Maximum_temperature, Minimum_temperature and Total_precipitation `parameter-name=Maximum_temperature,Minimum_temperature,Total_precipitation`

For the requested parameters which do not exist in the collection, null values should be returned. If none of the requested parameters exist in the collection, a `400` message SHOULD be returned.

8.2.1.4. Parameter crs

Apply Requirement /req/edr/REQ_rc-crs-definition for Parameter crs definition.

Apply Requirement /req/edr/REQ_rc-crs-response for Parameter crs response.

The value of the `crs` query parameter will be one of the name values described in the collection metadata for supported coordinate reference system transformations.

8.2.1.5. Parameter f

Apply Requirement /req/edr/rc-f-definition for Parameter f definition.

Apply Requirement /req/edr/REQ_rc-f-response for Parameter f response.

Example — Return data as coverageJSON: `f=coverageJSON`

If not specified, the query will return data in the native format of the collection. If the requested format system does not match an entry in the defined list of valid output formats for the collection, a `400` message SHOULD be returned.

8.2.2. Position query

The Position query returns data for the requested coordinate. Logic for identifying the best match for the coordinate will depend on the collection . The filter constraints are defined by the following query parameters:

8.2.2.1. Parameter coords

Apply Requirement /req/edr/coords-definition for Parameter coords definition.

Apply Requirement /req/edr/coords-response for Parameter coords response.

Accepts position(s) to return data for. The coordinates are defined by a Well Known Text (WKT) string. To retrieve a single position:

```
POINT(x y)
```

And for a list of positions:

```
MULTIPOINT((x y),(x1 y1),(x2 y2),(x3 y3))
```

And for a list of positions at defined heights:

```
MULTIPOINTZ((x y z),(x1 y1 z1),(x2 y2 z2),(x3 y3 z3))
```

See http://portal.opengeospatial.org/files/?artifact_id=25355 and https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry .

The coordinate values will depend on the CRS parameter. If this is not defined, the values will be assumed to be WGS84 values (i.e x=longitude and y=latitude).

Example 1 — Single position: Retrieve data for Greenwich, London: `coords=POINT(0 51.48)`

Example 2 — Multiple positions: Retrieve data for a list of positions: 38.9N 77W, 48.85N 2.35E, 39.92N 116.38E, 35.29S 149.1E, 51.5N 0.1W: `coords=MULTIPOINT((-77 38.9),(2.35 48.85),(116.38 39.92),(149.1 -35.29),(-0.1 51.5))` Note that for this example, the coordinate reference system has the coordinate order: (longitude/latitude)

8.2.2.2. Parameter z

Apply Requirement /req/edr/z-definition for Parameter z definition.

Apply Requirement /req/edr/z-response for Parameter z response.

Define the vertical level to return data from i.e. z=level

Example 1 — A single vertical level: For example, if the 850hPa pressure level is being queried: `z=850`

Example 2 — Return data at all a levels defined by a list of vertical levels: Request data at levels 1000hPa, 900hPa, 850hPa, and 700hPa: `z=1000,900,850,700`

Example 3 — Return data for all levels between and including 2 defined levels: Request data for all levels between 2m and 100m: `z=2/100`

```
When not specified the API MUST return data from all available levels
```

8.2.2.3. Parameter datetime

For Parameter datetime, see Clause 8.2.1.2.

8.2.2.4. Parameter parameter-name

For Parameter parameter-name, see Clause 8.2.1.3.

8.2.2.5. Parameter crs

For Parameter crs, see Clause 8.2.1.4.

8.2.2.6. Parameter f

For Parameter f, see Clause 8.2.1.5.

8.2.3. Radius query

The Radius query returns data within the defined radius of the requested coordinate. The filter constraints are defined by the following query parameters:

8.2.3.1. Parameter coords

Apply Requirement /req/edr/coords-definition for Parameter coords definition.

Apply Requirement /req/edr/coords-response for Parameter coords response.

position(s) to return data for, the coordinates are defined by a Well Known Text (wkt) string.

To retrieve data for a single position:

POINT(x y)

A position at height z:

POINT(x y z)

And for a list of positions:

MULTIPOINT((x y),(x1 y1),(x2 y2),(x3 y3))

And for a list of positions at defined heights:

MULTIPOINTZ((x y z),(x1 y1 z1),(x2 y2 z2),(x3 y3 z3))

See http://portal.opengeospatial.org/files/?artifact_id=25355 and https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry

The coordinate values will depend on the CRS parameter. If this is not defined, the values will be assumed to be WGS84 values (i.e x=longitude and y=latitude)

Example — Single position: Retrieve data for Greenwich, London `coords=POINT(0 51.48)`

8.2.3.2. Parameter within

Apply Requirement /req/edr/within-definition for Parameter `within` definition.

Apply Requirement /req/edr/REQ_rc-within-response for Parameter `within` response.

8.2.3.3. Parameter within-units

The units supported by the collection will be listed in the collections end point

Apply Requirement /req/edr/within-units-definition for Parameter `within-units` definition.

Apply Requirement /req/edr/REQ_rc-within-units-response for Parameter `within-units` response.

Example 1 — Define a 20Km radius: I.e. Define a Radius of 20 Km from the position defined by the coords query parameter `within=20&within-units=km`

Example 2 — Define a 5 mile radius: i.e. Define a Radius of 5 miles from the position defined by the coords query parameter `within=5&within-units=miles`

8.2.3.4. Parameter z

Define the vertical level to return data from, i.e. z=level

Example — A single vertical level: For example if the 80m level is being queried: `z=80`

When z is not specified, the API MUST return data from all available levels.

8.2.3.5. Parameter datetime

For Parameter datetime, see Clause 8.2.1.2.

8.2.3.6. Parameter parameter-name

For Parameter parameter-name, see Clause 8.2.1.3.

8.2.3.7. Parameter crs

For Parameter crs, see Clause 8.2.1.4.

8.2.3.8. Parameter f

For Parameter f, see Clause 8.2.1.5.

8.2.4. Area query

The Area query returns data within the polygon defined by the `coords` parameter. The height or time of the area are specified through separate parameters. The results are further filtered by the constraints defined by the following query parameters:

8.2.4.1. Parameter coords

Apply Requirement /req/edr/coords-definition for Parameter coords definition.

Apply Requirement /req/edr/coords-response for Parameter coords response.

Only data that has a geometry that intersects the area defined by the polygon are selected.

The polygon is defined using a Well Known Text string following

```
coords=POLYGON((x y,x1 y1,x2 y2,…,xn yn, x y))
```

which are values in the coordinate system defined by the crs query parameter (if crs is not defined the values will be assumed to be WGS84 longitude/latitude coordinates).

For instance a polygon that roughly describes an area that contains South West England in WGS84 would look like:

```
coords=POLYGON((-6.1 50.3,-4.35 51.4,-2.6 51.6,-2.8 50.6,-5.3 49.9,-6.1 50.3))
```

see http://portal.opengeospatial.org/files/?artifact_id=25355 and https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry

The coords parameter will only support 2D POLYGON.

Example 1 — A polygon covering the UK: An area covering the UK in WGS84 (from 15°W to 5°E and from 60.95°S to 48.8°S) `coords=POLYGON((-15 48.8,-15 60.95,5 60.85,5 48.8,-15 48.8))`

Example 2 — Multiple areas: Selecting data for two different regions
`coords=MULTIPOLYGON((-15 48.8,-15 60.95,5 60.85,5 48.8,-15 48.8),(-6.1 50.3,-4.35 51.4,-2.6 51.6,-2.8 50.6,-5.3 49.9,-6.1 50.3))`

8.2.4.2. Parameter z

Apply Requirement /req/edr/z-definition for Parameter z definition.

Apply Requirement /req/edr/z-response for Parameter z response.

Define the vertical level to return data from i.e. z=level

Example 1 — A single vertical level: For example if the 850hPa pressure level is being queried `z=850`

Example 2 — Return data at all a levels defined by a list of vertical levels: Request data at levels 1000hPa, 900hPa, 850hPa, and 700hPa `z=1000,900,850,700`

Example 3 — Return data for all levels between and including 2 defined levels: Request data for all levels between 2m and 100m `z=2/100`

`When not specified the API MUST return data from all available levels`

8.2.4.3. Parameter datetime

For Parameter datetime, see Clause 8.2.1.2.

8.2.4.4. Parameter parameter-name

For Parameter parameter-name, see Clause 8.2.1.3.

8.2.4.5. Parameter crs

For Parameter crs, see Clause 8.2.1.4.

8.2.4.6. Parameter f

For Parameter f, see Clause 8.2.1.5.

8.2.5. Cube query

The `Cube` query returns a data cube defined by the `bbox` and `z` parameters. The results are further filtered by the constraints defined by the following query parameters:

8.2.5.1. Parameter bbox

Apply Requirement /req/core/rc-bbox-definition for Parameter bbox definition.

Apply Requirement /req/core/rc-bbox-response for Parameter bbox response.

Only data that has a geometry that intersects the area defined by the bbox are selected.

- Lower left corner, coordinate axis 1
- Lower left corner, coordinate axis 2
- Upper right corner, coordinate axis 1
- Upper right corner, coordinate axis 2

bbox=minx,miny,maxx,maxy

The X and Y coordinates are values in the coordinate system defined by the crs query parameter. If crs is not defined, the values will be assumed to be WGS84 longitude/latitude coordinates and heights will be assumed to be in metres above mean sea level.

For instance a bbox that roughly describes an area that contains South West England in WGS84 would look like

Example — A cube covering the South West of the UK: `bbox=-6.0 50.0,-4.35 52.0`

8.2.5.2. Parameter z

Apply Requirement /req/edr/z-definition for Parameter z definition.

Apply Requirement /req/edr/cube-z-response for Parameter z response for cube queries.

Example 1 — A cube covering all data between vertical levels 100 and 550: `z=100/550`

Example 2 — A cube covering data at vertical levels 10,80,100: `z=10,80,200`

Example 3 — A cube covering data for 20 vertical levels at 50 unit intervals starting a level 100: `z=R20/100/50`

8.2.5.3. Parameter datetime

For Parameter datetime, see Clause 8.2.1.2.

8.2.5.4. Parameter parameter-name

For Parameter parameter-name, see Clause 8.2.1.3.

8.2.5.5. Parameter crs

For Parameter crs, see Clause 8.2.1.4.

8.2.5.6. Parameter f

For Parameter f, see Clause 8.2.1.5.

8.2.6. Trajectory query

The Trajectory query returns data along the path defined by the `coords` parameter. **The logic to match the data for the requested path will depend on and be defined by the collection**. The results are further filtered by the constraints defined by the following query parameters:

8.2.6.1. Parameter coords

Apply Requirement /req/edr/coords-definition for Parameter coords definition.

Apply Requirement /req/edr/coords-response for Parameter coords response.

“Intersects” means that the geospatial shape specified by the parameter `coords`, includes a coordinate that is part of the (spatial) geometry of the resource. This includes the boundaries of the geometries.

The trajectory query supports the Linestring Well Known Text (WKT) geometry type, the trajectory query SHOULD support 2D, 3D and 4D queries allowing the definition of a vertical level value (z) and a time value (as an epoch time) therefore coordinates for geometries may be 2D (x, y), 3D (x, y, z) or 4D (x, y, z, t).

A 2D trajectory, on the surface of earth with no time or vertical dimensions:
coords=LINESTRING(-3.53 50.72, -3.35 50.92, -3.11 51.02, -2.85 51.42, -2.59 51.46)

A 2D trajectory, on the surface of earth all at the same time and no vertical dimension, time value defined in ISO8601 format by the `datetime` query parameter : coords=LINESTRING(-3.53 50.72, -3.35 50.92, -3.11 51.02, -2.85 51.42, -2.59 51.46)&datetime=2018-02-12T23:00:00Z

A 2D trajectory, on the surface of earth with no time value but at a fixed height level, height defined in the collection height units by the `z` query parameter : coords=LINESTRING(-3.53 50.72, -3.35 50.92, -3.11 51.02, -2.85 51.42, -2.59 51.46)&z=850

A 2D trajectory, on the surface of earth all at the same time and at a fixed height level, time value defined in ISO8601 format by the `datetime` query parameter and height defined in the collection height units by the `z` query parameter : coords=LINESTRING(-3.53 50.72, -3.35 50.92, -3.11 51.02, -2.85 51.42, -2.59 51.46)&datetime=2018-02-12T23:00:00Z&z=850

A 3D trajectory, on the surface of the earth but over a range of time values with no height values: coords=LINESTRINGM(-3.53 50.72 1560507000,-3.35 50.92 1560508800,-3.11 51.02 1560510600,-2.85 51.42 1560513600,-2.59 51.46 1560515400)

A 3D trajectory, on the surface of the earth but over a range of time values with a fixed vertical height value, height defined in the collection height units by the `z` query parameter : coords=LINESTRINGM(-3.53 50.72 1560507000,-3.35 50.92 1560508800,-3.11 51.02 1560510600,-2.85 51.42 1560513600,-2.59 51.46 1560515400)&z=200

A 3D trajectory, through a 3D volume with vertical height or depth, but no defined time: coords=LINESTRINGZ(-3.53 50.72 0.1,-3.35 50.92 0.2,-3.11 51.02 0.3,-2.85 51.42 0.4,-2.59 51.46 0.5)

A 3D trajectory, through a 3D volume with height or depth, but at a fixed time value defined in ISO8601 format by the `datetime` query parameter: coords=LINESTRINGZ(-3.53 50.72 0.1,-3.35 50.92 0.2,-3.11 51.02 0.3,-2.85 51.42 0.4,-2.59 51.46 0.5)&datetime=2018-02-12T23:00:00Z

A 4D trajectory, through a 3D volume and over a range of time values: coords=LINESTRINGZM(-3.53 50.72 0.1 1560507000,-3.35 50.92 0.2 1560508800,-3.11 51.02 0.3 1560510600,-2.85 51.42 0.4 1560513600,-2.59 51.46 0.5 1560515400)

If the coords specify a 4D trajectory i.e. coords=LINESTRINGZM(...) an error MUST be thrown by the server if the client application defines either the `z` or `datetime` query parameters

where `Z` in `LINESTRINGZ` and `LINESTRINGZM` refers to the height value. `If the specified CRS does not define the height units, the heights units will default to metres above mean sea level`

and the `M` in `LINESTRINGM` and `LINESTRINGZM` refers to the number of seconds that have elapsed since the Unix epoch, that is the time 00:00:00 UTC on 1 January 1970. See https://en.wikipedia.org/wiki/Unix_time

Example 1 — A basic surface route: From Bristol to Exeter `coords=LINESTRING(-3.53 50.72,-3.35 50.92,-3.11 51.02,-2.85 51.42,-2.59 51.46)`

Example 2 — A basic surface route with defined time intervals: From Bristol to Exeter `coords=LINESTRINGM(-3.53 50.72 1560507000,-3.35 50.92 1560508800,-3.11 51.02 1560510600,-2.85 51.42 1560513600,-2.59 51.46 1560515400)`

8.2.6.2. Parameter z

Used when the entire trajectory occurs at the same vertical coordinate. The `z` query parameter is used to define the height

Example — A single vertical level: If the entire route is at the 850hPa pressure level:
`coords=LINESTRINGM(-3.53 50.72 1560507000,-3.35 50.92 1560508800,-3.11 51.02 1560510600,-2.85 51.42 1560513600,-2.59 51.46 1560515400)&z=850`

8.2.6.3. Parameter datetime

For Parameter datetime, see Clause 8.2.1.2.

8.2.6.4. Parameter parameter-name

For Parameter parameter-name, see Clause 8.2.1.3.

8.2.6.5. Parameter crs

For Parameter crs, see Clause 8.2.1.4.

8.2.6.6. Parameter f

For Parameter f, see Clause 8.2.1.5.

8.2.7. Corridor query

The Corridor query returns data along and around the path defined by the `coords` parameter. **The logic to match the data for the requested path will depend on, and be defined by, the collection**. The results are further filtered by the constraints defined by the following query parameters:

8.2.7.1. Parameter coords

Apply Requirement /req/edr/coords-definition for Parameter coords definition.

Apply Requirement /req/edr/coords-response for Parameter coords response.

“Intersects” means that the geospatial shape specified by the parameter `coords`, includes a coordinate that is part of the (spatial) geometry of the resource. This includes the boundaries of the geometries.

The corridor query supports the Linestring Well Known Text (WKT) geometry type, the corridor query SHOULD support 2D, 3D and 4D queries allowing the definition of a vertical height value (z) and a time value (as an epoch time) therefore coordinates for geometries may be 2D (x, y), 3D (x, y, z) or 4D (x, y, z, t). The Linestring described by the coords parameter defines the center

point of the corridor with the corridor-height and corridor-width query parameters defining the depth and breadth of the corridor.

A 2D corridor, on the surface of earth with no time or vertical dimensions: coords=LINESTRING(-3.53 50.72,-3.35 50.92,-3.11 51.02,-2.85 51.42,-2.59 51.46)

A 2D corridor, on the surface of earth all at the same time and no vertical dimension, time value defined in ISO8601 format by the `datetime` query parameter : coords=LINESTRING(-3.53 50.72,-3.35 50.92,-3.11 51.02,-2.85 51.42,-2.59 51.46)&datetime=2018-02-12T23:00:00Z

A 2D corridor, on the surface of earth with no time value but at a fixed vertical height, height defined in the collection height units by the `z` query parameter : coords=LINESTRING(-3.53 50.72,-3.35 50.92,-3.11 51.02,-2.85 51.42,-2.59 51.46)&z=850

A 2D corridor, on the surface of earth all at a the same time and at a fixed vertical height, time value defined in ISO8601 format by the `datetime` query parameter and height defined in the collection height units by the `z` query parameter : coords=LINESTRING(51.14 -2.98, 51.36 -2.87, 51.03 -3.15, 50.74 -3.48, 50.9 -3.36)&datetime=2018-02-12T23:00:00Z&z=850

A 3D corridor, on the surface of the earth but over a range of time values with no height values: coords=LINESTRINGM(51.14 -2.98 1560507000, 51.36 -2.87 1560507600, 51.03 -3.15 1560508200, 50.74 -3.48 1560508500, 50.9 -3.36 1560510240)

A 3D corridor, on the surface of the earth but over a range of time values with a fixed height value, height defined in the collection height units by the `z` query parameter : coords=LINESTRINGM(51.14 -2.98 1560507000, 51.36 -2.87 1560507600, 51.03 -3.15 1560508200, 50.74 -3.48 1560508500, 50.9 -3.36 1560510240)&z=200

A 3D corridor, through a 3D volume with vertical height or depth, but no defined time: coords=LINESTRINGZ(-3.53 50.72 0.1,-3.35 50.92 0.2,-3.11 51.02 0.3,-2.85 51.42 0.4,-2.59 51.46 0.5)

A 3D corridor, through a 3D volume with a vertical extent, but at a fixed time, time value defined in ISO8601 format by the `datetime` query parameter: coords=LINESTRINGZ(-3.53 50.72 0.1,-3.35 50.92 0.2,-3.11 51.02 0.3,-2.85 51.42 0.4,-2.59 51.46 0.5)&datetime=2018-02-12T23:00:00Z

A 4D corridor, through a 3D volume but over a range of time values: coords=LINESTRINGZM (-3.53 50.72 0.1 1560507000,-3.35 50.92 0.2 1560508800,-3.11 51.02 0.3 1560510600,-2.85 51.42 0.41560513600,-2.59 51.46 0.5 1560515400)

If the coords specify a 4D corridor i.e. coords=LINESTRINGZM(...) an error MUST be thrown by the server if the client application defines either the `z` or `datetime` query parameters

where **Z** in `LINESTRINGZ` and `LINESTRINGZM` refers to the height value. `If the specified CRS does not define the height units, the heights units will default to metres above mean sea level`

and the **M** in `LINESTRINGM` and `LINESTRINGZM` refers to the number of seconds that have elapsed since the Unix epoch, that is the time 00:00:00 UTC on 1 January 1970. See https://en.wikipedia.org/wiki/Unix_time

Example 1 — A basic surface route 2D corridor: From Bristol to Exeter, for a width of 5 km. The units are specified by the `width-units`. `coords=LINESTRING(-3.53 50.72,-3.35 50.92,-3.11 51.02,-2.85 51.42,-2.59 51.46)&corridor-width=5&width-units=km`

Example 2 — A surface route 2D corridor with defined time intervals, 5km wide: From Bristol to Exeter `coords=LINESTRINGM(-3.53 50.72 1560507000,-3.35 50.92 1560508800,-3.11 51.02 1560510600,-2.85 51.42 1560513600,-2.59 51.46 1560515400)&corridor-width=5000&width-units=m`

Example 3 — A 4D corridor with defined time intervals and changing pressure heights: From Bristol to Exeter `coords=LINESTRINGZM(-3.53 50.72 1000 1560507000,-3.35 50.92 850 1560508800,-3.11 51.02 700 1560510600,-2.85 51.42 850 1560513600,-2.59 51.46 1000 1560515400)&corridor-width=5&width-units=km`

8.2.7.2. Parameter z

Used when the entire corridor occurs at the same vertical height. The `z` query parameter is used to define the height.

Example — A corridor with a single vertical height: If the entire route corridor is at the 850hPa pressure level `coords=LINESTRINGM(-3.53 50.72 1560507000,-3.35 50.92 1560508800,-3.11 51.02 1560510600,-2.85 51.42 1560513600,-2.59 51.46 1560515400)&corridor-width=5&width-units=km&z=850`

8.2.7.3. Parameter datetime

For Parameter datetime, see Clause 8.2.1.2.

8.2.7.4. Parameter resolution-x

For Parameter resolution-x, see Annex A.2.21.

Example 1 — Interpolate corridor values across the width: Return 10 values across the defined corridor width `resolution-x=10`

Example 2 — Get values across the width of the corridor at stored resolution: Return values at the stored resolution `resolution-x=0`

8.2.7.5. Parameter resolution-z

For Parameter resolution-z, see Annex A.2.26.

Example 1 — Interpolate corridor values over the corridor height: Return 8 values over the defined corridor height `resolution-z=8`

Example 2 — Get values over the defined corridor height at stored resolution: Return values at the stored resolution `resolution-z=0`

8.2.7.6. Parameter corridor-height

For Parameter corridor-height, see Annex A.2.28.

8.2.7.7. Parameter height-units

For Parameter height-units, see Annex A.2.30.

8.2.7.8. Parameter corridor-width

For Parameter corridor-width, see Annex A.2.32.

8.2.7.9. Parameter width-units

For Parameter width-units, see Annex A.2.34.

8.2.7.10. Parameter parameter-name

For Parameter parameter-name, see Clause 8.2.1.3.

8.2.7.11. Parameter crs

For Parameter crs, see Clause 8.2.1.4.

8.2.7.12. Parameter f

For Parameter f, see Clause 8.2.1.5.

8.2.8. Items query

The EDR Items query is an OGC API — Features endpoint that may be used to catalog pre-existing EDR sampling features. The pre-existence of an EDR sampling feature may be because of the existence of a monitoring location, because a particular query has been cached for later use, or for an array of application-specific use cases (e.g. a catalog of spatio-temporal domains of anomalies in a dataset). A GeoJSON-compatible JSON-Schema has been specified to document

an EDR query endpoint and valid query parameters including time range, parameters, and spatial characteristics.

RECOMMENDATION 1
/rec/core/edr-geojson
A: If a collection using other EDR queries uses the `items` query, implementations SHOULD consider support for the EDR GeoJSON Schema as an encoding.

8.2.8.1. Parameter itemId

If an itemId is not specified, the query will return a list of the available itemId's. This behavior is specified in OGC API — Features. All other parameters for use with the Items query are defined by OGC API — Features.

Example 1 — List available items: `/collections/{collectionId}/items` e.g. return query parameters to retrieve tropical storms using OGC API — Features and the EDR FeatureCollection GeoJSON schema. Each item would include an area query end point, a time range, a list of available parameters, and a representative geojson geometry. `/collections/tropical_storms/items` e.g. return query parameters to retrieve monitoring data from a collection end point, a time range, a list of available parameters and a representative geojson geometry. `/collections/stream_gages/items`

Example 2 — itemId: `/collections/{collectionId}/items/{itemId}` e.g. return information for the requested item with an id of KIAD_2020-05-19T00Z from the Metar collection. Returned data would include a location query end point, time range, a list of available parameters, and a representative geometry for the KIAD METAR station. `/collections/metar/items/KIAD_2020-05-19T00Z` e.g. return information for the requested item with an id of warning_12345 from the forecast collection. Returned data would include an area query end point, time range, a list of available parameters and a representative geometry for the warning_12345 warning area. `/collections/forecast/items/warning_12345`

8.2.9. Locations query

8.2.9.1. Parameter locationId

With the locations query a position is defined by a unique identifier that is a string value. It can be anything as long as it is unique for the required position, for instance a GeoHash `gbsvn` or a World Meteorological Organization (WMO) station identifier like `03772`, or place name like `Devon`. The metadata returned by the API must supply a geospatial definition for the identifier.

Example 1 — get a list of locationId's: /collections/{collectionID}/locations/ return a list of location identifiers and relevant metadata for the metar collection /collections/metar/locations/ Valid locationId's can also be discovered via another mechanism such as the items query.

Example 2 — locationId: /collections/{collectionID}/locations/{locationId} return all available data for the metar collection for the requested location identifier, where the location is defined by the Heathrow METAR identifier /collections/metar/locations/EGLL

8.2.9.2. Parameter datetime

For Parameter datetime, see Clause 8.2.1.2.

8.2.9.3. Parameter parameter-name

For Parameter parameter-name, see Clause 8.2.1.3.

8.2.9.4. Parameter crs

For Parameter crs, see Clause 8.2.1.4.

8.2.9.5. Parameter f

For Parameter f, see Clause 8.2.1.5.

8.2.10. Instances query

It is not unusual in the scientific world for there to be multiple versions or instances of the same collection, where the same information is reprocessed or regenerated. Although they could be described as new collections the instance query type allows this data to be described as different views of the same collection.

8.2.10.1. Parameter instanceId

A unique identifier for the instance of the collection

/collections/{collectionId}/instance/{instanceId}

Example 1 — Return the Raw data instance metadata (instanceId = raw) for the Metar ((collectionId = metar) collection: /collections/metar/instance/raw

Example 2 — Return the Level 1 Quality controlled data instance (instanceId = qc_lvl_1) metadata for the Metar (collectionId = metar) collection: /collections/metar/instance/qc_lvl_1

8.2.10.2. Parameter queryType

The queryType options are exactly the same as those available to collections that do not have multiple instances and support the same query parameters and functionality. See the Table 5 for the mappings of the query types.

/collections/{collectionId}/instance/{instanceId}/{queryType}

See the Clause 8.2 section for details of the query parameters supported by the queryTypes.

Example 1 — A position query on a Raw data instance(instanceId = raw) for the Metar ((collectionId = metar) collection: /collections/metar/instance/raw/position

Example 2 — A trajectory query on a Raw data instance(instanceId = raw) for the Metar ((collectionId = metar) collection: /collections/metar/instance/raw/trajectory

9

GENERAL REQUIREMENTS

9 GENERAL REQUIREMENTS

The following general requirements and recommendations apply to all OGC APIs.

9.1. HTTP 1.1

The standards used for Web APIs are built on the HTTP protocol. Therefore, conformance with HTTP or a closely related protocol is required.

Apply Requirement /req/core/http for HTTP support.

9.2. HTTP Status Codes

Table 6 lists the main HTTP status codes that clients should be prepared to receive. This includes support for specific security schemes or URI redirection. In addition, other error situations may occur in the transport layer outside of the server.

Table 6 — Typical HTTP status codes

STATUS CODE	DESCRIPTION
`200`	A successful request.
`202`	A successful request, but the response is still being generated. The response will include a `Retry-After` header field giving a recommendation in seconds for the client to retry.
`204`	A successful request, but the resource has no data resulting from the request. No additional content or message body is provided.
`304`	An entity tag was provided in the request and the resource has not been changed since the previous request.
`308`	The server cannot process the data through a synchronous request. The response includes a `Location` header field which contains the URI of the location the result will be available at once the query is complete Asynchronous queries.
`400`	The server cannot or will not process the request due to an apparent client error. For example, a query parameter had an incorrect value.
`401`	The request requires user authentication. The response includes a `WWW-Authenticate` header field containing a challenge applicable to the requested resource.

STATUS CODE	DESCRIPTION
403	The server understood the request, but is refusing to fulfill it. While status code 401 indicates missing or bad authentication, status code 403 indicates that authentication is not the issue, but the client is not authorised to perform the requested operation on the resource.
404	The requested resource does not exist on the server. For example, a path parameter had an incorrect value.
405	The request method is not supported. For example, a POST request was submitted, but the resource only supports GET requests.
406	Content negotiation failed. For example, the Accept header submitted in the request did not support any of the media types supported by the server for the requested resource.
413	Request entity too large. For example the query would involve returning more data than the server is capable of processing, the implementation should return a message explaining the query limits imposed by the server implementation.
500	An internal error occurred in the server.

More specific guidance is provided for each resource, where applicable.

Permission 1	/per/core/additional-status-codes
A	Servers MAY support other capabilities of the HTTP protocol and, therefore, MAY return other status codes than those listed in Table 6, too.

9.3. Web Caching

Entity tags are a mechanism for web cache validation and for supporting conditional requests to reduce network traffic. Entity tags are specified by HTTP/1.1 (RFC 2616).

RECOMMENDATION 2
/rec/core/etag
A: The service SHOULD support entity tags and the associated headers as specified by HTTP/1.1.

9.4. Support for Cross-Origin Requests

Access to data from a HTML page is by default prohibited for security reasons, if the data is located on another host than the webpage ("same-origin policy"). A typical example is a web-application accessing feature data from multiple distributed datasets.

RECOMMENDATION 3
/rec/core/cross-origin
A: If the server is intended to be accessed from the browser, cross-origin requests SHOULD be supported. Note that support can also be added in a proxy layer on top of the server.

Two common mechanisms to support cross-origin requests are:

- Cross-origin resource sharing (CORS)
- JSONP (JSON with padding)

9.5. Asynchronous queries

It will not always be possible to respond to queries synchronously. This standard does not specify how to handle any asynchrony. Different services may propose different best practices.

For example, if the query requires handling requests asynchronously, one option, but there are others, is that the system could respond with a HTTP code of `308` and include a `Location` response header field with the URI of the location of the data once the query has completed. If the user queries the URI of the product of the query before the data is available that response should respond with a HTTP code of `202` and include a `Retry-after` response header field with a suggested interval in seconds to retry the data retrieval.

9.6. Coordinate Reference Systems

As discussed in Chapter 9 of the OGC/W3C Spatial Data on the Web Best Practices document, how to express and share the location of resources in a consistent way is one of the most fundamental aspects of publishing geospatial or spatio-temporal data and it is important to be clear about the coordinate reference system that the coordinates use.

For the reasons discussed in the Best Practices, EDR APIs MUST always support WGS84 longitude and latitude (http://www.opengis.net/def/crs/OGC/1.3/CRS84) as a coordinate reference system.

Apply Requirement /req/core/crs84 for CRS84 support.

9.7. Encodings

While the OGC API — EDR standard does not specify any mandatory encoding, the following encodings are recommended. See Clause 2.2 (Optional Requirements Classes) for a discussion of this issue.

HTML encoding recommendation:

RECOMMENDATION 4
/rec/core/html
A: To support browsing an API definition through a web browser and to enable search engines to crawl and index the dataset, implementations SHOULD consider to support an HTML encoding.

GeoJSON encoding recommendation:

RECOMMENDATION 5
/rec/core/geojson
A: If the resource can be represented for the intended use in GeoJSON, implementations SHOULD consider to support GeoJSON as an encoding.

CoverageJSON encoding recommendation. This is specific to the EDR API:

RECOMMENDATION 6
/rec/core/covjson
A: If the resource can be represented for the intended use in CoverageJSON, implementations SHOULD consider to support CoverageJSON as an encoding.

Requirement `/req/core/http` implies that the encoding of a response is determined using content negotiation as specified by the HTTP RFC.

The section Media Types includes guidance on media types for encodings that are specified in this document.

Note that any API that supports multiple encodings will have to support a mechanism to create encoding-specific URIs for resources in order to express links, for example, to alternative representations of the same resource. This document does not mandate any particular approach to how this is supported by the API.

As clients simply need to dereference the URI of the link, the implementation details and the mechanism of how the encoding is included in the URI of the link are not important. Developers interested in the approach of a particular implementation, can study the API definition.

NOTE: Two common approaches are:

- an additional path for each encoding of each resource (this can be expressed, for example, using format specific suffixes like `.html`);
- an additional query parameter (for example, `accept` or `f`) that overrides the Accept header of the HTTP request.

9.8. Link Headers

RECOMMENDATION 7
/rec/core/link-header
A: Links included in the payloads of responses SHOULD also be included as `Link` headers in the HTTP response according to RFC 8288, Clause 3.
B: This recommendation does not apply, if there are a large number of links included in a response or a link is not known when the HTTP headers of the response are created.

9.9. OpenAPI 3.0

9.9.1. Basic requirements

Apply the OpenAPI 3.0 Requirements Class.

The OpenAPI 3.0 Requirements Class used in OGC API — Common is applicable to the EDR API as well. So an implementation of EDR API which supports OpenAPI 3.0 as an API Description

format must also comply with the OpenAPI 3.0 Requirements Class (http://www.opengis.net/spec/ogcapi-common-1/1.0/req/oas30) specified in OGC API — Common.

Apply Requirement /req/oas30/oas-definition-2 for OpenAPI 3.0 conformance.

Implementations must also advertise conformance with this Requirements Class.

Apply Requirement /req/oas30/oas-impl for OpenAPI 3.0 implementation.

An example OpenAPI definition document is available at http://schemas.opengis.net/ogcapi/edr/1.0/openapi/EDR_OpenAPI.yaml

9.9.2. Complete definition

Apply Requirement /req/oas30/completeness for OpenAPI 3.0 Completeness.

Note, for example, that APIs which are access-controlled (see Security), support web cache validation, CORS or that use HTTP redirection will make use of additional HTTP status codes beyond regular codes such as `200` for successful GET requests and `400`, `404` or `500` for error situations. See Clause 9.2.

Clients have to be prepared to receive responses not documented in the OpenAPI definition. For example, additional errors may occur in the transport layer outside of the server.

9.9.3. Exceptions

Apply Requirement /req/oas30/exceptions-codes for OpenAPI 3.0 Exception codes.

Example — An exception response object definition:

```
description: An error occurred.
content:
  application/json:
    schema:
      $ref: http://schemas.opengis.net/ogcapi/edr/1.0/openapi/schemas/
exception.yaml
  text/html:
    schema:
      type: string
```

9.10. Security

The OGC API — Common — Part 1: Core specification does not mandate any specific security controls. However, it was constructed so that security controls can be added without impacting conformance.

Apply Requirement /req/oas30/security for OpenAPI 3.0 Security support.

The OpenAPI specification currently supports the following security schemes:

- HTTP authentication,
- an API key (either as a header or as a query parameter),
- OAuth2's common flows (implicit, password, application and access code) as defined in RFC6749, and
- OpenID Connect Discovery.

A

ANNEX A (INFORMATIVE) REQUIREMENTS DETAIL

A

ANNEX A (INFORMATIVE) REQUIREMENTS DETAIL

A.1. Introduction

For clarity, the complete requirements class descriptions are omitted in the body of this specification. This annex contains the complete requirements classes.

A.2. Requirements Class "Core" in Detail

A.2.1. Requirements Class: OGC API — Environmental Data Retrieval Core

REQUIREMENTS CLASS: OGC API — ENVIRONMENTAL DATA RETRIEVAL CORE REQUIREMENTS CLASS	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/core	
Obligation	requirement
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
Dependency	http://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections
Requirement A.1	/req/core/root-op
Requirement A.2	/req/core/root-success
Requirement A.3	/req/core/api-definition-op
Requirement A.4	/req/core/api-definition-success

REQUIREMENTS CLASS: OGC API — ENVIRONMENTAL DATA RETRIEVAL CORE REQUIREMENTS CLASS	
Requirement A.5	/req/core/conformance
Requirement A.6	/req/core/conformance-success
Requirement A.7	/req/core/rc-bbox-definition
Requirement A.8	/req/core/rc-bbox-response
Requirement A.9	/req/edr/coords-definition
Requirement A.10	/req/edr/coords-response
Requirement A.11	/req/core/datetime-definition
Requirement A.12	/req/core/datetime-response
Requirement A.13	/req/edr/REQ_rc-parameter-name-definition
Requirement A.14	/req/edr/parameter-name-response
Requirement A.15	/req/edr/REQ_rc-crs-definition
Requirement A.16	/req/edr/REQ_rc-crs-response
Requirement A.17	/req/edr/rc-f-definition
Requirement A.18	/req/edr/REQ_rc-f-response
Requirement A.19	/req/edr/z-definition
Requirement A.20	/req/edr/z-response
Requirement A.21	/req/edr/within-definition
Requirement A.22	/req/edr/REQ_rc-within-response
Requirement A.23	/req/edr/within-units-definition
Requirement A.24	/req/edr/REQ_rc-within-units-response
Requirement A.25	/req/edr/resolution-x-definition
Requirement A.26	/req/edr/resolution-x-response
Requirement A.27	/req/edr/cube-z-response
Requirement A.28	/req/edr/resolution-y-definition

REQUIREMENTS CLASS: OGC API — ENVIRONMENTAL DATA RETRIEVAL CORE REQUIREMENTS CLASS	
Requirement A.29	/req/edr/resolution-y-response
Requirement A.30	/req/edr/resolution-z-definition
Requirement A.31	/req/edr/resolution-z-response
Requirement A.32	/req/edr/REQ_rc-corridor-height-definition
Requirement A.33	/req/edr/REQ_rc-corridor-height-response
Requirement A.34	/req/edr/REQ_rc-height-units-definition
Requirement A.35	/req/edr/height-units-response
Requirement A.36	/req/edr/corridor-width-definition
Requirement A.37	/req/edr/REQ_rc-corridor-width-response
Requirement A.38	/req/edr/REQ_rc-width-units-definition
Requirement A.39	/req/edr/width-units-response
Requirement A.40	/req/core/http
Requirement A.41	/req/core/crs84

REQUIREMENT A.1
/req/core/root-op
A: The server SHALL support the HTTP GET operation at the path `/`.

REQUIREMENT A.2
/req/core/root-success
A: A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B: The content of that response SHALL be based upon the OpenAPI 3.0 schema landingPage.yaml and include at least links to the following resources: • the API definition (relation type `service-desc` or `service-doc`)

REQUIREMENT A.2

- `/conformance` (relation type `conformance`)
- `/collections` (relation type `data`)

REQUIREMENT A.3

/req/core/api-definition-op

A:
The server SHALL support the HTTP GET operation on all links from the landing page which have the relation type `service-desc`.

B:
The server SHALL support the HTTP GET operation on all links from the landing page which have the relation type `service-doc`.

C:
The responses to all HTTP GET requests issued in `A` and `B` SHALL satisfy requirement /req/core/api-definition-success.

REQUIREMENT A.4

/req/core/api-definition-success

A:
A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.

B:
The content of that response SHALL be an API Definition document.

C:
The API Definition document SHALL shall be consistent with the media type identified through HTTP content negotiation.

Note:
The `f` parameter MAY be used to satisfy this requirement.

A.2.2. Requirement /req/core/conformance Core conformance classes

REQUIREMENT A.5
/req/core/conformance
The list of Conformance Classes advertised by the API SHALL include: **A:** http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/core
B: http://www.opengis.net/spec/ogcapi-common-2/1.0/conf/collections
C: http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core

REQUIREMENT A.6
/req/core/conformance-success
A: A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B: The content of that response SHALL be based upon the schema confClasses.yaml and list all OGC API conformance classes that the API conforms to.

A.2.3. Requirement /req/core/rc-bbox-definition Parameter bbox definition

REQUIREMENT A.7
/req/core/rc-bbox-definition

A:
The operation SHALL support a parameter `bbox` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
`name: bbox`
`in: query`
`required: false`
`schema:`
`  type: array`

REQUIREMENT A.7

```
  oneOf:
  - minItems: 4
    maxItems: 4
  - minItems: 6
    maxItems: 6
  items:
    type: number
style: form
explode: false
```

A.2.4. Requirement /req/core/rc-bbox-response Parameter bbox response

REQUIREMENT A.8

/req/core/rc-bbox-response

A:
Only features that have a spatial geometry that intersects the bounding box SHALL be part of the result set, if the `bbox` parameter is provided.

B:
If a feature has multiple spatial geometry properties, it is the decision of the server whether only a single spatial geometry property is used to determine the extent or all relevant geometries.

C:
The `bbox` parameter SHALL match all features in the collection that are not associated with a spatial geometry, too.

D:
The bounding box is provided as four or six numbers, depending on whether the coordinate reference system includes a vertical axis (height or depth):

- Lower left corner, coordinate axis 1
- Lower left corner, coordinate axis 2
- Minimum value, coordinate axis 3 (optional)
- Upper right corner, coordinate axis 1
- Upper right corner, coordinate axis 2
- Maximum value, coordinate axis 3 (optional)

REQUIREMENT A.8

E:
The bounding box SHALL consist of four numbers and the coordinate reference system of the values SHALL be interpreted as the coordinate reference system that is specified in a parameter `crs`.

F:
If the `crs` query parameter **is not** defined, the bounding box SHALL consist of four numbers and the coordinate reference system of the values SHALL be interpreted as the default coordinate reference system specified for the query type.

G:
If the `crs` query parameter **is not** defined **and** a default `crs` **is not** defined for the query, the bounding box SHALL consist of four numbers and the coordinate reference system of the values SHALL be interpreted as WGS 84 longitude/latitude (http://www.opengis.net/def/crs/OGC/1.3/CRS84).

H:
The coordinate values SHALL be within the extent specified for the coordinate reference system.

A.2.5. Requirement /req/edr/coords-definition Parameter coords definition

REQUIREMENT A.9

/req/edr/coords-definition

A:
Each geometry based resource (Position, Radius, Area, Cube , Trajectory, Corridor) collection operation SHALL support a parameter `coords` with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```
name: coords
in: query
required: true
schema:
  type: string
style: form
explode: false
```

B:

REQUIREMENT A.9

The `coords` string value will be a Well Known Text of representation geometry as defined in Simple Feature Access — Part 1: Common Architecture. The representation type will depend on the queryType of the API

A.2.6. Requirement /req/edr/coords-response Parameter coords response

REQUIREMENT A.10

/req/edr/coords-response

A:
Only those resources that have a spatial geometry that intersects the area defined by the `coords` parameter SHALL be part of the result set.

B:
The coordinates SHALL consist of a Well Known Text (WKT) geometry string.

C:
The coordinate reference system of the values SHALL be interpreted as WGS84 longitude/latitude

WKT: GEOGCS["WGS 84", DATUM["WGS_1984", SPHEROID["WGS 84", 6378137, 298.257223563, AUTHORITY["EPSG", "7030"]], AUTHORITY["EPSG", "6326"]], PRIMEM["Greenwich", 0 , AUTHORITY["EPSG", "8901"]], UNIT["degree", 0. 01745329251994328, AUTHORITY["EPSG", "9122"]], AUTHORITY["EPSG", "4326"]]

unless a different coordinate reference system is specified in a parameter `crs`.

A.2.7. Requirement /req/core/datetime-definition datetime definition

REQUIREMENT A.11

/req/core/datetime-definition

A:
The `datetime` parameter SHALL have the following characteristics (using an OpenAPI Specification 3.0 fragment):

```
name: datetime
in: query
required: false
schema:
  type: string
style: form
```

REQUIREMENT A.11

```
explode: false
```

A.2.8. Requirement /req/core/datetime-response datetime response

REQUIREMENT A.12

/req/core/datetime-response

A:
If the `datetime` parameter is provided, only resources that have a temporal geometry that intersects the temporal information in the `datetime` parameter SHALL be part of the result set.

B:
If a resource has multiple temporal properties, the API implementor decides whether only a single temporal property is used to determine the extent or all relevant temporal properties.

C:
The `datetime` parameter SHALL match all resources in the collection that are not associated with a temporal geometry.

D:
The temporal information is either a date-time or a time interval. The parameter value SHALL conform to the following syntax (using ABNF):

```
interval-closed     = date-time "/" date-time
interval-open-start = "../" date-time
interval-open-end   = date-time "/.."
interval            = interval-closed / interval-open-start / interval-open-
end
datetime            = date-time / interval
```

E:
The syntax of `date-time` is specified by RFC 3339, 5.6.

F:
Open ranges in time intervals at the start or end SHALL be supported using a double-dot (`..`).

A.2.9. Requirement /req/edr/REQ_rc-parameter-name-definition Parameter parameter-name definition

REQUIREMENT A.13

/req/edr/REQ_rc-parameter-name-definition

A:
Each resource collection operation SHALL support a parameter `parameter-name` with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```
name: parameter-name
in: query
required: true
explode: false
schema:
    minItems: 1
    type: array
    items:
        type: string
```

A.2.10. Requirement /req/edr/parameter-name-response Parameter parameter-name response

REQUIREMENT A.14

/req/edr/parameter-name-response

A:
If the `parameter-name` parameter is provided, only those parameters named SHALL be returned. If the `parameter-name` parameter is not specified all parameters in the collection SHALL be returned.

B:
The `parameter-name` parameter SHALL consist of a comma delimited string value based on an enumerated list of options listed in the collections metadata.

A.2.11. Requirement /req/edr/REQ_rc-crs-definition Parameter crs definition

REQUIREMENT A.15

/req/edr/REQ_rc-crs-definition

A:

REQUIREMENT A.15

Each resource collection operation SHALL support a parameter `crs` with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```
name: crs
in: query
required: false
schema:
  type: string
style: form
explode: false
```

A.2.12. Requirement /req/edr/REQ_rc-crs-response Parameter crs response

REQUIREMENT A.16

/req/edr/REQ_rc-crs-response

A:
If the `crs` parameter is provided, the returned information should be reprojected (if required) to the defined coordinate system. If the `crs` parameter is not specifed the data will be returned in its native projection.

B:
The `crs` parameter SHALL consist of an identifier selected from the enumerated list of valid values supplied in the collections metadata.

C:
If an unsupported `crs` value is requested a `400` error message SHOULD be returned.

A.2.13. Requirement /req/edr/rc-f-definition Parameter `f` definition

REQUIREMENT A.17

/req/edr/rc-f-definition

A:
Each resource collection operation SHALL support a parameter `f` with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```
name: f
in: query
required: false
schema:
  type: string
style: form
```

REQUIREMENT A.17

```
explode: false
```

A.2.14. Requirement /req/edr/REQ_rc-f-response Parameter `f` response

REQUIREMENT A.18

/req/edr/REQ_rc-f-response

A:
If the `f` parameter is provided, the returned information should be transformed to the defined data format.

B:
The `f` parameter SHALL consist of a string value based on an enumerated list of available options provided in the collections metadata.

C:
If an unsupported `f` value is requested a `400` error message should be returned.

A.2.15. Requirement /req/edr/z-definition Parameter z definition

REQUIREMENT A.19

/req/edr/z-definition

A:
Each resource collection operation MAY support a parameter `z` with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```
name: z
in: query
required: false
schema:
  type: string
style: form
explode: false
```

A.2.16. Requirement /req/edr/z-response Parameter z response

REQUIREMENT A.20

/req/edr/z-response

REQUIREMENT A.20

A:
If the z parameter is provided, only resources that have a vertical geometry that intersects the vertical information in the z parameter SHALL be part of the result set.

B:
The z can be defined as a height range by specifying a min-level and max-level seperated by a forward slash "/"

C:
A list of z can be defined be specifying a comma delimited list of values level1, level2, level3 etc

D:
An Arithmetic sequence using Recurring height intervals can be specified by **R**`number of intervals/min height/height interval`

E:
If the z parameter is not provided, the server SHOULD return data at all available vertical levels

```
single-level  = level
interval-closed  = min-level "/" max-level
level-list  = level1 "," level2 "," level3
repeating-interval = "R"number of intervals "/" min-level "/" height to
 increment by
```

```
Single level at level 850
z=850

All data between levels 100 and 550
z=100/550

Data at levels 10,80,100
z=10,80,200

Data at 20 levels at 50 unit intervals starting a level 100
z=R20/100/50
```

A.2.17. Requirement /req/edr/within-definition Parameter within definition

REQUIREMENT A.21

/req/edr/within-definition

REQUIREMENT A.21

A:
Each resource collection operation MAY support a parameter `within`with the following characteristics (using an OpenAPI Specification 3.0 fragment):

B:
If the instance metadata does not provide `within-units` values the API SHALL NOT support `within` queries:

```
name: within
in: query
required: true
schema:
  type: string
style: form
explode: false
```

A.2.18. Requirement /req/edr/REQ_rc-within-response Parameter within response

REQUIREMENT A.22

/req/edr/REQ_rc-within-response

A:
If the `within` parameter is provided, all selected information within the specified radius SHALL be part of the result set.

B:
If a `within-units` parameter is not provided, a `400` error WILL be returned.

A.2.19. Requirement /req/edr/within-units-definition Parameter within-units definition

REQUIREMENT A.23

/req/edr/within-units-definition

A:
Each resource collection operation MAY support a parameter `within-units` with the following characteristics (using an OpenAPI Specification 3.0 fragment):

B:
A within-units value MUST be one of the values defined in the instance metadata:

REQUIREMENT A.23

```
name: within-units
in: query
required: false
schema:
  type: string
style: form
explode: false
```

A.2.20. Requirement /req/edr/REQ_rc-within-units-response Parameter within-units response

REQUIREMENT A.24

/req/edr/REQ_rc-within-units-response

A:
The `within-units` parameter defines the distance units of the `within` query parameter value.

A.2.21. Requirement /req/edr/resolution-x-definition Parameter resolution-x definition

REQUIREMENT A.25

/req/edr/resolution-x-definition

A:
Each resource collection operation MAY support a parameter `resolution-x` with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```
name: resolution-x
in: query
required: false
schema:
  type: string
style: form
explode: false
```

A.2.22. Requirement /req/edr/resolution-x-response Parameter resolution-x response

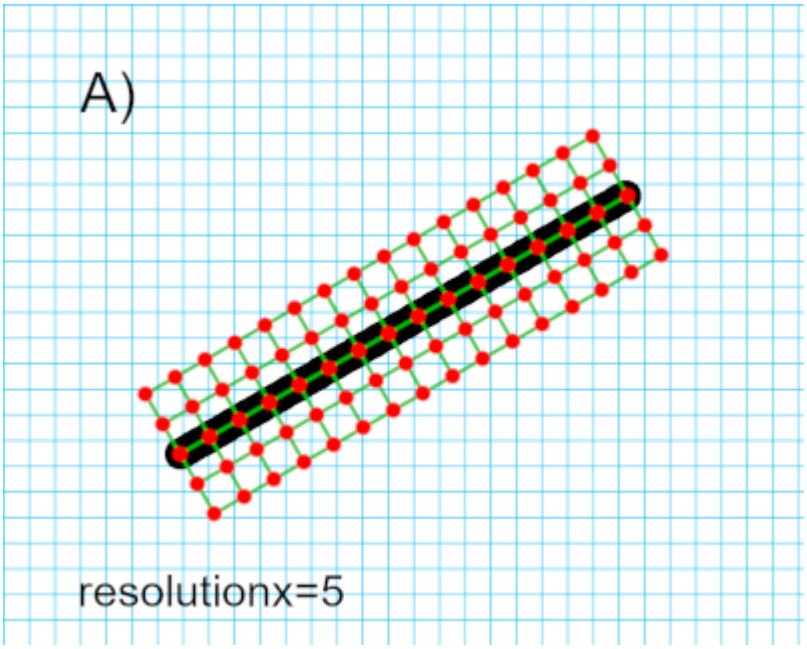
REQUIREMENT A.26

/req/edr/resolution-x-response

A:

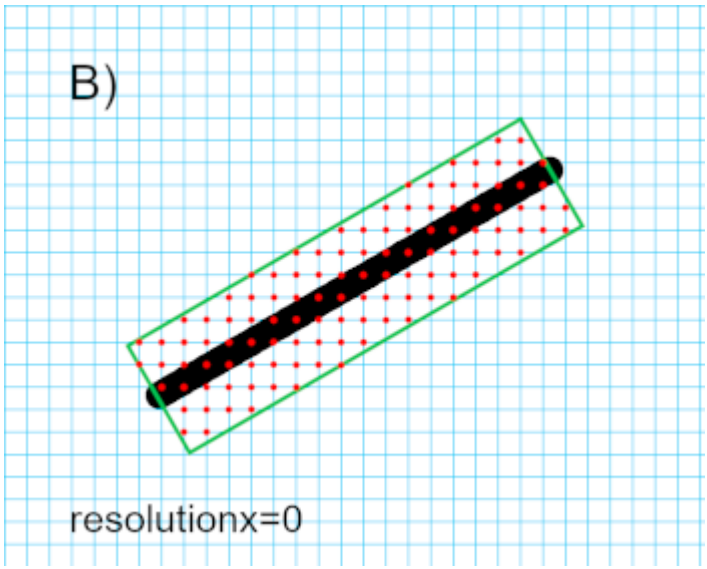
REQUIREMENT A.26

If the `resolution-x` parameter is provided, it denotes the number of positions to retrieve data for, across the width of the corridor path, including its minimum and maximum width coordinates.



B:

A `resolution-x` value of 0 SHALL return all available data at the stored resolution between (and including) the minimum and maximum coordinates of the defined corridor.



C:

REQUIREMENT A.26

If `resolution-x` is not specified, the API SHOULD return all available data at a resolution determined by the server, including the minimum and maximum coordinates of the defined corridor.
`resolution-x = number of intervals + 1`

A.2.23. Requirement /req/edr/cube-z-response Parameter z response for cube queries

REQUIREMENT A.27

/req/edr/cube-z-response

A:
If the `z` parameter is provided, only resources that have a vertical geometry that intersects the vertical information in the `z` parameter SHALL be part of the result set.

B:
The `z` can be defined as a height range by specifying a min-level and max-level seperated by a forward slash "/"

C:
A list of `z` can be defined by specifying a comma delimited list of values level1, level2, level3 etc

D:
An Arithmetic sequence using Recurring height intervals can be specified by **R**`number of intervals/min height/height interval`

E:
If the `z` parameter is not provided, the server SHOULD return data at all available vertical levels

A.2.24. Requirement /req/edr/resolution-y-definition Parameter resolution-y definition

REQUIREMENT A.28

/req/edr/resolution-y-definition

REQUIREMENT A.28

A:
Each resource collection operation MAY support a parameter `resolution-y` with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```
name: resolution-y
in: query
required: false
schema:
  type: string
style: form
explode: false
```

A.2.25. Requirement /req/edr/resolution-y-response Parameter resolution-y response

REQUIREMENT A.29

/req/edr/resolution-y-response

A:
If the `resolution-y` parameter is provided, denotes the number of intervals to retrieve data for along the path between the minimum and maximum y coordinates

B:
The total number of intervals includes the values for the minimum and maximum coordinates

C:
A `resolution-y` value of 0 MUST return all available data at the native y resolution between the minimum and maximum coordinates

D:
IF `resolution-y` is not specified, data should be returned for just the locations specified in the requested coordinates (**ONLY IF** interpolation is supported by the API)

```
resolution-y = number of intervals
```

A.2.26. Requirement /req/edr/resolution-z-definition Parameter resolution-z definition

REQUIREMENT A.30

/req/edr/resolution-z-definition

REQUIREMENT A.30

A:
Each resource collection operation MAY support a parameter `resolution-z` with the following characteristics (using an OpenAPI Specification 3.0 fragment):

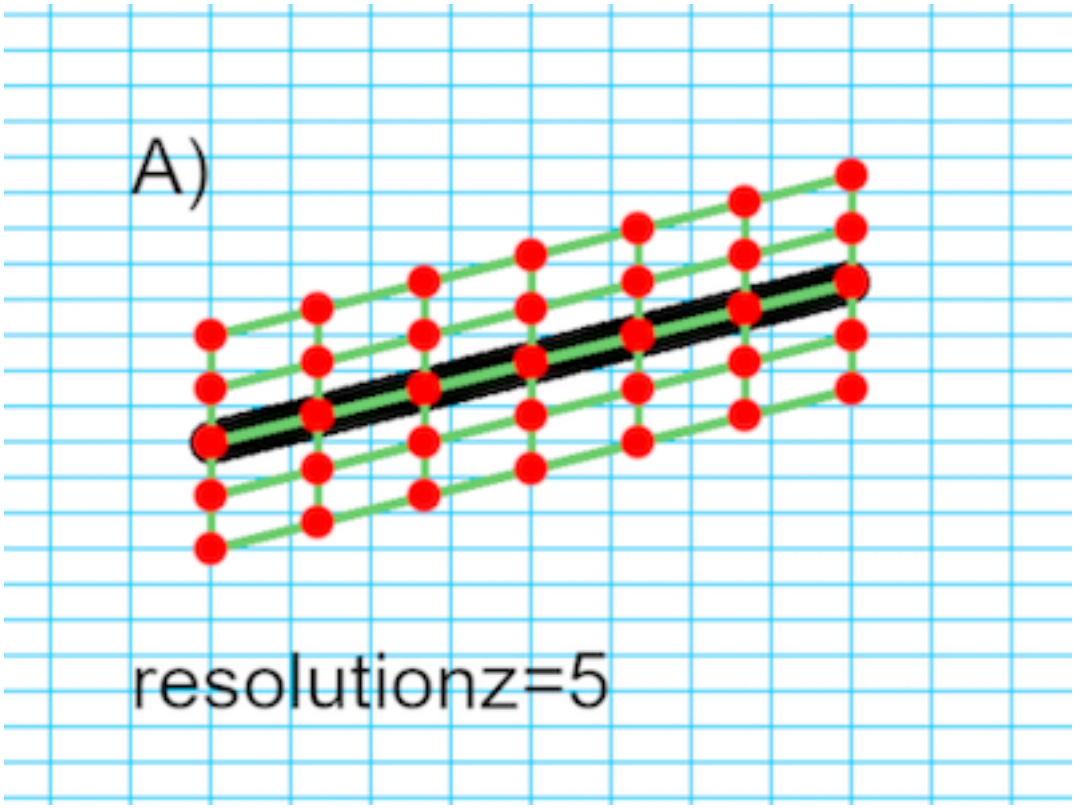
```
name: resolution-z
in: query
required: false
schema:
  type: string
style: form
explode: false
```

A.2.27. Requirement /req/edr/resolution-z-response Parameter resolution-z response

REQUIREMENT A.31

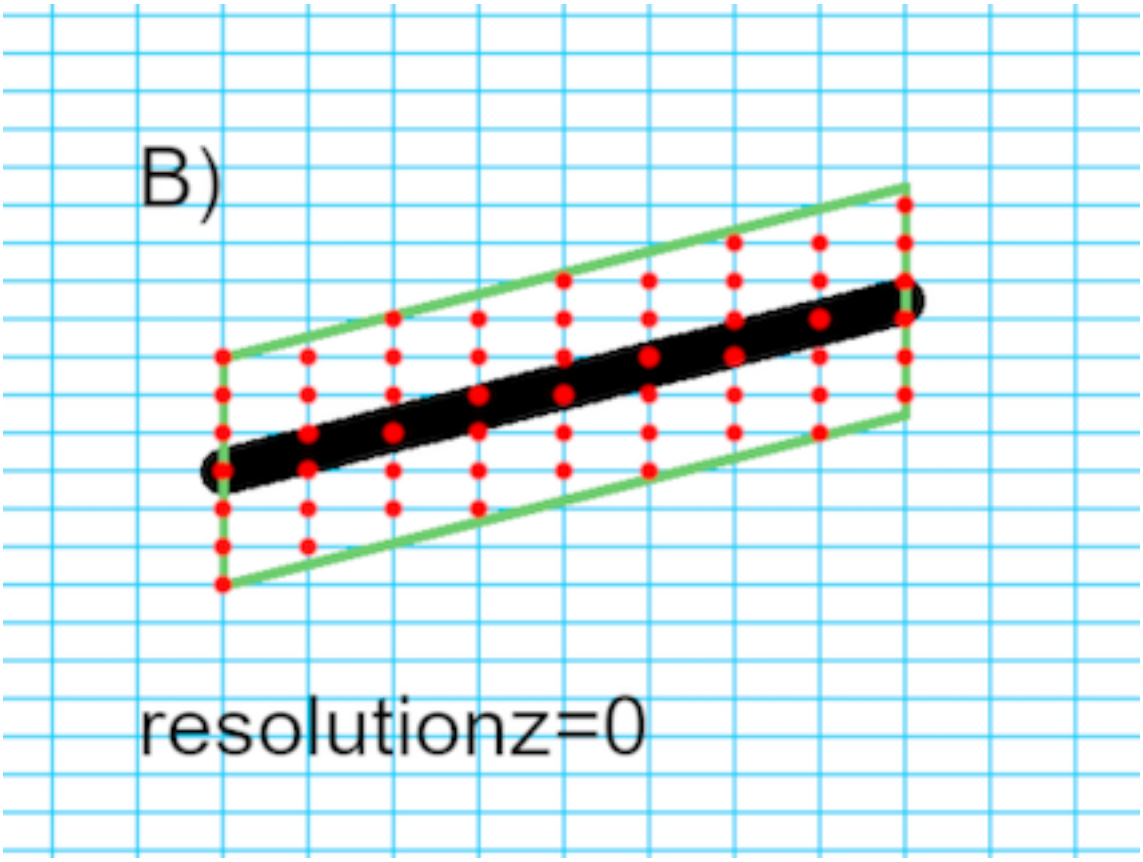
/req/edr/resolution-z-response

A:
If the `resolution-z` parameter is provided, it denotes the number of positions to retrieve data for, over the depth of the corridor path including its minimum and maximum width coordinates.



REQUIREMENT A.31

B:
A `resolution-z` value of 0 SHALL return all available data at the stored vertical resolution between (and including) the minimum and maximum coordinates of the defined corridor.



C:
If `resolution-z` is not specified the API SHOULD return all available data at a resolution determined by the server, including the minimum and maximum coordinates of the defined corridor.
`resolution-z = number of intervals + 1`

A.2.28. Requirement /req/edr/REQ_rc-corridor-height-definition Parameter corridor-height definition

REQUIREMENT A.32

/req/edr/REQ_rc-corridor-height-definition

A:
Each resource collection operation SHALL support a parameter `corridor-height` with the following characteristics (using an OpenAPI Specification 3.0 fragment):

REQUIREMENT A.32

```
name: corridor-height
in: query
required: true
schema:
  type: string
style: form
explode: false
```

A.2.29. Requirement /req/edr/REQ_rc-corridor-height-response Parameter corridor-height response

REQUIREMENT A.33

/req/edr/REQ_rc-corridor-height-response

A:
If the `corridor-height` parameter is defined the result set SHALL contain values derived from the chosen interpolation algorithm at the number of specifed intervals.
`corridor-height = height`

B:
The height of corridor parameter is the total height of the required corridor.

C:
The coordinates of the `coords` parameter define the centre point of the corridor.

D:
If an unsupported units value is requested a `400` error should be returned.

A.2.30. Requirement /req/edr/REQ_rc-height-units-definition Parameter height-units definition

REQUIREMENT A.34

/req/edr/REQ_rc-height-units-definition

A:
Each corridor resource collection operation SHALL support a parameter `height-units` with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```
name: height-units
in: query
required: true
```

REQUIREMENT A.34

```
schema:
  type: string
style: form
explode: false
```

A.2.31. Requirement /req/edr/height-units-response Parameter height-units response

REQUIREMENT A.35

/req/edr/height-units-response

A:
If the `height-units` parameter is defined the result set SHALL contain values derived based on the chosen units.
`height-units = units`

B:
If an unsupported units value is requested a `400` error should be returned.

A.2.32. Requirement /req/edr/corridor-width-definition Parameter corridor-width definition

REQUIREMENT A.36

/req/edr/corridor-width-definition

A:
Each resource collection operation SHALL support a parameter `corridor-width`with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```
name: corridor-width
in: query
required: true
schema:
  type: string
style: form
explode: false
```

A.2.33. Requirement /req/edr/REQ_rc-corridor-width-response Parameter corridor-width response

REQUIREMENT A.37

/req/edr/REQ_rc-corridor-width-response

A:
The `corridor-width` information is the total width of the required corridor.

B:
The supported `corridor-width` width units will be supplied by the query metatdata information.
`corridor-width = width`

C:
If the width value is the total width of the corridor.

D:
The coordinates of the `coords` parameter define the centre point of the corridor.

E:
If an unsupported units value is requested a `400` error should be returned.

A.2.34. Requirement /req/edr/REQ_rc-width-units-definition Parameter width-units definition

REQUIREMENT A.38

/req/edr/REQ_rc-width-units-definition

A:
Each corridor resource collection operation SHALL support a parameter `width-units` with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```
name: width-units
in: query
required: true
schema:
  type: string
style: form
```

REQUIREMENT A.38

`explode: false`

A.2.35. Requirement /req/edr/width-units-response Parameter width-units response

REQUIREMENT A.39

/req/edr/width-units-response

A:
If the `width-units` parameter is defined the result set SHALL contain values derived based on the chosen units.
`width-units = units`

B:
If an unsupported units value is requested a `400` error should be returned.

A.2.36. Requirement /req/core/http HTTP

REQUIREMENT A.40

/req/core/http

A:
The API SHALL conform to HTTP 1.1.

B:
If the API supports HTTPS, then the API SHALL also conform to HTTP over TLS.

A.2.37. Requirement /req/core/crs84 CRS84

REQUIREMENT A.41

/req/core/crs84

A:
Unless the client explicitly requests a different coordinate reference system, all spatial geometries SHALL be in the CRS84 (WGS 84 longitude/latitude) coordinate reference system

REQUIREMENT A.41
for geometries without height information and CRS84h (WGS 84 longitude/latitude plus ellipsoidal height) for geometries with height information.

A.3. Requirements Class "Collections" in Detail

A.3.1. Requirements Class: Collections

REQUIREMENTS CLASS: COLLECTIONS REQUIREMENTS CLASS	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/collections	
Obligation	requirement
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
Dependency	ISO 19107
Dependency	ISO 19108
Dependency	ISO 19111
Dependency	ISO 19108
Dependency	ISO 8601
Requirement A.42	/req/collections/rc-md-op
Requirement A.43	/req/collections/rc-md-success
Requirement A.44	/req/collections/src-md-op
Requirement A.45	/req/collections/src-md-success
Requirement A.46	/req/edr/rc-collection-info
Requirement A.47	/req/core/rc-collection-info-links
Requirement A.48	/req/core/rc-extent

REQUIREMENTS CLASS: COLLECTIONS REQUIREMENTS CLASS	
Requirement A.49	/req/core/rc-md-query-links
Requirement A.50	/req/edr/rc-crs
Requirement A.51	/req/edr/rc-parameters

REQUIREMENT A.42
/req/collections/rc-md-op
A: The API SHALL support the HTTP GET operation at the path `/collections`.

REQUIREMENT A.43
/req/collections/rc-md-success
A: A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B: The content of that response SHALL be based upon the schema collections.yaml.

REQUIREMENT A.44
/req/collections/src-md-op
A: The API SHALL support the HTTP GET operation at the path `/collections/{collectionId}`.
B: The parameter `collectionId` is each `id` property in the resource collections response (JSONPath: `$.collections[*].id`).

REQUIREMENT A.45
/req/collections/src-md-success
A: A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.

REQUIREMENT A.45

B:
The content of that response SHALL be based upon the schema collection.yaml.

C:
The content of that response SHALL be consistent with the content for this resource collection in the `/collections` response. That is, the values for `id`, `title`, `description` and `extent` SHALL be identical.

REQUIREMENT A.46

/req/edr/rc-collection-info

A:
Every Collection within a collections array MUST have an unique (within the array) `id` parameter.

B:
Every Collection within a collections array SHOULD have a `title` parameter.

C:
Every Collection within a collections array SHOULD have a `description` parameter.

D:
Every Collection within a collections array SHOULD have a `Keywords` parameter containing an array of values with describe the collection.

E:
Every Collection within a collections array MUST have a `links` parameter which must comply with the requirement `/req/core/rc-collection-info-links`.

F:
Every Collection within a collections array MUST have an `extent` parameter which must comply with the requirement `/req/core/rc-extent`.

G:
Every Collection within a collections array MUST have a `crs` parameter which must comply with the requirement `/req/edr/rc-crs`.

REQUIREMENT A.46

H:
Every Collection within a collections array MAY have a `distanceunits` parameter containing an array of supported distance units.

I:
If the `links` parameter includes a link to a `Radius` query there MUST be a `distanceunits` parameter

J:
Every Collection within a collections array MUST have an `f` parameter containing an array of values with describe the output formats supported by the collection.

K:
Every Collection within a collections array MUST have an `parameters` parameter containing a list of parameters must comply with the requirement `/req/edr/rc-parameters`.

REQUIREMENT A.47

/req/core/rc-collection-info-links

A:
A `200`-response SHALL include the following links in the `links` property of the response:

- a link to this response document (relation: `self`),
- a link to the response document in every other media type supported by the server (relation: `alternate`).
- at least one link to a query end point.

B:
All links SHALL include the `rel` and `type` link parameters.

REQUIREMENT A.48

/req/core/rc-extent

A:
For each spatial resource collection, the `extent` property, if provided, SHALL provide bounding boxes that include all spatial geometries and time intervals that include all temporal geometries in this collection. The temporal extent may use `null` values to indicate an open time interval.

REQUIREMENT A.48

B:
If a spatial resource has multiple properties with spatial or temporal information, it is the decision of the API implementation whether only a single spatial or temporal geometry property is used to determine the extent or all relevant geometries.

REQUIREMENT A.49

/req/core/rc-md-query-links

A:
For each collection included in the response, the `links` property of the collection SHALL include at least one link to a query resource (relation: `data`)

or an instance resource (relation: `instance`).

B:
All links SHALL include the `rel` and `type` link parameters.

REQUIREMENT A.50

/req/edr/rc-crs

A:
A crs object MUST have a unique (to the collection) `name` property, it MAY be an EPSG code.

B:
A crs object MUST have a `wkt` property which MUST be a correctly structured Well Known Text definition for the CRS.

C:
A crs object MAY have a `proj4` property which MAY be a correctly structured proj4 definition for the CRS (and MUST be equivalent to the wkt property).

REQUIREMENT A.51

/req/edr/rc-parameters

A:
A parameter object MAY have any number of members (name/value pairs).

REQUIREMENT A.51

B:
A parameter object MUST have a member with the name “type” and the value “Parameter”.

C:
A parameter object MAY have a member with the name “id” where the value MUST be a string and SHOULD be a common identifier.

D:
A parameter object MAY have a member with the name “label” where the value MUST be an i18n object that is the name of the parameter and which SHOULD be short. Note that this SHOULD be left out if it would be identical to the “label” of the “observedProperty” member.

E:
A parameter object MAY have a member with the name “description” where the value MUST be an i18n object which is a, perhaps lengthy, textual description of the parameter.

F:
A parameter object MUST have a member with the name “observedProperty” where the value is an object which MUST have the members “label” and “id” and which MAY have the members “description”, and “categories”. The value of “label” MUST be an i18n object that is the name of the observed property and which SHOULD be short. The value of “id” MUST be a string and SHOULD be a common identifier. If given, the value of “description” MUST be an i18n object with a textual description of the observed property. If given, the value of “categories” MUST be a non-empty array of category objects. A category object MUST an “id” and a “label” member, and MAY have a “description” member. The value of “id” MUST be a valid URI string and SHOULD resolve to more detailed information. The value of “label” MUST be an i18n object of the name of the category and SHOULD be short. If given, the value of “description” MUST be an i18n object with a textual description of the category.

G:
A parameter object MAY have a member with the name “categoryEncoding” where the value is an object where each key is equal to an “id” value of the “categories” array within the “observedProperty” member of the parameter object. There MUST be no duplicate keys. The value is either an integer or an array of integers where each integer MUST be unique within the object.

H:

REQUIREMENT A.51
A parameter object MAY have a member with the name “unit” where the value is an object which MUST have either or both the members “label” or/and “symbol”, and which MAY have the member “id”. If given, the value of “symbol” MUST either be a string of the symbolic notation of the unit, or an object with the members “value” and “type” where “value” is the symbolic unit notation and “type” references the unit serialization scheme that is used. “type” MUST HAVE the value “http://www.opengis.net/def/uom/UCUM/” if UCUM is used, or a custom value as recommended in section “Extensions”. If given, the value of “label” MUST be an i18n object of the name of the unit and SHOULD be short. If given, the value of “id” MUST be a string and SHOULD be a common identifier. It is RECOMMENDED to reference a unit serialization scheme to allow automatic unit conversion.

I: A parameter object MUST NOT have a “unit” member if the “observedProperty” member has a “categories” member.

A.4. Requirements Class “Queries” for Position, Area, Cube, Trajectory, Corridor, Items, Locations, and Instances

A.4.1. Requirements Class: Queries

REQUIREMENTS CLASS: QUERIES REQUIREMENTS CLASS	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/queries	
Obligation	requirement
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/collections
Requirement A.52	/req/queries/position
Requirement A.53	/req/edr/rc-area
Requirement A.54	/req/edr/rc-cube
Requirement A.55	/req/edr/rc-trajectory
Requirement A.56	/req/edr/rc-corridor

REQUIREMENTS CLASS: QUERIES REQUIREMENTS CLASS	
Requirement A.57	/req/edr/rc-items
Requirement A.58	/req/edr/rc-locations
Requirement A.59	/req/instances/rc-md-op
Requirement A.60	/req/instances/rc-md-success
Requirement A.61	/req/instances/src-md-op
Requirement A.62	/req/instances/src-md-success

A.4.2. Requirements for Position Queries

REQUIREMENT A.52
/req/queries/position
A: For every collection identified in the feature collections response (path `/collections`), the server MAY support the HTTP GET operation at the path `/collections/{collectionId}/position`.
B: The parameter `collectionId` is each `id` property in the collections response (JSONPath: `$.collections[*].id`).
C: A `position` GET operation MUST include a `coords` query parameter
D: If the `coords` query parameter is not specified a HTTP `400` error should be generated
E: A `position` GET operation MAY include a `z` query parameter
F: A `position` GET operation SHOULD include a `parameter-name` query parameter

REQUIREMENT A.52

G:
A `position` GET operation MAY include a `datetime` query parameter

H:
A `position` GET operation SHOULD include a `crs` query parameter

I:
A `position` GET operation SHOULD include a `f` query parameter

A.4.3. Requirements for Area Queries

REQUIREMENT A.53

/req/edr/rc-area

A:
For every collection identified in the feature collections response (path `/collections`), the server MAY support the HTTP GET operation at the path `/collections/{collectionId}/area`.

B:
The parameter `collectionId` is each `id` property in the collections response (JSONPath: `$.collections[*].id`).

C:
An `area` GET operation MUST include a `coords` query parameter

D:
If the `coords` query parameter is not specified a HTTP `400` error should be generated

E:
An `area` GET operation MAY include a `z` query parameter

F:

REQUIREMENT A.53

An `area` GET operation SHOULD include a `parameter-name` query parameter

G:
An `area` GET operation MAY include a `datetime` query parameter

H:
An `area` GET operation SHOULD include a `crs` query parameter

I:
An `area` GET operation SHOULD include a `f` query parameter

A.4.4. Requirements for Cube Queries

REQUIREMENT A.54

/req/edr/rc-cube

A:
For every collection identified in the collections response (path `/collections`), the server MAY support the HTTP GET operation at the path `/collections/{collectionId}/cube`.

B:
The parameter `collectionId` is each `id` property in the collections response (JSONPath: `$.collections[*].id`).

C:
A `cube` GET operation MUST include a `coords` query parameter

D:
If the `coords` query parameter is not specified a HTTP `400` error should be generated

E:
A `cube` GET operation MUST include a `min-z` query parameter

F:

REQUIREMENT A.54

If the `min-z` query parameter is not specified a HTTP `400` error should be generated

G:
A `cube` GET operation MUST include a `max-z` query parameter

H:
If the `max-z` query parameter is not specified a HTTP `400` error should be generated

I:
A `cube` GET operation SHOULD include a `parameter-name` query parameter

J:
A `cube` GET operation MAY include a `datetime` query parameter

K:
A `cube` GET operation SHOULD include a `crs` query parameter

L:
A `cube` GET operation SHOULD include a `f` query parameter

A.4.5. Requirements for Trajectory Queries

REQUIREMENT A.55

/req/edr/rc-trajectory

A:
For every collection identified in the collections response (path `/collections`), the server MAY support the HTTP GET operation at the path `/collections/{collectionId}/trajectory`.

B:
The parameter `collectionId` is each `id` property in the collections response (JSONPath: `$.collections[*].id`).

REQUIREMENT A.55

C:
A `trajectory` GET operation MUST include a `coords` query parameter

D:
If the `coords` query parameter is not specified a HTTP `400` error should be generated

E:
A `trajectory` GET operation MAY include a `z` query parameter

F:
A `trajectory` GET operation SHOULD include a `parameter-name` query parameter

G:
A `trajectory` GET operation MAY include a `datetime` query parameter

H:
A `trajectory` GET operation SHOULD include a `crs` query parameter

I:
A `trajectory` GET operation SHOULD include a `f` query parameter

A.4.6. Requirements for Corridor Queries

REQUIREMENT A.56

/req/edr/rc-corridor

A:
For every collection identified in the collections response (path `/collections`), the server MAY support the HTTP GET operation at the path `/collections/{collectionId}/corridor`.

B:
The parameter `collectionId` is each `id` property in the collections response (JSONPath: `$.collections[*].id`).

REQUIREMENT A.56

C:
A `corridor` GET operation MUST include a `coords` query parameter

D:
If the `coords` query parameter is not specified a HTTP `400` error should be generated

E:
A `corridor` GET operation MUST include a `corridor-width` query parameter

F:
If the `corridor-width` query parameter is not specified a HTTP `400` error should be generated

G:
A `corridor` GET operation MUST include a `corridor-height` query parameter

H:
If the `corridor-height` query parameter is not specified a HTTP `400` error should be generated

I:
A `corridor` GET operation MUST include a `width-units` query parameter

J:
If the `width-units` query parameter is not specified a HTTP `400` error should be generated

K:
If the `width-units` query parameter value is not one of the supported values a HTTP `400` error should be generated

L:
A `corridor` GET operation MUST include a `height-units` query parameter

REQUIREMENT A.56

M:
If the `height-units` query parameter is not specified a HTTP `400` error should be generated

N:
If the `height-units` query parameter value is not one of the supported values a HTTP `400` error should be generated

O:
A `corridor` GET operation MAY include a `z` query parameter

P:
A `corridor` GET operation SHOULD include a `parameter-name` query parameter

Q:
A `corridor` GET operation MAY include a `datetime` query parameter

R:
A `corridor` GET operation SHOULD include a `crs` query parameter

S:
A `corridor` GET operation SHOULD include a `f` query parameter

A.4.7. Requirements for Items Queries

REQUIREMENT A.57

/req/edr/rc-items

A:
For every collection identified in the collections response (path `/collections`), the server MAY support the HTTP GET operation at the path `/collections/{collectionId}/items`.

B:
The parameter `collectionId` is each `id` property in the feature collections response (JSONPath: `$.collections[*].id`).

A.4.8. Requirements for Locations Queries

REQUIREMENT A.58
/req/edr/rc-locations
A: For every collection identified in the collections response (path `/collections`), the server MAY support the HTTP GET operation at the path `/collections/{collectionId}/locations`.
B: The parameter `collectionId` is each `id` property in the feature collections response (JSONPath: `$.collections[*].id`).
C: If a `locationId` is not specified a list of valid locationId's MUST be returned with a description of their geo-spatial extent.
D: A `locations` GET operation SHOULD include a `parameter-name` query parameter
E: A `locations` GET operation MAY include a `datetime` query parameter
F: A `locations` GET operation SHOULD include a `crs` query parameter
G: A `locations` GET operation SHOULD include a `f` query parameter

A.4.9. Requirements for Instances Queries

REQUIREMENT A.59
/req/instances/rc-md-op

REQUIREMENT A.59

A:
The API MAY support the HTTP GET operation at the path `/collections/{collection_id}/instances`.

REQUIREMENT A.60

/req/instances/rc-md-success

A:
A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.

REQUIREMENT A.61

/req/instances/src-md-op

A:
The API SHALL support the HTTP GET operation at the path `/collections/{collectionId}/instances/{instanceId}`.

B:
The parameter `collectionId` is each `id` property in the resource collections response (JSONPath: `$.collections[*].id`) and `instanceId` is each id property of instances of the chosen collection.

REQUIREMENT A.62

/req/instances/src-md-success

A:
A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.

B:
The content of that response SHALL be based upon the JSON schema instances.yaml.

C:
The content of that response SHALL be consistent with the content for this resource collection in the `/collections` response. That is, the values for `id`, `title`, `description` and `extent` SHALL be identical.

A.5. Requirements Class “JSON” in Detail

A.5.1. Requirements Class: JSON

REQUIREMENTS CLASS: JSON REQUIREMENTS CLASS	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/json	
Obligation	requirement
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/json
Requirement A.63	/req/json/content
Requirement A.64	/req/json/definition

REQUIREMENT A.63
/req/json/content
A: Every `200`-response with the media type `application/json` SHALL be a payload encoded according to the JSON Interchange Format.
B: The links specified in the requirements `/req/core/rc-collection-info-links` and `/req/core/rc-collection-info-links` MAY be added in an extension property (foreign member) with the name `links`.
C: The parameters specified in the requirements `/req/edr/rc-parameters` MAY be added in an extension property (foreign member) with the name `parameters`.
D: The schema of all responses with the media type `application/json` SHALL conform with the JSON Schema specified for the resource in the Core requirements class.

REQUIREMENT A.64
/req/json/definition
A: `200`-responses of the server SHALL support the following media types: • `application/json` for all resources.

A.6. Requirements Class "GeoJSON" in Detail

A.6.1. Requirements Class: GeoJSON

REQUIREMENTS CLASS: GEOJSON REQUIREMENTS CLASS	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/geojson	
Obligation	requirement
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
Requirement A.65	/req/geojson/content
Requirement A.66	/req/geojson/definition

REQUIREMENT A.65
/req/geojson/content
A: Every `200`-response with the media type `application/geo+json` SHALL be • a GeoJSON FeatureCollection Object for features, and • a GeoJSON Feature Object for a single feature.
B: The links specified in the requirements `/req/core/rc-collection-info-links` and `/req/core/rc-collection-info-links` SHALL be added in an extension property (foreign member) with the name `links`.
C:

REQUIREMENT A.65
The parameters specified in the requirements `/req/edr/rc-parameters` MAY be added in an extension property (foreign member) with the name `parameters`.

D:
The schema of all responses with the media type `application/json` SHALL conform with the JSON Schema specified for the resource in the Core requirements class.

REQUIREMENT A.66
/req/geojson/definition

A:
`200`-responses of the server SHALL support the following media types:

- `application/geo+json` for resources that include feature content, and
- `application/json` for all other resources.

A.7. Requirements Class "EDR GeoJSON" in Detail

A.7.1. Requirements Class: EDR GeoJSON

REQUIREMENTS CLASS: EDR GEOJSON REQUIREMENTS CLASS	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/edr-geojson	
Obligation	requirement
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
Requirement A.67	/req/edr-geojson/content
Requirement A.68	/req/edr-geojson/definition

A.7.2. Requirement /req/edr-geojson/content

REQUIREMENT A.67
/req/edr-geojson/content
A: Every `200`-response with the media type `application/geo+json` SHALL be • a GeoJSON FeatureCollection Object for features, and • a GeoJSON Feature Object for a single feature.
B: The links specified in the requirements `/req/core/rc-collection-info-links` and `/req/core/rc-collection-info-links` SHALL be added in an extension property (foreign member) with the name `links`.
C: The parameters specified in the requirements `/req/edr/rc-parameters` MAY be added in an extension property (foreign member) with the name `parameters`.
D: The schema of all responses with the media type `application/json` SHALL conform with the JSON Schema specified for the resource in the Core requirements class.

REQUIREMENT A.68
/req/edr-geojson/definition
A: `200`-responses of the server SHALL support the following media types: • `application/geo+json` for resources that include feature content, and • `application/json` for all other resources.

A.8. Requirements Class “CoverageJSON” in Detail

A.8.1. Requirements Class: CoverageJSON

REQUIREMENTS CLASS: COVERAGEJSON REQUIREMENTS CLASS	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/covjson	
Obligation	requirement
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
Requirement A.69	/req/covjson/content
Requirement A.70	/req/covjson/definition

REQUIREMENT A.69
/req/covjson/content
A: Every `200`-response with the media type `application/prs.coverage+json` SHALL be • a CoverageJSON Object
B: The schema of all responses with the media type `application/prs.coverage+json` SHALL conform with the JSON Schema specified for the resource in the Core requirements class.

REQUIREMENT A.70
/req/covjson/definition
A: `200`-responses of the server SHALL support the following media types: • `application/prs.coverage+json` for resources that include data content, and • `application/json` for all other resources.

A.9. Requirements Class “HTML” in Detail

A.9.1. Requirements Class: HTML

REQUIREMENTS CLASS: HTML REQUIREMENTS CLASS	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/html	
Obligation	requirement
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
Requirement A.71	/req/html/content
Requirement A.72	/req/html/definition

REQUIREMENT A.71
/req/html/content
A: Every `200`-response of the server with the media type `text/html` SHALL be a HTML 5 document that includes the following information in the HTML body: • all information identified in the schemas of the Response Object in the HTML `<body>`, and • all links in HTML `<a>` elements in the HTML `<body>`.

REQUIREMENT A.72
/req/html/definition
A: Every `200`-response of an operation of the server SHALL support the media type `text/html`.

A.10. Requirements Class “OpenAPI 3.0” in Detail

A.10.1. Requirements Class: OpenAPI 3.0

REQUIREMENTS CLASS: OPENAPI 3.0 REQUIREMENTS CLASS	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/oas30	
Obligation	requirement
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/core
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/oas30
Dependency	OpenAPI Specification 3.0.3
Requirement A.73	/req/oas30/oas-impl
Requirement A.74	/req/oas30/oas-definition-1
Requirement A.75	/req/oas30/oas-definition-2
Requirement A.76	/req/oas30/completeness
Requirement A.77	/req/oas30/exceptions-codes
Requirement A.78	/req/oas30/security

A.10.2. Requirement /req/oas30/oas-impl OpenAPI 3.0 implementation

REQUIREMENT A.73
/req/oas30/oas-impl
A: The server SHALL implement all capabilities specified in the OpenAPI definition.

A.10.3. Requirement /req/oas30/oas-definition-1

REQUIREMENT A.74

/req/oas30/oas-definition-1

A:
The content of the response of the HTTP GET operation at the landing page SHALL include the following links to the API definition:

- relation type `service-desc` and content type `application/vnd.oai.openapi+json;version=3.0`,
- relation type `service-doc` and content type `text/html`.

A.10.4. Requirement /req/oas30/oas-definition-2 OpenAPI 3.0 conformance

REQUIREMENT A.75

/req/oas30/oas-definition-2

A:
The JSON representation SHALL conform to the OpenAPI Specification, version 3.0.

A.10.5. Requirement /req/oas30/completeness OpenAPI 3.0 Completeness

REQUIREMENT A.76

/req/oas30/completeness

A:
The OpenAPI definition SHALL specify for each operation all HTTP Status Codes and Response Objects that the API uses in responses.

B:
This includes the successful execution of an operation as well as all error situations that originate from the server.

A.10.6. Requirement /req/oas30/exceptions-codes OpenAPI 3.0 Exception codes

REQUIREMENT A.77
/req/oas30/exceptions-codes
A: For error situations that originate from an API server, the API definition SHALL cover all applicable HTTP Status Codes.

A.10.7. Requirement /req/oas30/security OpenAPI 3.0 Security

REQUIREMENT A.78
/req/oas30/security
A: For cases, where the operations of the API are access-controlled, the security scheme(s) and requirements SHALL be documented in the OpenAPI definition.

B

ANNEX B (INFORMATIVE) ABSTRACT TEST SUITE (NORMATIVE)

B

ANNEX B (INFORMATIVE) ABSTRACT TEST SUITE (NORMATIVE)

B.1. Introduction

The Abstract Test Suite (ATS) is a compendium of test assertions applicable to implementations of the EDR API. An ATS provides a basis for developing an Executable Test Suite to verify that the implementation under test conforms to all the relevant functional specifications.

The abstract test cases (assertions) are organized into test groups that correspond to distinct conformance test classes defined in the EDR API specification.

OGC APIs are not Web Services in the traditional sense. Rather, they define the behavior and content of a set of Resources exposed through a Web Application Programing Interface (Web API). Therefore, an API may expose resources in addition to those defined by the standard. A test engine must be able to traverse the API, identify and validate test points, and ignore resource paths which are not to be tested.

B.2. Conformance Class Core

Table B.1 — Conformance Class "Core"

Conformance Class	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core	
Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/core
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/core

B.2.1. General Tests

B.2.1.1. HTTP

Table B.2 — Abstract Test 1

Abstract Test 1	**/conf/core/http**
Test Purpose	Validate that the resource paths advertised through the API conform with HTTP 1.1 and, where approprate, TLS.
Requirement	/req/core/http
Test Method	a) All compliance tests shall be configured to use the HTTP 1.1 protocol exclusively. b) For APIs which support HTTPS, all compliance tests shall be configured to use HTTP over TLS (RFC 2818) with their HTTP 1.1 protocol.

B.2.2. Landing Page {root}/

Table B.3 — Abstract Test 2

Abstract Test 2	**/conf/core/root-op**
Test Purpose	Validate that a landing page can be retrieved from the expected location.
Requirement	/req/core/root-op
Test Method	a) Issue an HTTP GET request to the URL {root}/ b) Validate that a document was returned with a status code 200 c) Validate the contents of the returned document using test /conf/core/root-success.

Table B.4 — Abstract Test 3

Abstract Test 3	**/conf/core/root-success**

Test Purpose	Validate that the landing page complies with the require structure and contents.
Requirement	/req/core/root-success
Test Method	Validate the landing page for all supported media types using the resources and tests identified in Table B.5 For formats that require manual inspection, perform the following: 1. Validate that the landing page includes a “service-desc” and/or “service-doc” link to an API Definition 2. Validate that the landing page includes a “conformance” link to the conformance class declaration 3. Validate that the landing page includes a “data” link to the Feature contents.

The landing page may be retrieved in a number of different formats. The following table identifies the applicable schema document for each format and the test to be used to validate the landing page against that schema. All supported formats should be exercised.

Table B.5 — Schema and Tests for Landing Pages

FORMA	SCHEMA DOCUMENT	TEST ID
HTML	landingPage.yaml	/conf/html/content
JSON	landingPage.yaml	/conf/geojson/content

B.2.3. API Definition Path {root}/api (link)

Table B.6 — Abstract Test 4

Abstract Test 4	**/conf/core/api-definition**
Test Purpose	Validate that the API Definition document can be retrieved from the expected location.
Requirement	/req/core/api-definition-op
Test Purpose	Validate that the API Definition document can be retrieved from the expected location.
Test Method	a) Construct a path for each API Definition link on the landing page b) Issue a HTTP GET request on each path

c) Validate that a document was returned with a status code 200
d) Validate the contents of the returned document using test /conf/core/api-definition-success.

Table B.7 — Abstract Test 5

Abstract Test 5	**/conf/core/api-definition-success**
Test Purpose	Validate that the API Definition complies with the required structure and contents.
Requirement	/req/core/api-definition-success
Test Method	Validate the API Definition document against an appropriate schema document.

B.2.4. Conformance Path {root}/conformance

Table B.8 — Abstract Test 6

Abstract Test 6	**/conf/core/conformance**
Test Purpose	Validate that a Conformance Declaration can be retrieved from the expected location.
Requirement	/req/core/conformance
Test Method	a) Construct a path for each “conformance” link on the landing page as well as for the {root}/conformance path. b) Issue an HTTP GET request on each path c) Validate that a document was returned with a status code 200 d) Validate the contents of the returned document using test /req/core/conformance-success.

Table B.9 — Abstract Test 7

Abstract Test 7	**/conf/core/conformance-success**
Test Purpose	Validate that the Conformance Declaration response complies with the required structure and contents.
Requirement	/req/core/conformance-success

Test Method	a) Validate the response document against OpenAPI 3.0 schema confClasses.yaml b) Validate that the document includes the conformance class "http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core" c) Validate that the document lists all OGC API conformance classes that the API implements.

B.3. Conformance Class Collections

Table B.10 — Conformance Class "Collections"

Conformance Class	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/collections	
Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/collections
Dependency	http://www.opengis.net/spec/ogcapi-common-2/1.0/conf/collections

B.3.1. General Tests

B.3.1.1. CRS 84

Table B.11 — Abstract Test 8

Abstract Test 8	**/conf/core/crs84**
Test Purpose	Validate that all spatial geometries provided through the API are in the CRS84 spatial reference system unless otherwise requested by the client.
Requirement	/req/core/crs84
Test Method	a) Do not specify a coordinate reference system in any request. All spatial data should be in the CRS84 reference system.

	b) Validate retrieved spatial data using the CRS84 reference system.

B.3.2. Environmental Data Collections {root}/collections

Table B.12 — Abstract Test 9

Abstract Test 9	/conf/collections/rc-md-op
Test Purpose	Validate that information about the Collections can be retrieved from the expected location.
Requirement	/req/collections/rc-md-op
Test Method	a) Issue an HTTP GET request to the URL {root}/collections b) Validate that a document was returned with a status code 200 c) Validate the contents of the returned document using test /conf/rc-md-success.

Table B.13 — Abstract Test 10

Abstract Test 10	/conf/rc-md-success
Test Purpose	Validate that the Collections content complies with the required structure and contents.
Requirement	/req/collections/rc-md-success
Test Method	a) Validate that all response documents comply with /req/core/rc-collection-info-links b) In case the response includes a “crs” property, validate that it includes a valid Well Known Text definition c) Validate the collections content for all supported media types using the resources and tests identified in Table B.14

The Collections content may be retrieved in a number of different formats. The following table identifies the applicable schema document for each format and the test to be used to validate the against that schema. All supported formats should be exercised.

Table B.14 — Schema and Tests for Collections content

FORMA˙	SCHEMA DOCUMENT	TEST ID
HTML	collections.yaml	/conf/html/content

FORMA	SCHEMA DOCUMENT	TEST ID
JSON	collections.yaml	/conf/geojson/content

B.3.3. Environmental Data Collection {root}/collections/{collectionId}

Table B.15 — Abstract Test 11

Abstract Test 11	/conf/collections/src-md-op
Test Purpose	Validate that the Collection content can be retrieved from the expected location.
Requirement	/req/collections/src-md-op
Test Method	For every Feature Collection described in the Collections content, issue an HTTP GET request to the URL `/collections/{collectionId}` where `{collectionId}` is the `id` property for the collection. • Validate that a Collection was returned with a status code 200 • Validate the contents of the returned document using test /conf/collections/src-md-success.

Table B.16 — Abstract Test 12

Abstract Test 12	/conf/collections/src-md-success
Test Purpose	Validate that the Collection content complies with the required structure and contents.
Requirement	/req/collections/src-md-success
Test Method	Verify that the content of the response is consistent with the content for this Resource Collection in the `/collections` response. That is, the values for `id`, `title`, `description` and `extent` are identical.

B.3.4. Second Tier Collections Tests

These tests are invoked by other tests.

B.3.4.1. Collection Extent

Table B.17 — Abstract Test 13

Abstract Test 13	**/conf/core/rc-extent**
Test Purpose	Validate the `extent` property if it is present
Requirement	/req/core/rc-extent
Test Method	• Verify that the `extent` provides bounding boxes that include all spatial geometries • Verify that if the `extent` provides time intervals that they include all temporal geometries in this collection. • A temporal extent of `null` indicates an open time interval. • Verify that if the `extent` provides vertical intervals that they include all vertical geometries in this collection. • A vertical extent of `null` indicates an open vertical interval.

B.3.4.2. Collection Queries

Table B.18 — Abstract Test 14

Abstract Test 14	**/conf/edr/rc-collection-info**
Test Purpose	Validate that each collection provided by the server is described in the Collections Metadata.
Requirement	/req/edr/rc-collection-info
Test Method	a) Verify that all collections listed in the `collections` array of the Collections Metadata exist. b) Verify that each collection entry includes an identifier. c) Verify that each collection entry includes links in accordance with /conf/core/rc-collection-info-links. d) Verify that if the collection entry includes an `extent` property, the property complies with /conf/core/rc-extent e) Verify that if the collection data_queries entry includes a `crs` property, the property complies with /conf/edr/rc-crs f) Verify that if the collection entry includes an `parameter-names` property, the property complies with /req/edr/rc-parameters

	g) Validate each collection entry for all supported media types using the resources and tests identified in Table B.19

The collection entries may be encoded in a number of different formats. The following table identifies the applicable schema document for each format and the test to be used to validate the against that schema. All supported formats should be exercised.

Table B.19 — Schema and Tests for Collection Entries

FORMAT	SCHEMA DOCUMENT	TEST ID
HTML	collection.yaml	/conf/html/content
JSON	collection.yaml	/conf/json/content

Table B.20 — Abstract Test 15

Abstract Test 15	**/conf/edr/rc-md-query-links**
Test Purpose	Validate that each Collection metadata entry in the Collections Metadata document includes all required links.
Requirement	/req/core/rc-md-query-links
Test Method	a) Verify that each Collection item in the Collections Metadata document includes a `link` property for each supported encoding. b) Verify that the `links` properties of the collection includes an item for each supported encoding with at least one link to either a query resource (relation: `data`) or instance resource (relation: `collection`). c) Verify that all links include the `rel` and `type` link parameters.

B.3.4.3. Collection Links

Table B.21 — Abstract Test 16

Abstract Test 16	**/conf/core/rc-collection-info-links**
Test Purpose	Validate that the required links are included in the Collections Metadata document.
Requirement	/req/core/rc-collection-info-links

Test Method	Verify that the response document includes: a) a link to this response document (relation: `self`), b) a link to the response document in every other media type supported by the server (relation: `alternate`). Verify that all links include the `rel` and `type` link parameters.

B.3.4.4. Collection Parameters

Table B.22 — Abstract Test 17

Abstract Test 17	**/conf/edr/rc-parameters**
Test Purpose	Validate that each parameter in a collection is correctly defined.
Requirement	/req/edr/rc-parameters
Test Method	a) Verify that all parameters listed in a `collection` have the required properties. b) Verify that each parameter property has a unique name (in the collection). c) Verify that each parameter property has a `type` property. d) Verify that each parameter property has a `observedProperty` property. e) Verify that the `observedProperty` property has a `label` property with correctly defined i18n compliant values. f) Verify that the `observedProperty` property has an `id` property.

B.4. Conformance Class JSON

Table B.23 — Conformance Class "JSON"

Conformance Class	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/json	
Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/json

Dependency	http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/json

B.4.1. JSON Definition

Table B.24 — Abstract Test 18

Abstract Test 18	**/conf/json/definition**
Test Purpose	Verify support for JSON
Requirement	/req/json/definition
Test Method	a) A resource is requested with response media type of `application/json` b) All `200`-responses SHALL support the media type:- `application/json`

B.4.2. JSON Content

Table B.25 — Abstract Test 19

Abstract Test 19	**/conf/json/content**
Test Purpose	Verify the content of a JSON document given an input document and schema.
Requirement	/req/json/content
Test Method	a) Validate that the document is a JSON document. b) Validate the document against the schema using an JSON Schema validator.

B.5. Conformance Class GeoJSON

Table B.26 — Conformance Class "GeoJSON"

Conformance Class	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/geojson	
Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/geojson
Dependency	http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core

B.5.1. GeoJSON Definition

Table B.27 — Abstract Test 20

Abstract Test 20	/conf/geojson/definition
Test Purpose	Verify support for JSON and GeoJSON
Requirement	/req/geojson/definition
Test Method	a) A resource is requested with response media type of `application/geo+json` b) All `200`-responses SHALL support the following media types: • `application/geo+json` for resources that include feature content, and • `application/json` for all other resources.

B.5.2. GeoJSON Content

Table B.28 — Abstract Test 21

Abstract Test 21	/conf/geojson/content
Test Purpose	Verify the content of a GeoJSON document given an input document and schema.
Requirement	/req/geojson/content

Test Method	a) Validate that the document is a GeoJSON document. b) Validate the document against the schema using an JSON Schema validator. c) Validate the document against the schema using an Geo JSON Schema validator.

B.6. Conformance Class EDR GeoJSON

Table B.29 — Conformance Class "EDR GeoJSON"

Conformance Class	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/edr-geojson	
Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/edr-geojson
Dependency	http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core

B.6.1. EDR GeoJSON Definition

Table B.30 — Abstract Test 22

Abstract Test 22	**/conf/edr-geojson/definition**
Test Purpose	Verify support for the EDR GeoJSON Schema
Requirement	/req/edr-geojson/definition
Test Method	a) A resource is requested with response media type of `application/json` and adheres to the EDR Feature Collection GeoJSON Schema. b) All `200`-responses SHALL support the following media types: • `application/json` for resources

B.6.2. EDR GeoJSON Content

Table B.31 — Abstract Test 23

Abstract Test 23	**/conf/edr-geojson/content**
Test Purpose	Verify the content of an EDR GeoJSON document given an input document and schema.
Requirement	/req/edr-geojson/content
Test Method	a) Validate that the document is an EDR GeoJSON document. b) Validate the document against the schema using an JSON Schema validator. c) Validate the document against the schema using an EDR GeoJSON Schema validator.

B.7. Conformance Class CoverageJSON

Table B.32 — Conformance Class "CoverageJSON"

Conformance Class	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/covjson	
Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/covjson
Dependency	http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core

B.7.1. CoverageJSON Definition

Table B.33 — Abstract Test 24

Abstract Test 24	**/conf/covjson/definition**

Test Purpose	Verify support for CoverageJSON
Requirement	/req/covjson/definition
Test Method	a) A resource is requested with response media type of `application/prs.coverage+json` b) All `200`-responses SHALL support the following media types: • `application/prs.coverage+json` for resources

B.7.2. CoverageJSON Content

Table B.34 — Abstract Test 25

Abstract Test 25	**/conf/covjson/content**
Test Purpose	Verify the content of a CoverageJSON document given an input document and schema.
Requirement	/req/covjson/content
Test Method	a) Validate that the document is a CoverageJSON document. b) Validate the document against the schema using an JSON Schema validator. c) Validate the document against the schema using an CoverageJSON Schema validator.

B.8. Conformance Class HTML

Table B.35 — Conformance Class "HTML"

Conformance Class	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/html	
Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/html
Dependency	http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core

Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/html

B.8.1. HTML Definition

Table B.36 — Abstract Test 26

Abstract Test 26	**/conf/html/definition**
Test Purpose	Verify support for HTML
Requirement	/req/html/definition
Test Method	Verify that every `200`-response of every operation of the API where HTML was requested is of media type `text/html`

B.8.2. HTML Content

Table B.37 — Abstract Test 27

Abstract Test 27	**/conf/html/content**
Test Purpose	Verify the content of an HTML document given an input document and schema.
Requirement	/req/html/content
Test Method	a) Validate that the document is an HTML 5 document b) Manually inspect the document against the schema.

B.9. Conformance Class OpenAPI 3.0

Table B.38 — Conformance Class "OpenAPI 3.0"

Conformance Class
http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/oas30

Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/oas30
Dependency	http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/oas30

Table B.39 — Abstract Test 28

Abstract Test 28	**/conf/oas30/completeness**
Test Purpose	Verify the completeness of an OpenAPI document.
Requirement	/req/oas30/completeness
Test Method	Verify that for each operation, the OpenAPI document describes all HTTP Status Codes and Response Objects that the API uses in responses.

Table B.40 — Abstract Test 29

Abstract Test 29	**/conf/oas30/exceptions-codes**
Test Purpose	Verify that the OpenAPI document fully describes potential exception codes.
Requirement	/req/oas30/exceptions-codes
Test Method	Verify that for each operation, the OpenAPI document describes all HTTP Status Codes that may be generated.

Table B.41 — Abstract Test 30

Abstract Test 30	**/conf/oas30/oas-definition-1**
Test Purpose	Verify that JSON and HTML versions of the OpenAPI document are available.
Requirement	/req/oas30/oas-definition-1

Test Method	a) Verify that an OpenAPI definition in JSON is available using the media type `application/vnd.oai.openapi+json;version=3.0` and link relation `service-desc` b) Verify that an HTML version of the API definition is available using the media type `text/html` and link relation `service-doc`.

Table B.42 — Abstract Test 31

Abstract Test 31	**/conf/oas30/oas-definition-2**
Test Purpose	Verify that the OpenAPI document is valid JSON.
Requirement	/req/oas30/oas-definition-2
Test Method	Verify that the JSON representation conforms to the OpenAPI Specification, version 3.0.

Table B.43 — Abstract Test 32

Abstract Test 32	**/conf/oas30/oas-impl**
Test Purpose	Verify that all capabilities specified in the OpenAPI definition are implemented by the API.
Requirement	/req/oas30/oas-impl
Test Method	a) Construct a path from each URL template including all server URL options and all enumerated path parameters. b) For each path defined in the OpenAPI document, validate that the path performs in accordance with the API definition and the API-Features standard.

Table B.44 — Abstract Test 33

Abstract Test 33	**/conf/oas30/security**
Test Purpose	Verify that any authentication protocols implemented by the API are documented in the OpenAPI document.
Requirement	/req/oas30/security
Test Method	a) Identify all authentication protocols supported by the API.

	b) Validate that each authentication protocol is described inthe OpenAPI document by a Security Schema Object and its use specified by a Security Requirement Object.

B.10. Conformance Class Queries

Table B.45 — Conformance Class "Queries"

Conformance Class	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/queries	
Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/queries
Dependency	http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core
Dependency	http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/collections

B.10.1. Query Pattern Tests

B.10.1.1. Position

Table B.46 — Abstract Test 34

Abstract Test 34	**/conf/position**
Test Purpose	Validate that an error is returned by a `Position` query if no query parameters are specified.
Requirement	/req/queries/position
Test Method	a) No query parameters are specified b) Validate that a document was returned with a status code 400.

Table B.47 — Abstract Test 35

Abstract Test 35	**/conf/position**
Test Purpose	Validate that an error is returned by a `Position` query when the coords query parameter is not specified.
Requirement	/req/queries/position
Test Method	a) coords query parameter is not specified b) Validate that a document was returned with a status code 400.

Table B.48 — Abstract Test 36

Abstract Test 36	**/conf/position**
Test Purpose	Validate that an error is returned by a `Position` query when the coords query parameter does not contain a valid POINT Well Known Text value.
Requirement	/req/queries/position
Test Method	a) Check coords query parameter is a valid Well Known Text Point value b) Validate that a document was returned with a status code 400.

Table B.49 — Abstract Test 37

Abstract Test 37	**/conf/position**
Test Purpose	Validate that resources can be identified and extracted from a Collection with a `Position` query using query parameters.
Requirement	/req/queries/position
Test Method	a) Test with valid query parameters b) Validate that a document was returned with a status code 200. Repeat these tests using the following parameter tests: **Coordinates** • Parameter /req/edr/coords-definition • Response /req/edr/coords-response **VerticalLevel**

	• Parameter /req/edr/z-definition • Response /req/edr/z-response **Parameters** • Parameter /req/edr/REQ_rc-parameter-name-definition • Response /req/edr/parameter-name-response **DateTime** • Parameter /req/core/datetime-definition • Response /req/core/datetime-response Execute requests with combinations of the "coords","time","parameter-name","z","crs" and "f" query parameters and verify that only information that matches the selection criteria is returned.

Table B.50 — Abstract Test 38

Abstract Test 38	**/conf/edr/rc-coords-definition**
Test Purpose	Validate that the coords query parameters are constructed correctly.
Requirement	/req/edr/coords-definition

Test Method	Verify that the `coords` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: coords`
	`in: query`
	`required: true`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`
	
	Use a coords value in all requests:
	• A valid Well-known text representation of geometry string

Table B.51 — Abstract Test 39

Abstract Test 39	**/conf/edr/rc-coords-response**
Test Purpose	Validate that the `coords` query parameters are processed corrrectly.
Requirement	/req/edr/coords-response

Test Method	a) Verify that only resources that have a spatial geometry that intersects the coordinates are returned as part of the result set.
Test Method	a) Verify `coords` values are valid for the specified coordinate reference system b) Verify that the coordinate reference system of the geometries are valid for the parameter defined by `crs`. If the `crs` parameter is not defined the geometries must be valid for WGS 84 longitude/latitude (`http://www.opengis.net/def/crs/OGC/1.3/CRS84` or `http://www.opengis.net/def/crs/OGC/0/CRS84h`).

Table B.52 — Abstract Test 40

Abstract Test 40	**/conf/edr/rc-z-definition**
Test Purpose	Validate that the vertical level query parameters are constructed correctly.
Requirement	/req/edr/rc-z-definition

Test Method	Verify that the `z` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: z`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.53 — Abstract Test 41

Abstract Test 41	**/conf/edr/rc-z-response**
Test Purpose	Validate that the vertical level query parameters are processed correctly.
Requirement	/req/edr/z-response
Test Method	a) Verify that only resources that have a vertical geometry that intersects the vertical information in the `z` parameter were included in the result set b) Validate that the vertical level parameter complies with the syntax described in /req/edr/z-response.

Table B.54 — Abstract Test 42

Abstract Test 42	/conf/core/datetime-definition
Test Purpose	Validate that the datetime query parameters are constructed correctly.
Requirement	/req/core/datetime-definition

Test Method	Verify that the `datetime` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: datetime`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.55 — Abstract Test 43

Abstract Test 43	/conf/core/datetime-response
Test Purpose	Validate that the `datetime` query parameters are processed correctly.
Requirement	/req/core/datetime-response
Test Method	a) Verify that only resources that have a temporal geometry that intersects the temporal information in the `datetime` parameter were included in the result set. b) Verify that all resources in the collection that are not associated with a temporal geometry are included in the result set. c) Validate that the `datetime` parameter complies with the syntax described in /req/core/datetime-response.

Table B.56 — Abstract Test 44

Abstract Test 44	/conf/collections/REQ_rc-parameter-name-definition
Test Purpose	Validate that the parameter-name query parameters are constructed correctly.
Requirement	/req/edr/REQ_rc-parameter-name-definition

Test Method	Verify that the `parameter-name` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: parameter-name`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.57 — Abstract Test 45

Abstract Test 45	**/conf/edr/rc-parameter-name-response**
Test Purpose	Validate that the parameter-name query parameters are processed correctly.
Requirement	/req/edr/parameter-name-response
Test Method	a) Verify that only resources for the requested parameters were included in the result set b) Validate that the parameter-name parameter complies with the syntax described in /req/edr/parameter-name-response.

Table B.58 — Abstract Test 46

Abstract Test 46	**/conf/edr/REQ_rc-crs-definition**
Test Purpose	Validate that the crs query parameters are constructed correctly.
Requirement	/req/edr/REQ_rc-crs-definition

Test Method	Verify that the `crs` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: crs`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.59 — Abstract Test 47

Abstract Test 47	**/conf/edr/REQ_rc-crs-response**
Test Purpose	Validate that the crs query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-crs-response
Test Method	a) Verify that the geometry of the resources returned are valid for the requested coordinate reference system
Test Method	a) Verify that all crs values defined in the collections metadata are supported by the collection
Test Method	a) Verify that all crs values not defined in the collections metadata will generate a HTTP `400` error b) Validate that the crs parameter complies with the syntax described in /req/edr/REQ_rc-crs-response.

Table B.60 — Abstract Test 48

Abstract Test 48	**/conf/edr/rc-f-definition**
Test Purpose	Validate that the `f` query parameter is constructed correctly.
Requirement	/req/edr/rc-f-definition

Test Method	Verify that the `f` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: f`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.61 — Abstract Test 49

Abstract Test 49	**/conf/collections/rc-f-response**
Test Purpose	Validate that the `f` query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-f-response
Test Method	a) Verify that the response is returned in the requested data format

Test Method	a) Verify that all output format values defined in the collections metadata are supported by the collection b) Validate that the `f` parameter complies with the syntax described in /req/edr/REQ_rc-f-response.

B.10.1.2. Area

Table B.62 — Abstract Test 50

Abstract Test 50	**/conf/area**
Test Purpose	Validate that an error is returned by a `Area` query if no query parameters are specified.
Requirement	/req/edr/rc-area
Test Method	a) No query parameters are specified b) Validate that a document was returned with a status code 400.

Table B.63 — Abstract Test 51

Abstract Test 51	**/conf/area**
Test Purpose	Validate that an error is returned by a `Area` query when the coords query parameter is not specified.
Requirement	/req/edr/rc-area
Test Method	a) coords query parameter is not specified b) Validate that a document was returned with a status code 400.

Table B.64 — Abstract Test 52

Abstract Test 52	**/conf/area**
Test Purpose	Validate that an error is returned by a `Area` query when the coords query parameter does not contain a valid POLYGON Well Known Text value.
Requirement	/req/edr/rc-area

Test Method	a) Check coords query parameter is a valid Well Known Text Polygon value b) Validate that a document was returned with a status code 400.

Table B.65 — Abstract Test 53

Abstract Test 53	**/conf/area**
Test Purpose	Validate that resources can be identified and extracted from a Collection with an `Area` query using query parameters.
Requirement	/req/edr/rc-area
Test Method	a) Test with valid query parameters b) Validate that a document was returned with a status code 200. Repeat these tests using the following parameter tests: **Coordinates** • Parameter /req/edr/coords-definition • Response /req/edr/coords-response **VerticalLevel** • Parameter /req/edr/z-definition • Response /req/edr/z-response **Parameters** • Parameter /req/edr/REQ_rc-parameter-name-definition • Response /req/edr/parameter-name-response **DateTime** • Parameter /req/core/datetime-definition • Response /req/core/datetime-response Execute requests with combinations of the “coords”,”time”,”parameter-name”,”z”,”crs” and “f” query parameters and verify that only information that matches the selection criteria is returned.

Table B.66 — Abstract Test 54

Abstract Test 54	**/conf/edr/rc-coords-definition**

Test Purpose	Validate that the coords query parameters are constructed correctly.
Requirement	/req/edr/coords-definition

Test Method	Verify that the `coords` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: coords`
	`in: query`
	`required: true`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`
	Use a coords value in all requests:
	• A valid Well-known text representation of geometry string

Table B.67 — Abstract Test 55

Abstract Test 55	**/conf/edr/rc-coords-response**
Test Purpose	Validate that the `coords` query parameters are processed corrrectly.
Requirement	/req/edr/coords-response
Test Method	a) Verify that only resources that have a spatial geometry that intersects the coordinates are returned as part of the result set.
Test Method	a) Verify `coords` values are valid for the specified coordinate reference system b) Verify that the coordinate reference system of the geometries are valid for the parameter defined by `crs`. If the `crs` parameter is not defined the geometries must be valid for WGS 84 longitude/latitude (`http://www.opengis.net/def/crs/OGC/1.3/CRS84` or `http://www.opengis.net/def/crs/OGC/0/CRS84h`).

Table B.68 — Abstract Test 56

Abstract Test 56	**/conf/edr/rc-z-definition**
Test Purpose	Validate that the vertical level query parameters are constructed correctly.
Requirement	/req/edr/rc-z-definition

Test Method	Verify that the `z` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: z`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.69 — Abstract Test 57

Abstract Test 57	**/conf/edr/rc-z-response**
Test Purpose	Validate that the vertical level query parameters are processed correctly.
Requirement	/req/edr/z-response
Test Method	a) Verify that only resources that have a vertical geometry that intersects the vertical information in the `z` parameter were included in the result set b) Validate that the vertical level parameter complies with the syntax described in /req/edr/z-response.

Table B.70 — Abstract Test 58

Abstract Test 58	**/conf/core/datetime-definition**
Test Purpose	Validate that the datetime query parameters are constructed correctly.
Requirement	/req/core/datetime-definition

Test Method	Verify that the `datetime` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: datetime`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.71 — Abstract Test 59

Abstract Test 59	**/conf/core/datetime-response**
Test Purpose	Validate that the `datetime` query parameters are processed correctly.
Requirement	/req/core/datetime-response
Test Method	a) Verify that only resources that have a temporal geometry that intersects the temporal information in the `datetime` parameter were included in the result set. b) Verify that all resources in the collection that are not associated with a temporal geometry are included in the result set. c) Validate that the `datetime` parameter complies with the syntax described in /req/core/datetime-response.

Table B.72 — Abstract Test 60

Abstract Test 60	**/conf/collections/REQ_rc-parameter-name-definition**
Test Purpose	Validate that the parameter-name query parameters are constructed correctly.
Requirement	/req/edr/REQ_rc-parameter-name-definition

Test Method	Verify that the `parameter-name` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: parameter-name`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.73 — Abstract Test 61

Abstract Test 61	**/conf/edr/rc-parameter-name-response**
Test Purpose	Validate that the parameter-name query parameters are processed correctly.
Requirement	/req/edr/parameter-name-response
Test Method	a) Verify that only resources for the requested parameters were included in the result set

	b) Validate that the parameter-name parameter complies with the syntax described in /req/edr/parameter-name-response.

Table B.74 — Abstract Test 62

Abstract Test 62	**/conf/edr/REQ_rc-crs-definition**
Test Purpose	Validate that the crs query parameters are constructed correctly.
Requirement	/req/edr/REQ_rc-crs-definition

Test Method	Verify that the `crs` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: crs`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.75 — Abstract Test 63

Abstract Test 63	**/conf/edr/REQ_rc-crs-response**
Test Purpose	Validate that the crs query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-crs-response
Test Method	a) Verify that the geometry of the resources returned are valid for the requested coordinate reference system
Test Method	a) Verify that all crs values defined in the collections metadata are supported by the collection
Test Method	a) Verify that all crs values not defined in the collections metadata will generate a HTTP `400` error b) Validate that the crs parameter complies with the syntax described in /req/edr/REQ_rc-crs-response.

Table B.76 — Abstract Test 64

Abstract Test 64	**/conf/edr/rc-f-definition**
Test Purpose	Validate that the `f` query parameter is constructed correctly.

Requirement	/req/edr/rc-f-definition

Test Method	Verify that the `f` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: f`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.77 — Abstract Test 65

Abstract Test 65	**/conf/collections/rc-f-response**
Test Purpose	Validate that the `f` query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-f-response
Test Method	a) Verify that the response is returned in the requested data format
Test Method	a) Verify that all output format values defined in the collections metadata are supported by the collection b) Validate that the `f` parameter complies with the syntax described in /req/edr/REQ_rc-f-response.

B.10.1.3. Cube

Table B.78 — Abstract Test 66

Abstract Test 66	**/conf/cube**
Test Purpose	Validate that an error is returned by a `Cube` query if no query parameters are specified.
Requirement	/req/edr/rc-cube
Test Method	a) No query parameters are specified b) Validate that a document was returned with a status code 400.

Table B.79 — Abstract Test 67

Abstract Test 67	**/conf/cube**
Test Purpose	Validate that an error is returned by a `Cube` query when the `bbox` query parameter is not specified.
Requirement	/req/edr/rc-cube
Test Method	a) bbox query parameter is not specified b) Validate that a document was returned with a status code 400.

Table B.80 — Abstract Test 68

Abstract Test 68	**/conf/cube**
Test Purpose	Validate that an error is returned by a `Cube` query when the `bbox` query parameter does not contain a valid bbox value.
Requirement	/req/edr/rc-cube
Test Method	a) Check bbox query parameter is valid b) Validate that a document was returned with a status code 400.

Table B.81 — Abstract Test 69

Abstract Test 69	**/conf/cube**
Test Purpose	Validate that resources can be identified and extracted from a Collection with a `Cube` query using query parameters.
Requirement	/req/edr/rc-cube
Test Method	a) Test with valid query parameters b) Validate that a document was returned with a status code 200. Repeat these tests using the following parameter tests: **bbox** • Parameter /req/core/rc-bbox-definition • Response /req/core/rc-bbox-response **VerticalLevel** • Parameter /req/edr/z-definition • Response /req/edr/cube-z-response

	Parameters • Parameter /req/edr/REQ_rc-parameter-name-definition • Response /req/edr/parameter-name-response **DateTime** • Parameter /req/core/datetime-definition • Response /req/core/datetime-response Execute requests with combinations of the “bbox”,”time”,”parameter-name”,”z”,”crs” and “f” query parameters and verify that only information that matches the selection criteria is returned.

Table B.82 — Abstract Test 70

Abstract Test 70	**/conf/edr/rc-coords-definition**
Test Purpose	Validate that the coords query parameters are constructed correctly.
Requirement	/req/edr/coords-definition

Test Method	Verify that the `coords` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: coords`
	`in: query`
	`required: true`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`
	
	Use a coords value in all requests:
	• A valid Well-known text representation of geometry string

Table B.83 — Abstract Test 71

Abstract Test 71	**/conf/edr/rc-coords-response**
Test Purpose	Validate that the `coords` query parameters are processed corrrectly.
Requirement	/req/edr/coords-response
Test Method	a) Verify that only resources that have a spatial geometry that intersects the coordinates are returned as part of the result set.

Test Method	a) Verify `coords` values are valid for the specified coordinate reference system b) Verify that the coordinate reference system of the geometries are valid for the parameter defined by `crs`. If the `crs` parameter is not defined the geometries must be valid for WGS 84 longitude/latitude (`http://www.opengis.net/def/crs/OGC/1.3/CRS84` or `http://www.opengis.net/def/crs/OGC/0/CRS84h`).

Table B.84 — Abstract Test 72

Abstract Test 72	**/conf/edr/rc-z-definition**
Test Purpose	Validate that the vertical level query parameters are constructed correctly.
Requirement	/req/edr/rc-z-definition

Test Method	Verify that the `z` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: z`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.85 — Abstract Test 73

Abstract Test 73	**/conf/edr/rc-cube-z-response**
Test Purpose	Validate that the vertical level query parameters are processed correctly.
Requirement	/req/edr/cube-z-response
Test Method	a) Verify that only resources that have a vertical geometry that intersects the vertical information in the `z` parameter were included in the result set b) Validate that the vertical level parameter complies with the syntax described in /req/edr/cube-z-response.

Table B.86 — Abstract Test 74

Abstract Test 74	**/conf/core/datetime-definition**

Test Purpose	Validate that the datetime query parameters are constructed correctly.
Requirement	/req/core/datetime-definition

Test Method	Verify that the `datetime` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: datetime`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.87 — Abstract Test 75

Abstract Test 75	**/conf/core/datetime-response**
Test Purpose	Validate that the `datetime` query parameters are processed correctly.
Requirement	/req/core/datetime-response
Test Method	a) Verify that only resources that have a temporal geometry that intersects the temporal information in the `datetime` parameter were included in the result set. b) Verify that all resources in the collection that are not associated with a temporal geometry are included in the result set. c) Validate that the `datetime` parameter complies with the syntax described in /req/core/datetime-response.

Table B.88 — Abstract Test 76

Abstract Test 76	**/conf/collections/REQ_rc-parameter-name-definition**
Test Purpose	Validate that the parameter-name query parameters are constructed correctly.
Requirement	/req/edr/REQ_rc-parameter-name-definition

Test Method	Verify that the `parameter-name` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: parameter-name`
	`in: query`
	`required: false`

```
schema:
  type: string
style: form
explode: false
```

Table B.89 — Abstract Test 77

Abstract Test 77	**/conf/edr/rc-parameter-name-response**
Test Purpose	Validate that the parameter-name query parameters are processed correctly.
Requirement	/req/edr/parameter-name-response
Test Method	a) Verify that only resources for the requested parameters were included in the result set b) Validate that the parameter-name parameter complies with the syntax described in /req/edr/parameter-name-response.

Table B.90 — Abstract Test 78

Abstract Test 78	**/conf/edr/REQ_rc-crs-definition**
Test Purpose	Validate that the crs query parameters are constructed correctly.
Requirement	/req/edr/REQ_rc-crs-definition

Test Method	Verify that the `crs` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: crs`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.91 — Abstract Test 79

Abstract Test 79	**/conf/edr/REQ_rc-crs-response**
Test Purpose	Validate that the crs query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-crs-response

Test Method	a) Verify that the geometry of the resources returned are valid for the requested coordinate reference system
Test Method	a) Verify that all crs values defined in the collections metadata are supported by the collection
Test Method	a) Verify that all crs values not defined in the collections metadata will generate a HTTP `400` error b) Validate that the crs parameter complies with the syntax described in /req/edr/REQ_rc-crs-response.

Table B.92 — Abstract Test 80

Abstract Test 80	**/conf/edr/rc-f-definition**
Test Purpose	Validate that the `f` query parameter is constructed correctly.
Requirement	/req/edr/rc-f-definition

Test Method	Verify that the `f` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: f`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.93 — Abstract Test 81

Abstract Test 81	**/conf/collections/rc-f-response**
Test Purpose	Validate that the `f` query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-f-response
Test Method	a) Verify that the response is returned in the requested data format
Test Method	a) Verify that all output format values defined in the collections metadata are supported by the collection b) Validate that the `f` parameter complies with the syntax described in /req/edr/REQ_rc-f-response.

B.10.1.4. Trajectory

Table B.94 — Abstract Test 82

Abstract Test 82	**/conf/trajectory**
Test Purpose	Validate that an error is returned by a `Trajectory` query if no query parameters are specified.
Requirement	/req/edr/rc-trajectory
Test Method	a) No query parameters are specified b) Validate that a document was returned with a status code 400.

Table B.95 — Abstract Test 83

Abstract Test 83	**/conf/trajectory**
Test Purpose	Validate that an error is returned by a `Trajectory` query when the coords query parameter is not specified.
Requirement	/req/edr/rc-trajectory
Test Method	a) coords query parameter is not specified b) Validate that a document was returned with a status code 400.

Table B.96 — Abstract Test 84

Abstract Test 84	**/conf/trajectory**
Test Purpose	Validate that an error is returned by a `Trajectory` query when the coords query parameter does not contain a valid LINESTRING Well Known Text value.
Requirement	/req/edr/rc-trajectory
Test Method	a) Check coords query parameter is a valid Well Known Text Linestring value b) Validate that a document was returned with a status code 400.

Table B.97 — Abstract Test 85

Abstract Test 85	**/conf/trajectory**

Test Purpose	Validate that an error is returned by a `Trajectory` query when the coords query parameter does not contain a valid LINESTRINGM Well Known Text value.
Requirement	/req/edr/rc-trajectory
Test Method	a) Check coords query parameter with time parameter is a valid Well Known Text LINESTRINGM value, the `M` coordinate must be a valid Epoch value (as known as UNIX time) b) Validate that a document was returned with a status code 400.

Table B.98 — Abstract Test 86

Abstract Test 86	**/conf/trajectory**
Test Purpose	Validate that an error is returned by a `Trajectory` query when the `coords` query parameter is a LINESTRINGZ coordinate and the `z`query parameter is specified
Requirement	/req/edr/rc-trajectory
Test Method	a) Check coords query parameter that the system throws an error when a vertical level is specified in both the `coords` and `z` parameters b) Validate that a document was returned with a status code 400.

Table B.99 — Abstract Test 87

Abstract Test 87	**/conf/trajectory**
Test Purpose	Validate that an error is returned by a `Trajectory` query when the `coords` query parameter is a LINESTRINGZM coordinate and the `z`query parameter is specified
Requirement	/req/edr/rc-trajectory
Test Method	a) Check coords query parameter that the system throws an error when a vertical level is specified in both the `coords` and `z` parameters b) Validate that a document was returned with a status code 400.

Table B.100 — Abstract Test 88

Abstract Test 88	**/conf/trajectory**
Test Purpose	Validate that an error is returned by a `Trajectory` query when the coords query parameter does not contain a valid LINESTRINGZM Well Known Text value.
Requirement	/req/edr/rc-trajectory
Test Method	a) Check coords query parameter with time parameter is a valid Well Known Text LINESTRINGZM value, the `Z` coordinate must be a within the range of vertical levels advertised in the Collection metadata b) Validate that a document was returned with a status code 400.

Table B.101 — Abstract Test 89

Abstract Test 89	**/conf/trajectory**
Test Purpose	Validate that an error is returned by a `Trajectory` query when the coords query parameter does not contain a valid LINESTRINGZ Well Known Text value.
Requirement	/req/edr/rc-trajectory
Test Method	a) Check coords query parameter with time parameter is a valid Well Known Text LINESTRINGZ value, the `Z` coordinate must be a within the range of vertical levels advertised in the Collection metadata b) Validate that a document was returned with a status code 400.

Table B.102 — Abstract Test 90

Abstract Test 90	**/conf/trajectory**
Test Purpose	Validate that an error is returned by a `Trajectory` query when the `coords` query parameter contains invalid time coordinates
Requirement	/req/edr/rc-trajectory
Test Method	a) If a time values are specified in the `coords` query parameter check that they are within the range of time values defined in the Collection metadata b) Validate that a document was returned with a status code 400.

Table B.103 — Abstract Test 91

Abstract Test 91	**/conf/trajectory**
Test Purpose	Validate that resources can be identified and extracted from a Collection with a `Trajectory` query using query parameters.
Requirement	/req/edr/rc-trajectory
Test Method	a) Test with valid query parameters b) Validate that a document was returned with a status code 200. Repeat these tests using the following parameter tests: **Coordinates** • Parameter /req/edr/coords-definition • Response /req/edr/coords-response **VerticalLevel** • Parameter /req/edr/z-definition • Response /req/edr/z-response **Parameters** • Parameter /req/edr/REQ_rc-parameter-name-definition • Response /req/edr/parameter-name-response **DateTime** • Parameter /req/core/datetime-definition • Response /req/core/datetime-response Execute requests with combinations of the “coords”, “parameter-name”,”z”,”crs” and “f” query parameters and verify that only information that matches the selection criteria is returned.

Table B.104 — Abstract Test 92

Abstract Test 92	**/conf/edr/rc-coords-definition**
Test Purpose	Validate that the coords query parameters are constructed correctly.
Requirement	/req/edr/coords-definition
Test Method	Verify that the `coords` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):

```
name: coords
in: query
required: true
schema:
  type: string
style: form
explode: false
```

Use a coords value in all requests:

- A valid Well-known text representation of geometry string

Table B.105 — Abstract Test 93

Abstract Test 93	**/conf/edr/rc-coords-response**
Test Purpose	Validate that the `coords` query parameters are processed corrrectly.
Requirement	/req/edr/coords-response
Test Method	a) Verify that only resources that have a spatial geometry that intersects the coordinates are returned as part of the result set.
Test Method	a) Verify `coords` values are valid for the specified coordinate reference system b) Verify that the coordinate reference system of the geometries are valid for the parameter defined by `crs`. If the `crs` parameter is not defined the geometries must be valid for WGS 84 longitude/latitude (`http://www.opengis.net/def/crs/OGC/1.3/CRS84` or `http://www.opengis.net/def/crs/OGC/0/CRS84h`).

Table B.106 — Abstract Test 94

Abstract Test 94	**/conf/collections/REQ_rc-parameter-name-definition**
Test Purpose	Validate that the parameter-name query parameters are constructed correctly.
Requirement	/req/edr/REQ_rc-parameter-name-definition

Test Method	Verify that the `parameter-name` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: parameter-name`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`

```
explode: false
```

Table B.107 — Abstract Test 95

Abstract Test 95	**/conf/edr/rc-parameter-name-response**
Test Purpose	Validate that the parameter-name query parameters are processed correctly.
Requirement	/req/edr/parameter-name-response
Test Method	a) Verify that only resources for the requested parameters were included in the result set b) Validate that the parameter-name parameter complies with the syntax described in /req/edr/parameter-name-response.

Table B.108 — Abstract Test 96

Abstract Test 96	**/conf/edr/REQ_rc-crs-definition**
Test Purpose	Validate that the crs query parameters are constructed correctly.
Requirement	/req/edr/REQ_rc-crs-definition

Test Method	Verify that the `crs` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: crs`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.109 — Abstract Test 97

Abstract Test 97	**/conf/edr/REQ_rc-crs-response**
Test Purpose	Validate that the crs query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-crs-response
Test Method	a) Verify that the geometry of the resources returned are valid for the requested coordinate reference system

Test Method	a) Verify that all crs values defined in the collections metadata are supported by the collection
Test Method	a) Verify that all crs values not defined in the collections metadata will generate a HTTP `400` error b) Validate that the crs parameter complies with the syntax described in /req/edr/REQ_rc-crs-response.

Table B.110 — Abstract Test 98

Abstract Test 98	**/conf/edr/rc-f-definition**
Test Purpose	Validate that the `f` query parameter is constructed correctly.
Requirement	/req/edr/rc-f-definition

Test Method	Verify that the `f` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: f`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.111 — Abstract Test 99

Abstract Test 99	**/conf/collections/rc-f-response**
Test Purpose	Validate that the `f` query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-f-response
Test Method	a) Verify that the response is returned in the requested data format
Test Method	a) Verify that all output format values defined in the collections metadata are supported by the collection b) Validate that the `f` parameter complies with the syntax described in /req/edr/REQ_rc-f-response.

B.10.1.5. Corridor

Table B.112 — Abstract Test 100

Abstract Test 100	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query if no query parameters are specified.
Requirement	/req/edr/rc-corridor
Test Method	a) No query parameters are specified b) Validate that a document was returned with a status code 400.

Table B.113 — Abstract Test 101

Abstract Test 101	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query when the coords query parameter is not specified.
Requirement	/req/edr/rc-corridor
Test Method	a) coords query parameter is not specified b) Validate that a document was returned with a status code 400.

Table B.114 — Abstract Test 102

Abstract Test 102	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query when the `corridor-width` query parameter is not specified.
Requirement	/req/edr/rc-corridor
Test Method	a) corridor-width query parameter is not specified b) Validate that a document was returned with a status code 400.

Table B.115 — Abstract Test 103

Abstract Test 103	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query when the `corridor-height` query parameter is not specified.

Requirement	/req/edr/rc-corridor
Test Method	a) corridor-height query parameter is not specified b) Validate that a document was returned with a status code 400.

Table B.116 — Abstract Test 104

Abstract Test 104	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query when the `width-units` query parameter is not specified.
Requirement	/req/edr/rc-corridor
Test Method	a) width-units query parameter is not specified b) Validate that a document was returned with a status code 400.

Table B.117 — Abstract Test 105

Abstract Test 105	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query when the `height-units` query parameter is not specified.
Requirement	/req/edr/rc-corridor
Test Method	a) height-units query parameter is not specified b) Validate that a document was returned with a status code 400.

Table B.118 — Abstract Test 106

Abstract Test 106	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query when the coords query parameter does not contain a valid LINESTRING Well Known Text value.
Requirement	/req/edr/rc-corridor
Test Method	a) Check coords query parameter is a valid Well Known Text Linestring value

	b) Validate that a document was returned with a status code 400.

Table B.119 — Abstract Test 107

Abstract Test 107	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query when the coords query parameter does not contain a valid LINESTRINGM Well Known Text value.
Requirement	/req/edr/rc-corridor
Test Method	a) Check coords query parameter with time parameter is a valid Well Known Text LINESTRINGM value, the `M` coordinate must be a valid Epoch value (as known as UNIX time) b) Validate that a document was returned with a status code 400.

Table B.120 — Abstract Test 108

Abstract Test 108	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query when the `coords` query parameter is a LINESTRINGZ coordinate and the `z`query parameter is specified
Requirement	/req/edr/rc-corridor
Test Method	a) Check coords query parameter that the system throws an error when a vertical level is specified in both the `coords` and `z` parameters b) Validate that a document was returned with a status code 400.

Table B.121 — Abstract Test 109

Abstract Test 109	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query when the `coords` query parameter is a LINESTRINGZM coordinate and the `z`query parameter is specified
Requirement	/req/edr/rc-corridor

Test Method	a) Check coords query parameter that the system throws an error when a vertical level is specified in both the `coords` and `z` parameters b) Validate that a document was returned with a status code 400.

Table B.122 — Abstract Test 110

Abstract Test 110	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query when the coords query parameter does not contain a valid LINESTRINGMZ Well Known Text value.
Requirement	/req/edr/rc-corridor
Test Method	a) Check coords query parameter with time parameter is a valid Well Known Text LINESTRINGMZ value, the `Z` coordinate must be a within the range of vertical levels advertised in the Collection metadata b) Validate that a document was returned with a status code 400.

Table B.123 — Abstract Test 111

Abstract Test 111	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query when the coords query parameter does not contain a valid LINESTRINGZ Well Known Text value.
Requirement	/req/edr/rc-corridor
Test Method	a) Check coords query parameter with time parameter is a valid Well Known Text LINESTRINGZ value, the `Z` coordinate must be a within the range of vertical levels advertised in the Collection metadata b) Validate that a document was returned with a status code 400.

Table B.124 — Abstract Test 112

Abstract Test 112	**/conf/corridor**

Test Purpose	Validate that an error is returned by a `corridor` query when the `coords` query parameter contains invalid time coordinates
Requirement	/req/edr/rc-corridor
Test Method	a) If a time values are specified in the `coords` query parameter check that they are within the range of time values defined in the Collection metadata b) Validate that a document was returned with a status code 400.

Table B.125 — Abstract Test 113

Abstract Test 113	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query when the `width-units` query parameter contains invalid units
Requirement	/req/edr/rc-corridor
Test Method	a) Specify a width-units value that is not listed in the collection response b) Validate that a document was returned with a status code 400.

Table B.126 — Abstract Test 114

Abstract Test 114	**/conf/corridor**
Test Purpose	Validate that an error is returned by a `corridor` query when the `height-units` query parameter contains invalid units
Requirement	/req/edr/rc-corridor
Test Method	a) Specify a height-units value that is not listed in the collection response b) Validate that a document was returned with a status code 400.

Table B.127 — Abstract Test 115

Abstract Test 115	**/conf/corridor**
Test Purpose	Validate that resources can be identified and extracted from a Collection with a `corridor` query using query parameters.

Requirement	/req/edr/rc-corridor
Test Method	a) Test with valid query parameters b) Validate that a document was returned with a status code 200. Repeat these tests using the following parameter tests: **Coordinates** • Parameter /req/edr/coords-definition • Response /req/edr/coords-response **VerticalLevel** • Parameter /req/edr/z-definition • Response /req/edr/z-response **Parameters** • Parameter /req/edr/REQ_rc-parameter-name-definition • Response /req/edr/parameter-name-response **DateTime** • Parameter /req/core/datetime-definition • Response /req/core/datetime-response Execute requests with combinations of the “coords”, “parameter-name”,”z”,”crs” and “f” query parameters and verify that only information that matches the selection criteria is returned.

Table B.128 — Abstract Test 116

Abstract Test 116	**/conf/edr/rc-coords-definition**
Test Purpose	Validate that the coords query parameters are constructed correctly.
Requirement	/req/edr/coords-definition

Test Method	Verify that the `coords` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: coords`
	`in: query`
	`required: true`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`
	
	Use a coords value in all requests:
	• A valid Well-known text representation of geometry string

Table B.129 — Abstract Test 117

Abstract Test 117	**/conf/edr/rc-coords-response**
Test Purpose	Validate that the `coords` query parameters are processed corrrectly.
Requirement	/req/edr/coords-response
Test Method	a) Verify that only resources that have a spatial geometry that intersects the coordinates are returned as part of the result set.
Test Method	a) Verify `coords` values are valid for the specified coordinate reference system b) Verify that the coordinate reference system of the geometries are valid for the parameter defined by `crs`. If the `crs` parameter is not defined the geometries must be valid for WGS 84 longitude/latitude (`http://www.opengis.net/def/crs/OGC/1.3/CRS84` or `http://www.opengis.net/def/crs/OGC/0/CRS84h`).

Table B.130 — Abstract Test 118

Abstract Test 118	**/conf/edr/REQ_rc-corridor-width-definition**
Test Purpose	Validate that the `corridor-width` query parameter is constructed correctly.
Requirement	/req/edr/corridor-width-definition

Test Method	Verify that the `corridor-width` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: corridor-width`
	`in: query`
	`required: true`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.131 — Abstract Test 119

Abstract Test 119	**/conf/collections/REQ_rc-corridor-width-response**

Test Purpose	Validate that the `corridor-width` query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-corridor-width-response
Test Method	a) Verify that an `400` error will be generated if corridor-width is not specified b) Validate that the `corridor-width` parameter complies with the syntax described in /req/edr/REQ_rc-corridor-width-response.

Table B.132 — Abstract Test 120

Abstract Test 120	**/conf/edr/REQ_rc-corridor-height-definition**
Test Purpose	Validate that the `corridor-height` query parameter is constructed correctly.
Requirement	/req/edr/REQ_rc-corridor-height-definition

Test Method	Verify that the `corridor-height` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: corridor-height`
	`in: query`
	`required: true`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.133 — Abstract Test 121

Abstract Test 121	**/conf/collections/REQ_rc-corridor-height-response**
Test Purpose	Validate that the `corridor-height` query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-corridor-height-response
Test Method	a) Verify that an `400` error will be generated if corridor-height is not specified b) Validate that the `corridor-height` parameter complies with the syntax described in /req/edr/REQ_rc-corridor-height-response.

Table B.134 — Abstract Test 122

Abstract Test 122	**/conf/edr/REQ_rc-width-units-definition**
Test Purpose	Validate that the `width-units` query parameter is constructed correctly.
Requirement	/req/edr/REQ_rc-width-units-definition

Test Method	Verify that the `width-units` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: width-units`
	`in: query`
	`required: true`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.135 — Abstract Test 123

Abstract Test 123	**/conf/collections/REQ_rc-width-units-response**
Test Purpose	Validate that the `width-units` query parameters are processed correctly.
Requirement	/req/edr/width-units-response
Test Method	a) Verify that units not listed in the metadata will generate an error message b) Validate that the `width-units` parameter complies with the syntax described in /req/edr/width-units-response.

Table B.136 — Abstract Test 124

Abstract Test 124	**/conf/edr/REQ_rc-height-units-definition**
Test Purpose	Validate that the `height-units` query parameter is constructed correctly.
Requirement	/req/edr/REQ_rc-height-units-definition

Test Method	Verify that the `within-units` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: height-units`
	`in: query`
	`required: true`

```
schema:
  type: string
style: form
explode: false
```

Table B.137 — Abstract Test 125

Abstract Test 125	**/conf/collections/rc-height-units-response**
Test Purpose	Validate that the `height-units` query parameters are processed correctly.
Requirement	/req/edr/height-units-response
Test Method	a) Verify that height units not listed in the metadata will generate an error message b) Validate that the `height-units` parameter complies with the syntax described in /req/edr/height-units-response.

Table B.138 — Abstract Test 126

Abstract Test 126	**/conf/collections/REQ_rc-parameter-name-definition**
Test Purpose	Validate that the parameter-name query parameters are constructed correctly.
Requirement	/req/edr/REQ_rc-parameter-name-definition

Test Method	Verify that the `parameter-name` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: parameter-name`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.139 — Abstract Test 127

Abstract Test 127	**/conf/edr/rc-parameter-name-response**
Test Purpose	Validate that the parameter-name query parameters are processed correctly.

Requirement	/req/edr/parameter-name-response
Test Method	a) Verify that only resources for the requested parameters were included in the result set b) Validate that the parameter-name parameter complies with the syntax described in /req/edr/parameter-name-response.

Table B.140 — Abstract Test 128

Abstract Test 128	**/conf/edr/REQ_rc-crs-definition**
Test Purpose	Validate that the crs query parameters are constructed correctly.
Requirement	/req/edr/REQ_rc-crs-definition

Test Method	Verify that the `crs` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: crs`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.141 — Abstract Test 129

Abstract Test 129	**/conf/edr/REQ_rc-crs-response**
Test Purpose	Validate that the crs query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-crs-response
Test Method	a) Verify that the geometry of the resources returned are valid for the requested coordinate reference system
Test Method	a) Verify that all crs values defined in the collections metadata are supported by the collection
Test Method	a) Verify that all crs values not defined in the collections metadata will generate a HTTP `400` error b) Validate that the crs parameter complies with the syntax described in /req/edr/REQ_rc-crs-response.

Table B.142 — Abstract Test 130

Abstract Test 130	**/conf/edr/rc-f-definition**
Test Purpose	Validate that the `f` query parameter is constructed correctly.
Requirement	/req/edr/rc-f-definition

Test Method	Verify that the `f` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: f`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.143 — Abstract Test 131

Abstract Test 131	**/conf/collections/rc-f-response**
Test Purpose	Validate that the `f` query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-f-response
Test Method	a) Verify that the response is returned in the requested data format
Test Method	a) Verify that all output format values defined in the collections metadata are supported by the collection b) Validate that the `f` parameter complies with the syntax described in /req/edr/REQ_rc-f-response.

B.10.1.6. Instances {root}/collections/{collectionId}/instances

Table B.144 — Abstract Test 132

Abstract Test 132	**/conf/instances/rc-md-op**
Test Purpose	Validate that information about the instances of a Collection can be retrieved from the expected location.
Requirement	/req/instances/rc-md-op
Test Method	a) Issue an HTTP GET request to the URL {root}/collections/{collectionId}/instances

	b) Validate that a document was returned with a status code 200 c) Validate the contents of the returned document using test /conf/instances_rc-md-success.

Table B.145 — Abstract Test 133

Abstract Test 133	**/conf/instances_rc-md-success**
Test Purpose	Validate that the instances of the Collection content complies with the required structure and contents.
Requirement	/req/instances/rc-md-success, /req/core/crs84
Test Method	a) Validate that all response documents comply with /req/core/rc-collection-info-links b) In case the response includes a "crs" property, validate that the first value is either "http://www.opengis.net/def/crs/OGC/1.3/CRS84" or "http://www.opengis.net/def/crs/OGC/0/CRS84h" c) Validate the collections content for all supported media types using the resources and tests identified in Table B.14

The Instances content, unlike the Collections content, may only be retrieved in the same formats as specified for the single parent collection.

Table B.146 — Schema and Tests for Collections content

FORMA	SCHEMA DOCUMENT	TEST ID
HTML	collections.yaml	/conf/html/content
JSON	collections.yaml	/conf/geojson/content

B.10.1.7. Instance {root}/collections/{collectionId}/instances/instanceId

Table B.147 — Abstract Test 134

Abstract Test 134	**/conf/instances/src-md-op**
Test Purpose	Validate that the Instances of the Collection content can be retrieved from the expected location.
Requirement	/req/collections/src-md-op

Test Method	For every Instance of a Collection described in the Collections content, issue an HTTP GET request to the URL `/collections/{collectionId}/instances/{instanceId}` where `{collectionId}` is the `id` property for the collection and `{instanceId}` is the `id` property for the instance. . Validate that an Instance of a Collection was returned with a status code 200 . Validate the contents of the returned document using test /conf/instances/src-md-success.

Table B.148 — Abstract Test 135

Abstract Test 135	**/conf/instances/src-md-success**
Test Purpose	Validate that the Collection Instance content complies with the required structure and contents.
Requirement	/req/collections/src-md-success
Test Method	Verify that the content of the response is consistent with the content for this Resource Collection in the `/collections` response. That is, the values for `id`, `title`, `description` and `extent` are identical.

B.10.1.8. Locations

Table B.149 — Abstract Test 136

Abstract Test 136	**/conf/locations**
Test Purpose	Validate that a list of valid locations are returned by a `Locations` query if no query parameters are specified.
Requirement	/req/edr/rc-locations
Test Method	a) No query parameters are specified b) Validate that a GeoJSON document was returned with a status code 200 containing at least a list of features one for each location supported by the collection.

Table B.150 — Abstract Test 137

Abstract Test 137	**/conf/locations**

Test Purpose	Validate that an error is returned by a `Locations` query when the the locationId is invalid.
Requirement	/req/edr/rc-locations
Test Method	a) Check that invalid `locationId` values return an error message b) Validate that a document was returned with a status code 404.

Table B.151 — Abstract Test 138

Abstract Test 138	**/conf/locations**
Test Purpose	Validate that resources can be identified and extracted from a Collection with a `Locations` query using query parameters.
Requirement	/req/edr/rc-locations
Test Method	a) Test with valid query parameters b) Validate that a document was returned with a status code 200. Repeat these tests using the following parameter tests: **Parameters** • Parameter /req/edr/REQ_rc-parameter-name-definition • Response /req/edr/parameter-name-response **DateTime** • Parameter /req/core/datetime-definition • Response /req/core/datetime-response Execute requests with combinations of the “time”,”parameter-name”,”crs” and “f” query parameters and verify that only information that matches the selection criteria is returned.

Table B.152 — Abstract Test 139

Abstract Test 139	**/conf/core/datetime-definition**
Test Purpose	Validate that the datetime query parameters are constructed correctly.
Requirement	/req/core/datetime-definition

Test Method	Verify that the `datetime` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: datetime`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.153 — Abstract Test 140

Abstract Test 140	**/conf/core/datetime-response**
Test Purpose	Validate that the `datetime` query parameters are processed correctly.
Requirement	/req/core/datetime-response
Test Method	a) Verify that only resources that have a temporal geometry that intersects the temporal information in the `datetime` parameter were included in the result set. b) Verify that all resources in the collection that are not associated with a temporal geometry are included in the result set. c) Validate that the `datetime` parameter complies with the syntax described in /req/core/datetime-response.

Table B.154 — Abstract Test 141

Abstract Test 141	**/conf/collections/REQ_rc-parameter-name-definition**
Test Purpose	Validate that the parameter-name query parameters are constructed correctly.
Requirement	/req/edr/REQ_rc-parameter-name-definition

Test Method	Verify that the `parameter-name` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: parameter-name`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.155 — Abstract Test 142

Abstract Test 142	**/conf/edr/rc-parameter-name-response**
Test Purpose	Validate that the parameter-name query parameters are processed correctly.
Requirement	/req/edr/parameter-name-response
Test Method	a) Verify that only resources for the requested parameters were included in the result set b) Validate that the parameter-name parameter complies with the syntax described in /req/edr/parameter-name-response.

Table B.156 — Abstract Test 143

Abstract Test 143	**/conf/edr/REQ_rc-crs-definition**
Test Purpose	Validate that the crs query parameters are constructed correctly.
Requirement	/req/edr/REQ_rc-crs-definition

Test Method	Verify that the `crs` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: crs`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.157 — Abstract Test 144

Abstract Test 144	**/conf/edr/REQ_rc-crs-response**
Test Purpose	Validate that the crs query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-crs-response
Test Method	a) Verify that the geometry of the resources returned are valid for the requested coordinate reference system
Test Method	a) Verify that all crs values defined in the collections metadata are supported by the collection

Test Method	a) Verify that all crs values not defined in the collections metadata will generate a HTTP `400` error b) Validate that the crs parameter complies with the syntax described in /req/edr/REQ_rc-crs-response.

Table B.158 — Abstract Test 145

Abstract Test 145	**/conf/edr/rc-f-definition**
Test Purpose	Validate that the `f` query parameter is constructed correctly.
Requirement	/req/edr/rc-f-definition

Test Method	Verify that the `f` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	`name: f`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Table B.159 — Abstract Test 146

Abstract Test 146	**/conf/collections/rc-f-response**
Test Purpose	Validate that the `f` query parameters are processed correctly.
Requirement	/req/edr/REQ_rc-f-response
Test Method	a) Verify that the response is returned in the requested data format
Test Method	a) Verify that all output format values defined in the collections metadata are supported by the collection b) Validate that the `f` parameter complies with the syntax described in /req/edr/REQ_rc-f-response.

C

ANNEX C (INFORMATIVE) COLLECTION RESPONSE METADATA (INFORMATIVE)

C

ANNEX C (INFORMATIVE) COLLECTION RESPONSE METADATA (INFORMATIVE)

This Annex contains a more-easily human-readable view of the content in the OpenAPI definitions.

The collection response structure provides the details which describe the information available and the query capabilities supported by the collections served by the API. Collection objects describe both collections and instances of a collection.

C.1. EDR Collection Object Structure

Table C.1 — EDR Collection Object Structure

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
links	link Array	**Yes**	Array of Link objects
id	String	**Yes**	Unique identifier string for the collection, used as the value for the collection_id path parameter in all queries on the collection
title	String	No	A short text label for the collection
description	String	No	A text description of the information provided by the collection
keywords	String Array	No	Array of words and phrases that define the information that the collection provides
extent	extent object	**Yes**	Object describing the spatio-temporal extent of the information provided by the collection
data_queries	data_queries object	No	Object providing query specific information

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
crs	String Array	No	Array of coordinate reference system names, which define the output coordinate systems supported by the collection
output_formats	String Array	No	Array of data format names, which define the data formats to which information in the collection can be output
parameter_names	parameter_names object	**Yes**	Describes the data values available in the collection

C.1.1. Link Object

OGC Web API Standards use RFC 8288 (Web Linking) to express relationships between resources. The "link" elements provide a convention for associating resources related to the collection.

Table C.2 — Link Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
href	String	**Yes**	URL being referenced
rel	String	**Yes**	Relation type of the URL. A list of valid relation types can be found at http://www.opengis.net/def/rel
type	String	No	Type of information being returned by the URL
hreflang	String	No	Attribute used to specify the language and geographical targeting of information accessed by the URL. Can be defined by using a value from either languages ISO 639-1 or countries ISO 3166-1
title	String	No	A short text label to describe the URL
length		No	
templated	Boolean	No	If True the URL includes templated values for mandatory Query parameters
variables	variables object	No	Object providing custom information relevant to the link

C.1.2. Variables Object

The variables object provides fields to describe information that only applies to the owning link

Table C.3 — Variables Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
title	String	No	A short text label for the query
description	String	No	A description of the query
query_type	String	No	One of: `position`, `radius`, `area`, `cube`, `trajectory`, `corridor`, `items`, `locations`, `instances`
coords	String	No	An example of valid `coords` query parameter values
within_units	String Array	No	A list of the valid `within` units for radius queries
width_units	String Array	No	A list of the valid `width` units
height_units	String Array	No	A list of the valid `height` units
output_formats	String Array	No	A list of output formats supported by the query, **if this field exists it overrides the output formats definition supplied at a collection level**.
default_output_format	String Array	No	Specifies the default output format for the query
crs_details	crs_details object Array	No	A list of coordinate reference systems supported by the query, **if this field exists it overrides the `crs` values defined at a collection level**.

C.1.3. CRS Details Object

A crs details object describes a coordinate system

Table C.4 — CRS Details Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
crs	String	**Yes**	Name of the coordinate reference system, used as the value in the `crs` query parameter to define the required output coordinate reference system
wkt	String	**Yes**	Well Known Text description of the coordinate reference system

A simple link example

```
"link" : {
    "href": "https://www.example.org/sourcedata/help",
    "hreflang": "en",
    "rel": "service-doc",
    "type": "text/html",
}
```

Figure C.1

A more complex link example supporting a templated `href` as the `coords` parameter is mandatory

```
"link": {
    "href": "http://www.example.org/sourcedata/position?coords={coords}",
    "hreflang": "en",
    "rel": "data",
    "templated": true,
    "variables": {
        "title": "Position query",
        "query_type": "position",
        "output_formats": [
            "CoverageJSON",
            "GeoJSON",
            "IWXXM"
        ],
        "default_output_format": "GeoJSON"
    }
}
```

Figure C.2

C.1.4. Extent Object

The extent object describes the spatio-temporal area covered by the information available in the collection

Table C.5 — Extent Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
spatial	spatial object	**Yes**	Object defining the spatial extent of the information in the collection
temporal	temporal object	No	Object defining the temporal extent of the information in the collection
vertical	vertical object	No	Object defining the vertical extent of the information in the collection

C.1.5. Spatial Object

The spatial object describes the spatial area covered by the information available in the collection

Table C.6 — Spatial Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
bbox	Number Array	**Yes**	A bounding box is provided as four numbers: • Lower left corner, coordinate axis 1 • Lower left corner, coordinate axis 2 • Upper right corner, coordinate axis 1 • Upper right corner, coordinate axis 2
crs	String	**Yes**	This can either be a Well Known Text definition of the CRS or follow a convention of http://www.opengis.net/def/crs/{authority}/{version}/{code} where the token **{authority}** is a placeholder for a code the designates to authority responsible for the definition of this CRS. Typical values include “EPSG” and “OGC”.The token **{version}** is a placeholder for the specific version of the coordinate reference system definition or 0 for the latest version or if the version is unknown.The token **{code}** is a placeholder for the authority’s code for the CRS.

C.1.6. Temporal Object

The temporal object describes the time period covered by the information available in the collection

Table C.7 — Temporal Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
interval	ISO 8601 Date Array	**Yes**	An array of ISO 8601 datestrings, each member of the the array can either be a single time, an ISO8601 time interval or an ISO8601 time duration (see https://en.wikipedia.org/wiki/ISO_8601)
trs	String	**Yes**	This defaults to Gregorian, but other temporal systems can be supported following the conventions defined by the Well Known Text standard.

C.1.7. Vertical Object

The vertical object describes the vertical extent of information available in the collection

Table C.8 — Vertical Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
interval	String Array	**Yes**	Array of height values supported by the collection.
vrs	String	**Yes**	Follows the conventions defined by the Well Known Text standard.

A simple extent object example for collection with no vertical or temporal dimensions

```
"extent": {
    "spatial": {
        "bbox": [1393.0196, 13494.9764, 671196.3657, 1230275.0454],
        "crs": "PROJCS[\"OSGB 1936 / British National Grid\",
        GEOGCS[\"OSGB 1936\",DATUM[\"OSGB_1936\",
        SPHEROID[\"Airy 1830\",6377563.396,299.3249646,
        AUTHORITY[\"EPSG\",\"7001\"]],AUTHORITY[\"EPSG\",\"6277\"]],
        PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],
        UNIT[\"degree\",0.01745329251994328,
        AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",\"4277\"]],
        UNIT[\"metre\",1,AUTHORITY[\"EPSG\",\"9001\"]],
        PROJECTION[\"Transverse_Mercator\"],
        PARAMETER[\"latitude_of_origin\",49],PARAMETER[\"central_meridian\",-
2],
```

```
        PARAMETER[\"scale_factor\",0.9996012717],PARAMETER[\"false_easting
\",400000],
        PARAMETER[\"false_northing\",-100000],AUTHORITY[\"EPSG\",\"27700\"],
        AXIS[\"Easting\",EAST],AXIS[\"Northing\",NORTH]]"
    }
}
```

Figure C.3

This more complex extent object example is a collection with vertical and temporal dimensions

```
"extent": {
    "spatial": {
        "bbox": [-180.0,-90.0,180.0,90.0],
        "crs": "GEOGCS[\"WGS 84\",DATUM[\"WGS_1984\",
        SPHEROID[\"WGS 84\",6378137,298.257223563,
        AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",\"6326\"]],
        PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],
        UNIT[\"degree\",0.01745329251994328,
        AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",\"4326\"]]"
    },
    "temporal": {
        "interval": ["R82/2021-04-22T00:00:00Z/PT3H",
                     "R11/2021-05-02T12:00:00Z/PT12H"],
        "trs": "TIMECRS[\"DateTime\",TDATUM[\"Gregorian Calendar\"],
               CS[TemporalDateTime,1],AXIS[\"Time (T)\",future]"
    },
    "vertical": {
        "interval": ["1829.0","2743.0","3658.0"],
        "vrs": "VERT_CS['MSL height',
               VERT_DATUM['Mean Sea Level',2005,
               AUTHORITY['EPSG','5100']],
               UNIT['metre',1,AUTHORITY['EPSG','9001']],
               AXIS['Up',UP],AUTHORITY['EPSG','5714']]"
    }
}
```

Figure C.4

C.1.8. Data Queries Object

The data queries object provides the extra metadata required for the queries supported by the collection.

Table C.9 — Data Queries Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
position	EDRQuery object	No	Position query metadata
radius	EDRQuery object	No	Radius query metadata
area	EDRQuery object	No	Area query metadata
cube	EDRQuery object	No	Cube query metadata

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
trajectory	EDRQuery object	No	Trajectory query metadata
corridor	EDRQuery object	No	Corridor query metadata
item	EDRQuery object	No	Item query metadata
location	EDRQuery object	No	Location query metadata

C.1.9. EDR Query Object

The EDR query object provides the metadata for the specified query type.

Table C.10 — EDR Query Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
link	Link object	**Yes**	Array of height values supported by the collection.

A data query object example for a collection that supports Position and Radius queries

```
"data_queries": {
    "position": {
        "link": {
            "href": "http://www.example.org/collections/sampledata/position",
            "hreflang": "en",
            "rel": "data",
            "templated":false,
            "variables": {
                "title": "Position query",
                "query_type": "position",
                "output_formats": [
                    "CoverageJSON",
                    "GeoJSON"
                ],
                "default_output_format": "GeoJSON",
                "crs_details": [
                {
                    "crs": "CRS84",
                    "wkt": "GEOGCS[\"WGS 84\",DATUM[\"WGS_1984\",
                    SPHEROID[\"WGS 84\",6378137,298.257223563,
                    AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],
                    PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],
                    UNIT[\"degree\",0.01745329251994328,AUTHORITY[\"EPSG\",
\"9122\"]],
                    AUTHORITY[\"EPSG\",\"4326\"]]"
                }
                ]
            }
        }
```

```
    },
    "radius": {
        "link": {
            "href": "http://www.example.org/collections/sampledata/radius",
            "hreflang": "en",
            "rel": "data",
            "variables": {
                "title": "Radius query",
                "description": "Radius query",
                "query_type": "radius",
                "output_formats": [
                "CoverageJSON",
                "GeoJSON",
                "GeoTiff"
                ],
                "default_output_format": "CoverageJSON",
                "within_units": [
                    "km",
                    "miles"
                ],
                "crs_details": [
                {
                    "crs": "CRS84",
                    "wkt": "GEOGCS[\"WGS 84\",DATUM[\"WGS_1984\",
                    SPHEROID[\"WGS 84\",6378137,298.257223563,
                    AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],
                    PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],
                    UNIT[\"degree\",0.01745329251994328,AUTHORITY[\"EPSG\",
\"9122\"]],
                    AUTHORITY[\"EPSG\",\"4326\"]]"           }
                ]
            }
        }
    }
}
```


Figure C.5

C.1.10. Parameter Names Object

The parameter-names object provides information about the data parameters supported by the collection. As a set of key-value pairs, where the key is the name of the parameter and the value is a Parameter object i.e. as a Dictonary (Python) or HashMap(Java).

C.1.11. Parameter Object

Table C.11 — Parameter Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
id	String	**Yes**	parameter id
type	String	**Yes**	Always 'Parameter'

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
label	String	No	A short text label for the parameter
description	String	No	A description of the parameter
data-type	String	No	The data type of the parameter values [integer, float, string]
unit	unit object	No	A description of the units of the parameter values
observedProperty	observedProperty object	**Yes**	A formal definition of the parameter
extent	Extent object	No	Information on the spatio-temporal extent of the parameter values (if different from other parameters in the collection)
measurementType	measurementType object	No	Information on how the value was derived

C.1.12. Unit Object

The unit object provides the information to describe the units of measure of the parameter values.

Table C.12 — Unit Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
label	String	**Yes**	Name of the unit
symbol	symbol object	**Yes**	Information to describe the symbols used to represent the unit

C.1.13. Symbol Object

The symbol object provides the information to describe the symbols which represent the unit of a value.

Table C.13 — Symbol Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
title	String	No	Symbol name

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
description	String	No	A text description of the symbol
value	String	No	A Unicode representation for the symbol
type	String	No	A URI to a registry entry providing more detailed information about the unit (i.e. QUDT is one example of a registry that provide links for many common units)

C.1.14. Observed Property Object

The observedProperty object provides the metadata for the specified query type.

Table C.14 — Observed Property Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
id	String	No	URI linking to an external registry which contains the definitive definition of the observed property
label	String	**Yes**	A short text label for the property
description	String	No	A description of the observed property

C.1.15. Measurement Type object

The measurementType object provides basic information about how the parameter is calculated and over what time period

Table C.15 — Measurement Type object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
method	String	**Yes**	Calculation method e.g. Mean, Sum, Max, etc.
duration	String	**Yes**	Duration of calculation. For time durations, this follows the ISO8601 Duration standard. • **A negative sign before a duration value (i.e. -PT10M) infers that the time start starts at the specified**

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
			duration before the time value assigned to the parameter value. • So if the measurement had a time value of **2020-04-05T14:30Z** and a measurementType duration of **-PT10M** the value is representative of the period **2020-04-05T14:20Z/2020-04-05T14:30Z**; if the measurement had a time value of **2020-04-05T14:30Z** and a measurement Type duration of **PT10M** the value is representative of the period **2020-04-05T14:30Z/2020-04-05T14:40Z**

Parameter names example

```
"parameter_names": {
    "Temperature_altitude_above_msl": {
        "type": "Parameter",
        "description": "Temperature for Specific altitude above MSL",
        "unit": {
        "label": "K",
        "symbol": {
            "value": "K",
            "type": "http://qudt.org/vocab/unit/K"
        }
        },
        "observedProperty": {
        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-0-0",
        "label": "Temperature_altitude_above_msl"
        },
        "measurementType": {
        "method": "instantaneous",
        "period": "PT0S"
        }
    },
    "u-component_of_wind_altitude_above_msl": {
        "type": "Parameter",
        "description": "u-component of wind for Specific altitude above MSL",
        "unit": {
        "label": "m/s",
        "symbol": {
            "value": "m%20s",
            "type": "http://qudt.org/vocab/unit/M-PER-SEC.html"
        }
        },
        "observedProperty": {
            "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-2",
            "label": "u-component_of_wind_altitude_above_msl"
        },
        "measurementType": {
            "method": "instantaneous",
            "period": "PT0S"
        }
    },
```

```
    "v-component_of_wind_altitude_above_msl": {
        "type": "Parameter",
        "description": "v-component of wind for Specific altitude above MSL",
        "unit": {
        "label": "m/s",
        "symbol": {
            "value": "m%20s",
            "type": "http://qudt.org/vocab/unit/M-PER-SEC.html"
        }
        },
        "observedProperty": {
            "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-3",
            "label": "v-component_of_wind_altitude_above_msl"
        },
        "measurementType": {
            "method": "instantaneous",
            "period": "PT0S"
        }
    }
    }
}
```


Figure C.6

D

ANNEX D (INFORMATIVE) EXAMPLES (INFORMATIVE)

D

ANNEX D (INFORMATIVE) EXAMPLES (INFORMATIVE)

D.1. Example Landing Pages

Example — JSON Landing Page:

```
{
  "links": [
    { "href": "http://data.example.org/",
      "rel": "self", "type": "application/json", "title": "this document" },
    { "href": "http://data.example.org/api",
      "rel": "service-doc", "type": "application/vnd.oai.openapi+json;version=
3.0", "title": "the API definition" },
    { "href": "http://data.example.org/conformance",
      "rel": "conformance", "type": "application/json", "title": "OGC
 conformance classes implemented by this API" },
    { "href": "http://data.example.org/collections",
      "rel": "data", "type": "application/json", "title": "Metadata about the
 resource collections" }
  ]
}
```

D.2. API Description Examples

The API is described using the OpenAPI 3.0 specification, example responses for a server which supports all possible EDR query patterns can be found at:

YAML OpenAPI document

D.3. Conformance Examples

Example — Conformance Response: This example response in JSON is for an OGC API — EDR that supports OpenAPI 3.0 for the API definition and HTML and GeoJSON as encodings for resources.

```
{
  "conformsTo": [
    "http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core",
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/core",
    "http://www.opengis.net/spec/ogcapi-common-2/1.0/conf/collections",
    "http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/oas30",
    "http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/html",
    "http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/geojson"
  ]
}
```


D.4. Collections Metadata Examples

Example — Collections metadata response document: The example below shows a service with two collections, one for observations and another for forecast data. The forecast data is regenerated every hour so the collection provides access to multiple instances of the collection via an instances endpoint. There are links to the responses of the collections (link relation type: “self”). Representations of these resource in other formats are referenced using link relation type “alternate”. The data queries that are supported by each collection are referenced using link relation type “data”. There are also links to the license information for the observation and forecast data (using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] “license”) and also the terms and conditions of service (using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] “restrictions”) .


```
{
    "links": [
        {
            "href": "http://www.example.org/edr/collections/",
            "hreflang": "en",
            "rel": "self",
            "type": "application/json"
        },
        {
            "href": "http://www.example.org/edr/collections/",
            "hreflang": "en",
            "rel": "alternate",
            "type": "text/html"
        },
        {
            "href": "http://www.example.org/edr/collections/",
            "hreflang": "en",
            "rel": "alternate",
            "type": "application/xml"
        }
    ],
    "collections": [
        {
            "id": "hourly observations",
            "title": "Hourly Site Specific observations",
            "description": "Observation data for UK observing sites",
            "keywords": [
                "Wind Direction",
                "Wind Speed",
                "Wind Gust",
```

```
                "Air Temperature",
                "Weather",
                "Relative Humidity",
                "Dew point",
                "Pressure",
                "Pressure Tendancy",
                "Visibility"
            ],
            "links": [
                {
                    "href": "http://www.example.org/uk-hourly-site-specific-
observations",
                    "hreflang": "en",
                    "rel": "service-doc",
                    "type": "text/html"
                },
                {
                    "href": "http://www.example.org/terms-and-conditions---
datapoint#datalicence",
                    "hreflang": "en",
                    "rel": "license",
                    "type": "text/html"
                },
                {
                    "href": "https://www.metoffice.gov.uk/services/data/
datapoint/terms-and-conditions---datapoint#termsofservice",
                    "hreflang": "en",
                    "rel": "restrictions",
                    "type": "text/html"
                },
                {
                    "href": "http://www.example.org/edr/collections/hrly_obs/
position",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
                    "href": "http://www.example.org/edr/collections/hrly_obs/
radius",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
                    "href": "http://www.example.org/edr/collections/hrly_obs/
area",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
                    "href": "http://www.example.org/edr/collections/hrly_obs/
locations",
                    "hreflang": "en",
                    "rel": "data"
                }
            ],
            "extent": {
                "spatial": {
                    "bbox": [
                        -15.0,
                        48.0,
                        5.0,
                        62.0
                    ],
```

```
                    "crs": "GEOGCS[\"WGS 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS
 84\",6378137,298.257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                },
                "temporal": {
                    "interval": [
                        "2020-04-19T11:00:00Z/2020-06-30T09:00:00Z"
                    ],
                    "trs": "TIMECRS[\"DateTime\",TDATUM[\"Gregorian Calendar
\"],CS[TemporalDateTime,1],AXIS[\"Time (T)\",future]"
                }
            },
            "data_queries" : {
                "position": {
                    "link": {
                        "href": "http://www.example.org/edr/collections/hrly_
obs/position?coords={coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": true,
                        "variables": {
                            "title": "Position query",
                            "description": "Position query",
                            "query_type": "position",
                            "coords" :{
                                "description": "Well Known Text POINT value i.
e. POINT(-120, 55)"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON",
                                "IWXXM"
                            ],
                            "default_output_format": "IWXXM",
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
                "radius": {
                    "link": {
                        "href": "http://www.example.org/edr/collections/hrly_
obs/radius?coords={coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": true,
                        "variables": {
                            "title": "Radius query",
                            "description": "Radius query",
                            "query_type": "radius",
                            "coords" :{
```

```
                                "description": "Well Known Text POINT value i.
e. POINT(-120, 55)"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON",
                                "IWXXM"
                            ],
                            "default_output_format": "GeoJSON",
                            "within_units": [
                                "km",
                                "miles"
                            ],
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
                "area": {
                    "link":                 {
                        "href": "http://www.example.org/edr/collections/hrly_
obs/area?coords={coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": true,
                        "variables": {
                            "title": "Area query",
                            "description": "Area query",
                            "query_type": "area",
                            "coords" :{
                                "description": "Well Known Text POLYGON value
 i.e. POLYGON((-79 40,-79 38,-75 38,-75 41,-79 40))"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON",
                                "BUFR",
                                "IWXXM"
                            ],
                            "default_output_format": "CoverageJSON",
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
```

```
                "locations": {
                    "link": {
                        "href": "http://www.example.org/edr/collections/hrly_
obs/locations",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": false,
                        "variables": {
                            "title": "Location query",
                            "description": "Location query",
                            "query_type": "locations",
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON",
                                "BUFR",
                                "IWXXM"
                            ],
                            "default_output_format": "CoverageJSON",
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                }
            },
            "crs": [
                "CRS84"
            ],
            "output_formats": [
                "CoverageJSON",
                "GeoJSON",
                "IWXXM"
            ],
            "parameter_names": {
                "Wind Direction": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "degree true",
                        "symbol": {
                            "value": "°",
                            "type": "http://www.example.org/edr/metadata/units/
degree"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
windDirection",
                        "label":  "Wind Direction"
                    },
                    "measurementType": {
                        "method": "mean",
                        "period": "-PT10M/PT0M"
                    }
                },
```

```
                "Wind Speed": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "mph",
                        "symbol": {
                            "value": "mph",
                            "type": "http://www.example.org/edr/metadata/units/
mph"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
windSpeed",
                        "label": "Wind Speed"
                    },
                    "measurementType": {
                        "method": "mean",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Wind Gust": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "mph",
                        "symbol": {
                            "value": "mph",
                            "type": "http://www.example.org/edr/metadata/units/
mph"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
maximumWindGustSpeed",
                        "label": "Wind Gust"
                    },
                    "measurementType": {
                        "method": "maximum",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Air Temperature": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label":  "degC",
                        "symbol": {
                            "value": "°C",
                            "type": "http://www.example.org/edr/metadata/units/
degC"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
airTemperature",
                        "label": "Air Temperature"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
```

```
                "Weather": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "weather",
                        "symbol": {
                            "value": "",
                            "type": "http://www.example.org/edr/metadata/
lookup/mo_dp_weather"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/wmdr/
ObservedVariableAtmosphere/_266",
                        "label": "Weather"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Relative Humidity": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "percent",
                        "symbol": {
                            "value": "%",
                            "type": "http://www.example.org/edr/metadata/units/
percent"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/bufr4/b/13/_009",
                        "label": "Relative Humidity"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Dew point": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "degC",
                        "symbol": {
                            "value": "°C",
                            "type": "http://www.example.org/edr/metadata/units/
degC"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
dewPointTemperature",
                        "label": "Dew point"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Pressure": {
```

```
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "hPa",
                        "symbol": {
                            "value": "hPa",
                            "type": "http://www.example.org/edr/metadata/units/
hPa"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/bufr4/b/10/_051",
                        "label": "Pressure"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Pressure Tendancy": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "tendency",
                        "symbol": {
                            "value": "",
                            "type": "http://www.example.org/edr/metadata/units/
hPa"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
pressureTendency",
                        "label": "Pressure Tendancy"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Visibility": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "m",
                        "symbol": {
                            "value": "m",
                            "type": "http://www.example.org/edr/metadata/units/
m"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
horizontalVisibility",
                        "label": "Visibility"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                }
            }
        },
```

```
        {
            "id": "UK 3 hourly forecast",
            "title": "UK 3 Hourly Site Specific Forecast",
            "description": "Five day site specific forecast for 6000 UK
 locations",
            "keywords": [
                "Wind Direction",
                "Wind Speed",
                "Wind Gust",
                "Air Temperature",
                "Weather",
                "Relative Humidity",
                "Feels like temperature",
                "UV index",
                "Probabilty of precipitation",
                "Visibility"
            ],
            "links": [
                {
                    "href": "https://www.example.org/uk-3-hourly-site-specific-
forecast",
                    "hreflang": "en",
                    "rel": "service-doc",
                    "type": "text/html"
                },
                {
                    "href": "https://www.example.org/terms-and-conditions---
datapoint#datalicence",
                    "hreflang": "en",
                    "rel": "licence",
                    "type": "text/html"
                },
                {
                    "href": "https://www.example.org/terms-and-conditions---
datapoint#termsofservice",
                    "hreflang": "en",
                    "rel": "restrictions",
                    "type": "text/html"
                },
                {
                    "href": "http://www.example.org/edr/collections/3_hrly_
fcst/instances",
                    "hreflang": "en",
                    "rel": "collection"
                }
            ],
            "extent": {
                "spatial": {
                    "bbox": [
                        -15.0,
                        48.0,
                        5.0,
                        62.0
                    ],
                    "crs": "GEOGCS[\"WGS 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS
 84\",6378137,298.257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                },
                "temporal": {
                    "interval": [
                        "2020-06-23T18:00:00Z/2020-07-04T21:00:00Z"
```

```
                    ],
                    "trs": "TIMECRS[\"DateTime\",TDATUM[\"Gregorian Calendar
\"],CS[TemporalDateTime,1],AXIS[\"Time (T)\",future]"
                }
            },
            "crs": [
                "CRS84"
            ],
            "output_formats": [
                "CoverageJSON",
                "GeoJSON"
            ],
            "parameter_names": {
                "Wind Direction": {
                    "type": "Parameter",
                    "description": "Direction wind is from",
                    "unit": {
                        "label": "degree true",
                        "symbol": {
                            "value": "°",
                            "type": "http://www.example.org/edr/metadata/units/
degree"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-0",
                        "label": "Wind Direction"
                                        },
                    "measurementType": {
                        "method": "mean",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Wind Speed": {
                    "type": "Parameter",
                    "description":"Average wind speed",
                    "unit": {
                        "label": "mph",
                        "symbol": {
                            "value": "mph",
                            "type": "http://www.example.org/edr/metadata/units/
mph"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-1",
                        "label":  "Wind Speed"
                    },
                    "measurementType": {
                        "method": "mean",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Wind Gust": {
                    "type": "Parameter",
                    "description":  "Wind gusts are a rapid increase in
 strength of the wind relative to the wind speed.",
                    "unit": {
                        "label": "mph",
                        "symbol": {
                            "value": "mph",
                            "type": "http://www.example.org/edr/metadata/units/
mph"
```

```
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-1",
                        "label": "Wind Gust"
                    },
                    "measurementType": {
                        "method": "maximum",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Air Temperature": {
                    "type": "Parameter",
                    "description":  "2m air temperature in the shade and out of
 the wind",
                    "unit": {
                        "label":  "degC",
                        "symbol": {
                            "value": "°C",
                            "type": "http://www.example.org/edr/metadata/units/
degC"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
airTemperature",
                        "label": "Air Temperature"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Weather": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label":  "weather",
                        "symbol": {
                            "value": "",
                            "type": "http://www.example.org/edr/metadata/
lookup/mo_dp_weather"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/wmdr/
ObservedVariableAtmosphere/_266",
                        "label": "Weather"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Relative Humidity": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "percent",
                        "symbol": {
                            "value": "%",
                            "type": "http://www.example.org/edr/metadata/units/
percent"
```

```
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-1-1",
                        "label":  "Relative Humidity"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Feels like temperature": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "degC",
                        "symbol": {
                            "value": "°C",
                            "type": "http://www.example.org/edr/metadata/units/
degC"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
airTemperature",
                        "label": "Feels like temperature"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "UV index": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label":  "UV_index",
                        "symbol": {
                            "value": "",
                            "type": "http://www.example.org/edr/metadata/
lookup/mo_dp_uv"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-4-
51",
                        "label": "UV index"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Probabilty of precipitation": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "percent",
                        "symbol": {
                            "value": "%",
                            "type": "http://www.example.org/edr/metadata/units/
percent"
                        }
```

```
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-1-1",
                        "label": "Probabilty of precipitation"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Visibility": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "quality",
                        "symbol": {
                            "value": "",
                            "type": "http://www.example.org/edr/metadata/
lookup/mo_dp_visibility"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
horizontalVisibility",
                        "label":  "Visibility"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                }
            }
        }
    ]
}
```


D.5. Instance Metadata Examples

Example — Collection instance metadata response document: This is an example of the metadata returned by the instances query (link relation type: “items”). There is a link to the instance response itself (link relation type: “self”). Representations of this resource in other formats are referenced using link relation type “alternate”. The data queries that are supported by each instance are referenced using link relation type “data”. There are also links to the license information for the observation and forecast data (using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] “license”) and also the terms and conditions of service (using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] “restrictions”) .


```
{
    "links": [
        {
            "href": "http://http://www.example.org/edr/collections/3_hrly_fcst/
instances/",
            "hreflang": "en",
            "rel": "self",
            "type": "application/json"
```

```
        },
        {
            "href": "http://http://www.example.org/edr/collections/3_hrly_fcst/
instances/",
            "hreflang": "en",
            "rel": "alternate",
            "type": "text/html"
        },
        {
            "href": "http://http://www.example.org/edr/collections/3_hrly_fcst/
instances/",
            "hreflang": "en",
            "rel": "alternate",
            "type": "application/xml"
        },
        {
            "href": "https://http://www.example.org/terms-and-conditions---
datapoint#termsofservice",
            "hreflang": "en",
            "rel": "restrictions",
            "type": "text/html",
            "title": ""
        },
        {
            "href": "https://http://www.example.org/terms-and-conditions---
datapoint#datalicence",
            "hreflang": "en",
            "rel": "license",
            "type": "text/html",
            "title": ""
        },
        {
            "href": "https://http://www.example.org/uk-3-hourly-site-specific-
forecast",
            "hreflang": "en",
            "rel": "service-doc",
            "type": "text/html",
            "title": ""
        }
    ],
    "instances": [
        {
            "id": "2020-06-30T10:00:00Z",
            "title": "3 hrly fcst",
            "description": "Five day site specific forecast for 6000 UK
 locations 3 hrly fcst",
            "keywords": [
                "Wind Direction",
                "Wind Speed",
                "Wind Gust",
                "Air Temperature",
                "Weather",
                "Relative Humidity",
                "Feels like temperature",
                "UV index",
                "Probabilty of precipitation",
                "Visibility"
            ],
            "links": [
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/position",
                    "hreflang": "en",
```

```
                    "rel": "data"
                },
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/radius",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/area",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/locations",
                    "hreflang": "en",
                    "rel": "data"
                }
            ],
            "extent": {
                "spatial": {
                    "bbox": [
                        -15.0,
                        48.0,
                        5.0,
                        62.0
                    ],
                    "crs": "GEOGCS[\"WGS 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS
 84\",6378137,298.257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                },
                "temporal": {
                    "interval": [
                        "2020-06-30T06:00:00Z/2020-07-04T21:00:00Z"
                    ],
                    "trs": "TIMECRS[\"DateTime\",TDATUM[\"Gregorian Calendar
\"],CS[TemporalDateTime,1],AXIS[\"Time (T)\",future]"
                }
            },
            "data_queries": {
                "position": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/position?coords=
{coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "title": "",
                        "templated": true,
                        "variables": {
                            "title": "Position query",
                            "description": "Position query",
                            "query_type": "position",
                            "coords": {
                                "description": "Well Known Text POINT value i.
e. POINT(-120, 55)"
                            },
                            "output_formats": [
                                "CoverageJSON",
```

```
                                "GeoJSON"
                            ],
                            "default_output_format": "GeoJSON",
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
                "radius": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/radius?coords={coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": true,
                        "variables": {
                            "title": "Radius query",
                            "description": "Radius query",
                            "query_type": "radius",
                            "coords": {
                                "description": "Well Known Text POINT value i.
e. POINT(-120, 55)"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON",
                                "CSV"
                            ],
                            "default_output_format": "GeoJSON",
                            "within_units": [
                                "km",
                                "miles"
                            ],
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
                "area": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/area?coords={coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": true,
```

```
                        "variables": {
                            "title": "Area query",
                            "description": "Area query",
                            "query_type": "area",
                            "coords": {
                                "description": "Well Known Text POLYGON value
 i.e. POLYGON((-79 40,-79 38,-75 38,-75 41,-79 40))"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON",
                                "CSV"
                            ],
                            "default_output_format": "CoverageJSON",
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
                "locations": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/locations",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": false,
                        "variables": {
                            "title": "Locations query",
                            "description": "Locations query",
                            "query_type": "location",
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON"
                            ],
                            "default_output_format": "GeoJSON",
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                }
            },
            "crs": [
                "CRS84"
            ],
            "output_formats": [
```

```
                "GeoJSON",
                "CoverageJSON",
                "CSV"
            ],
            "parameter_names": {
                "Wind Direction": {
                    "type": "Parameter",
                    "description": "Direction wind is from",
                    "unit": {
                        "label": "degree true",
                        "symbol": {
                            "value": "°",
                            "type": "http://http://www.example.org/edr/
metadata/units/degree"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-0",
                        "label": "Wind Direction"
                    },
                    "measurementType": {
                        "method": "mean",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Wind Speed": {
                    "type": "Parameter",
                    "description": "Average wind speed",
                    "unit": {
                        "label": "mph",
                        "symbol": {
                            "value": "mph",
                            "type": "http://http://www.example.org/edr/
metadata/units/mph"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-1",
                        "label": "Wind Speed"
                    },
                    "measurementType": {
                        "method": "mean",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Wind Gust": {
                    "type": "Parameter",
                    "description": "Wind gusts are a rapid increase in strength
 of the wind relative to the wind speed.",
                    "unit": {
                        "label": "mph",
                        "symbol": {
                            "value": "mph",
                            "type": "http://http://www.example.org/edr/
metadata/units/mph"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-1",
                        "label": "Wind Gust"
                    },
                    "measurementType": {
                        "method": "maximum",
```

```
                        "period": "-PT10M/PT0M"
                    }
                },
                "Air Temperature": {
                    "type": "Parameter",
                    "description": "2m air temperature in the shade and out of
 the wind",
                    "unit": {
                        "label": "degC",
                        "symbol": {
                            "value": "°C",
                            "type": "http://http://www.example.org/edr/
metadata/units/degC"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
airTemperature",
                        "label": "Air Temperature"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Weather": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "weather",
                        "symbol": {
                            "value": "",
                            "type": "http://http://www.example.org/edr/
metadata/lookup/mo_dp_weather"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/wmdr/
ObservedVariableAtmosphere/_266",
                        "label": "Weather"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Relative Humidity": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "percent",
                        "symbol": {
                            "value": "%",
                            "type": "http://http://www.example.org/edr/
metadata/units/percent"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-1-1",
                        "label": "Relative Humidity"
                    },
                    "measurementType": {
                        "method": "instantaneous",
```

```
                        "period": "PT0M"
                    }
                },
                "Feels like temperature": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "degC",
                        "symbol": {
                            "value": "°C",
                            "type": "http://http://www.example.org/edr/
metadata/units/degC"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
airTemperature",
                        "label": "Feels like temperature"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "UV index": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "UV_index",
                        "symbol": {
                            "value": "",
                            "type": "http://http://www.example.org/edr/
metadata/lookup/mo_dp_uv"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-4-
51",
                        "label": "UV index"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Probabilty of precipitation": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "percent",
                        "symbol": {
                            "value": "%",
                            "type": "http://http://www.example.org/edr/
metadata/units/percent"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-1-1",
                        "label": "Probabilty of precipitation"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
```

```
                    }
                },
                "Visibility": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "quality",
                        "symbol": {
                            "value": "",
                            "type": "http://http://www.example.org/edr/
metadata/lookup/mo_dp_visibility"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
horizontalVisibility",
                        "label": "Visibility"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                }
            }
        },
        {
            "id": "2020-06-30T09:00:00Z",
            "title": "3 hrly fcst",
            "description": "Five day site specific forecast for 6000 UK
 locations 3 hrly fcst",
            "keywords": [
                "Wind Direction",
                "Wind Speed",
                "Wind Gust",
                "Air Temperature",
                "Weather",
                "Relative Humidity",
                "Feels like temperature",
                "UV index",
                "Probabilty of precipitation",
                "Visibility"
            ],
            "links": [
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/position",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/radius",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/area",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
```

```
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/locations",
                    "hreflang": "en",
                    "rel": "data"
                }
            ],
            "extent": {
                "spatial": {
                    "bbox": [
                        -15.0,
                        48.0,
                        5.0,
                        62.0
                    ],
                    "crs": "GEOGCS[\"WGS 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS
 84\",6378137,298.257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                },
                "temporal": {
                    "interval": [
                        "2020-06-30T06:00:00Z/2020-07-04T21:00:00Z"
                    ],
                    "trs": "TIMECRS[\"DateTime\",TDATUM[\"Gregorian Calendar
\"],CS[TemporalDateTime,1],AXIS[\"Time (T)\",future]"
                }
            },
            "data_queries": {
                "position": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/position?coords=
{coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": true,
                        "variables": {
                            "title": "Position query",
                            "description": "Position query",
                            "query_type": "position",
                            "coords": {
                                "description": "Well Known Text POINT value i.
e. POINT(-120, 55)"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON"
                            ],
                            "default_output_format": "GeoJSON",
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
```

```
                },
                "radius": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/radius?coords={coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": true,
                        "variables": {
                            "title": "Radius query",
                            "description": "Radius query",
                            "query_type": "radius",
                            "coords": {
                                "description": "Well Known Text POINT value i.
e. POINT(-120, 55)"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON",
                                "CSV"
                            ],
                            "default_output_format": "GeoJSON",
                            "within_units": [
                                "km",
                                "miles"
                            ],
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
                "area": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/area?coords={coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "title": "",
                        "templated": true,
                        "variables": {
                            "title": "Area query",
                            "description": "Area query",
                            "query_type": "area",
                            "coords": {
                                "description": "Well Known Text POLYGON value
 i.e. POLYGON((-79 40,-79 38,-75 38,-75 41,-79 40))"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON",
                                "CSV"
                            ],
                            "default_output_format": "CoverageJSON",
                            "crs_details": [
```

```
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
                "locations": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/locations",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": false,
                        "variables": {
                            "title": "Locations query",
                            "description": "Locations query",
                            "query_type": "location",
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON"
                            ],
                            "default_output_format": "GeoJSON",
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                }
            },
            "crs": [
                "CRS84"
            ],
            "output_formats": [
                "GeoJSON",
                "CoverageJSON",
                "CSV"
            ],
            "parameter_names": {
                "Wind Direction": {
                    "type": "Parameter",
                    "description": "Direction wind is from",
                    "unit": {
                        "label": "degree true",
                        "symbol": {
                            "value": "°",
                            "type": "http://http://www.example.org/edr/
metadata/units/degree"
                        }
```

```
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-0",
                        "label": "Wind Direction"
                    },
                    "measurementType": {
                        "method": "mean",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Wind Speed": {
                    "type": "Parameter",
                    "description": "Average wind speed",
                    "unit": {
                        "label": "mph",
                        "symbol": {
                            "value": "mph",
                            "type": "http://http://www.example.org/edr/
metadata/units/mph"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-1",
                        "label": "Wind Speed"
                    },
                    "measurementType": {
                        "method": "mean",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Wind Gust": {
                    "type": "Parameter",
                    "description": "Wind gusts are a rapid increase in strength
 of the wind relative to the wind speed.",
                    "unit": {
                        "label": "mph",
                        "symbol": {
                            "value": "mph",
                            "type": "http://http://www.example.org/edr/
metadata/units/mph"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-1",
                        "label": "Wind Gust"
                    },
                    "measurementType": {
                        "method": "maximum",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Air Temperature": {
                    "type": "Parameter",
                    "description": "2m air temperature in the shade and out of
 the wind",
                    "unit": {
                        "label": "degC",
                        "symbol": {
                            "value": "°C",
                            "type": "http://http://www.example.org/edr/
metadata/units/degC"
                        }
                    },
```

```
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
airTemperature",
                        "label": "Air Temperature"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Weather": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "weather",
                        "symbol": {
                            "value": "",
                            "type": "http://http://www.example.org/edr/
metadata/lookup/mo_dp_weather"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/wmdr/
ObservedVariableAtmosphere/_266",
                        "label": "Weather"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Relative Humidity": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "percent",
                        "symbol": {
                            "value": "%",
                            "type": "http://http://www.example.org/edr/
metadata/units/percent"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-1-1",
                        "label": "Relative Humidity"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Feels like temperature": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "degC",
                        "symbol": {
                            "value": "°C",
                            "type": "http://http://www.example.org/edr/
metadata/units/degC"
                        }
                    },
                    "observedProperty": {
```

```
                        "id": "http://codes.wmo.int/common/quantity-kind/_
airTemperature",
                        "label": "Feels like temperature"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "UV index": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "UV_index",
                        "symbol": {
                            "value": "",
                            "type": "http://http://www.example.org/edr/
metadata/lookup/mo_dp_uv"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-4-
51",
                        "label": "UV index"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Probabilty of precipitation": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "percent",
                        "symbol": {
                            "value": "%",
                            "type": "http://http://www.example.org/edr/
metadata/units/percent"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-1-1",
                        "label": "Probabilty of precipitation"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Visibility": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "quality",
                        "symbol": {
                            "value": "",
                            "type": "http://http://www.example.org/edr/
metadata/lookup/mo_dp_visibility"
                        }
                    },
                    "observedProperty": {
```

```
                        "id": "http://codes.wmo.int/common/quantity-kind/_
horizontalVisibility",
                        "label": "Visibility"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                }
            }
        },
        {
            "id": "2020-06-30T08:00:00Z",
            "title": "3 hrly fcst",
            "description": "Five day site specific forecast for 6000 UK
 locations 3 hrly fcst",
            "keywords": [
                "Wind Direction",
                "Wind Speed",
                "Wind Gust",
                "Air Temperature",
                "Weather",
                "Relative Humidity",
                "Feels like temperature",
                "UV index",
                "Probabilty of precipitation",
                "Visibility"
            ],
            "links": [
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/position",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/radius",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/area",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/locations",
                    "hreflang": "en",
                    "rel": "data"
                }
            ],
            "extent": {
                "spatial": {
                    "bbox": [
                        -15.0,
                        48.0,
                        5.0,
                        62.0
                    ],
```

```
                    "crs": "GEOGCS[\"WGS 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS
 84\",6378137,298.257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                },
                "temporal": {
                    "interval": [
                        "2020-06-30T03:00:00Z/2020-07-04T21:00:00Z"
                    ],
                    "trs": "TIMECRS[\"DateTime\",TDATUM[\"Gregorian Calendar
\"],CS[TemporalDateTime,1],AXIS[\"Time (T)\",future]"
                }
            },
            "data_queries": {
                "position": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/position?coords=
{coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": true,
                        "variables": {
                            "title": "Position query",
                            "description": "Position query",
                            "query_type": "position",
                            "coords": {
                                "description": "Well Known Text POINT value i.
e. POINT(-120, 55)"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON"
                            ],
                            "default_output_format": "GeoJSON",
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
                "radius": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/radius?coords={coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": true,
                        "variables": {
                            "title": "Radius query",
                            "description": "Radius query",
                            "query_type": "radius",
                            "coords": {
```

```
                                "description": "Well Known Text POINT value i.
e. POINT(-120, 55)"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON",
                                "CSV"
                            ],
                            "default_output_format": "GeoJSON",
                            "within_units": [
                                "km",
                                "miles"
                            ],
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
                "area": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/area?coords={coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "title": "",
                        "templated": true,
                        "variables": {
                            "title": "Area query",
                            "description": "Area query",
                            "query_type": "area",
                            "coords": {
                                "description": "Well Known Text POLYGON value
 i.e. POLYGON((-79 40,-79 38,-75 38,-75 41,-79 40))"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON",
                                "CSV"
                            ],
                            "default_output_format": "CoverageJSON",
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
```

```
                "locations": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/locations",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": false,
                        "variables": {
                            "title": "Locations query",
                            "description": "Locations query",
                            "query_type": "location",
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON"
                            ],
                            "default_output_format": "GeoJSON",
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                }
            },
            "crs": [
                "CRS84"
            ],
            "output_formats": [
                "GeoJSON",
                "CoverageJSON",
                "CSV"
            ],
            "parameter_names": {
                "Wind Direction": {
                    "type": "Parameter",
                    "description": "Direction wind is from",
                    "unit": {
                        "label": "degree true",
                        "symbol": {
                            "value": "°",
                            "type": "http://http://www.example.org/edr/
metadata/units/degree"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-0",
                        "label": "Wind Direction"
                    },
                    "measurementType": {
                        "method": "mean",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Wind Speed": {
                    "type": "Parameter",
                    "description": "Average wind speed",
```

```
                    "unit": {
                        "label": "mph",
                        "symbol": {
                            "value": "mph",
                            "type": "http://http://www.example.org/edr/
metadata/units/mph"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-1",
                        "label": "Wind Speed"
                    },
                    "measurementType": {
                        "method": "mean",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Wind Gust": {
                    "type": "Parameter",
                    "description": "Wind gusts are a rapid increase in strength
 of the wind relative to the wind speed.",
                    "unit": {
                        "label": "mph",
                        "symbol": {
                            "value": "mph",
                            "type": "http://http://www.example.org/edr/
metadata/units/mph"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-1",
                        "label": "Wind Gust"
                    },
                    "measurementType": {
                        "method": "maximum",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Air Temperature": {
                    "type": "Parameter",
                    "description": "2m air temperature in the shade and out of
 the wind",
                    "unit": {
                        "label": "degC",
                        "symbol": {
                            "value": "°C",
                            "type": "http://http://www.example.org/edr/
metadata/units/degC"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
airTemperature",
                        "label": "Air Temperature"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Weather": {
                    "type": "Parameter",
                    "description": "",
```

```
                    "unit": {
                        "label": "weather",
                        "symbol": {
                            "value": "",
                            "type": "http://http://www.example.org/edr/
metadata/lookup/mo_dp_weather"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/wmdr/
ObservedVariableAtmosphere/_266",
                        "label": "Weather"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Relative Humidity": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "percent",
                        "symbol": {
                            "value": "%",
                            "type": "http://http://www.example.org/edr/
metadata/units/percent"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-1-1",
                        "label": "Relative Humidity"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Feels like temperature": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "degC",
                        "symbol": {
                            "value": "°C",
                            "type": "http://http://www.example.org/edr/
metadata/units/degC"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
airTemperature",
                        "label": "Feels like temperature"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "UV index": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
```

```
                        "label": "UV_index",
                        "symbol": {
                            "value": "",
                            "type": "http://http://www.example.org/edr/
metadata/lookup/mo_dp_uv"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-4-
51",
                        "label": "UV index"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Probabilty of precipitation": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "percent",
                        "symbol": {
                            "value": "%",
                            "type": "http://http://www.example.org/edr/
metadata/units/percent"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-1-1",
                        "label": "Probabilty of precipitation"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Visibility": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "quality",
                        "symbol": {
                            "value": "",
                            "type": "http://http://www.example.org/edr/
metadata/lookup/mo_dp_visibility"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
horizontalVisibility",
                        "label": "Visibility"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                }
            }
        },
        {
            "id": "2020-06-30T07:00:00Z",
            "title": "3 hrly fcst",
```

```
            "description": "Five day site specific forecast for 6000 UK
 locations 3 hrly fcst",
            "keywords": [
                "Wind Direction",
                "Wind Speed",
                "Wind Gust",
                "Air Temperature",
                "Weather",
                "Relative Humidity",
                "Feels like temperature",
                "UV index",
                "Probabilty of precipitation",
                "Visibility"
            ],
            "links": [
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/position",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/radius",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/area",
                    "hreflang": "en",
                    "rel": "data"
                },
                {
                    "href": "http://http://www.example.org/edr/collections/3_
hrly_fcst/instances/2020-06-30T10:00:00Z/locations",
                    "hreflang": "en",
                    "rel": "data"
                }
            ],
            "extent": {
                "spatial": {
                    "bbox": [
                        -15.0,
                        48.0,
                        5.0,
                        62.0
                    ],
                    "crs": "GEOGCS[\"WGS 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS
 84\",6378137,298.257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                },
                "temporal": {
                    "interval": [
                        "2020-06-30T03:00:00Z/2020-07-04T21:00:00Z"
                    ],
                    "trs": "TIMECRS[\"DateTime\",TDATUM[\"Gregorian Calendar
\"],CS[TemporalDateTime,1],AXIS[\"Time (T)\",future]"
                }
            },
            "data_queries": {
```

```
                "position": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/position?coords=
{coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": true,
                        "variables": {
                            "title": "Position query",
                            "description": "Position query",
                            "query_type": "position",
                            "coords": {
                                "description": "Well Known Text POINT value i.
e. POINT(-120, 55)"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON"
                            ],
                            "default_output_format": "GeoJSON",
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
                "radius": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/radius?coords={coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": true,
                        "variables": {
                            "title": "Radius query",
                            "description": "Radius query",
                            "query_type": "radius",
                            "coords": {
                                "description": "Well Known Text POINT value i.
e. POINT(-120, 55)"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON",
                                "CSV"
                            ],
                            "default_output_format": "GeoJSON",
                            "within_units": [
                                "km",
                                "miles"
                            ],
                            "crs_details": [
                                {
                                    "crs": "CRS84",
```

```
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
                "area": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/area?coords={coords}",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": true,
                        "variables": {
                            "title": "Area query",
                            "description": "Area query",
                            "query_type": "area",
                            "coords": {
                                "description": "Well Known Text POLYGON value
 i.e. POLYGON((-79 40,-79 38,-75 38,-75 41,-79 40))"
                            },
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON",
                                "CSV"
                            ],
                            "default_output_format": "CoverageJSON",
                            "crs_details": [
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                },
                "locations": {
                    "link": {
                        "href": "http://http://www.example.org/edr/
collections/3_hrly_fcst/instances/2020-06-30T10:00:00Z/locations",
                        "hreflang": "en",
                        "rel": "data",
                        "templated": false,
                        "variables": {
                            "title": "Locations query",
                            "description": "Locations query",
                            "query_type": "location",
                            "output_formats": [
                                "CoverageJSON",
                                "GeoJSON"
                            ],
                            "default_output_format": "GeoJSON",
                            "crs_details": [
```

```
                                {
                                    "crs": "CRS84",
                                    "wkt": "GEOGCS[\"WGS
 84\",DATUM[\"WGS_1984\",SPHEROID[\"WGS 84\",6378137,298.
257223563,AUTHORITY[\"EPSG\",\"7030\"]],AUTHORITY[\"EPSG\",
\"6326\"]],PRIMEM[\"Greenwich\",0,AUTHORITY[\"EPSG\",\"8901\"]],UNIT[\"degree
\",0.01745329251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",
\"4326\"]]"
                                }
                            ]
                        }
                    }
                }
            },
            "crs": [
                "CRS84"
            ],
            "output_formats": [
                "GeoJSON",
                "CoverageJSON",
                "CSV"
            ],
            "parameter_names": {
                "Wind Direction": {
                    "type": "Parameter",
                    "description": "Direction wind is from",
                    "unit": {
                        "label": "degree true",
                        "symbol": {
                            "value": "°",
                            "type": "http://http://www.example.org/edr/
metadata/units/degree"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-0",
                        "label": "Wind Direction"
                    },
                    "measurementType": {
                        "method": "mean",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Wind Speed": {
                    "type": "Parameter",
                    "description": "Average wind speed",
                    "unit": {
                        "label": "mph",
                        "symbol": {
                            "value": "mph",
                            "type": "http://http://www.example.org/edr/
metadata/units/mph"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-1",
                        "label": "Wind Speed"
                    },
                    "measurementType": {
                        "method": "mean",
                        "period": "-PT10M/PT0M"
                    }
                },
```

```
                "Wind Gust": {
                    "type": "Parameter",
                    "description": "Wind gusts are a rapid increase in strength
 of the wind relative to the wind speed.",
                    "unit": {
                        "label": "mph",
                        "symbol": {
                            "value": "mph",
                            "type": "http://http://www.example.org/edr/
metadata/units/mph"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-2-1",
                        "label": "Wind Gust"
                    },
                    "measurementType": {
                        "method": "maximum",
                        "period": "-PT10M/PT0M"
                    }
                },
                "Air Temperature": {
                    "type": "Parameter",
                    "description": "2m air temperature in the shade and out of
 the wind",
                    "unit": {
                        "label": "degC",
                        "symbol": {
                            "value": "°C",
                            "type": "http://http://www.example.org/edr/
metadata/units/degC"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
airTemperature",
                        "label": "Air Temperature"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Weather": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "weather",
                        "symbol": {
                            "value": "",
                            "type": "http://http://www.example.org/edr/
metadata/lookup/mo_dp_weather"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/wmdr/
ObservedVariableAtmosphere/_266",
                        "label": "Weather"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
```

```
                },
                "Relative Humidity": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "percent",
                        "symbol": {
                            "value": "%",
                            "type": "http://http://www.example.org/edr/
metadata/units/percent"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-1-1",
                        "label": "Relative Humidity"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Feels like temperature": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "degC",
                        "symbol": {
                            "value": "°C",
                            "type": "http://http://www.example.org/edr/
metadata/units/degC"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
airTemperature",
                        "label": "Feels like temperature"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "UV index": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "UV_index",
                        "symbol": {
                            "value": "",
                            "type": "http://http://www.example.org/edr/
metadata/lookup/mo_dp_uv"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-4-
51",
                        "label": "UV index"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
```

```
                "Probabilty of precipitation": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "percent",
                        "symbol": {
                            "value": "%",
                            "type": "http://http://www.example.org/edr/
metadata/units/percent"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/grib2/codeflag/4.2/_0-1-1",
                        "label": "Probabilty of precipitation"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                },
                "Visibility": {
                    "type": "Parameter",
                    "description": "",
                    "unit": {
                        "label": "quality",
                        "symbol": {
                            "value": "",
                            "type": "http://http://www.example.org/edr/
metadata/lookup/mo_dp_visibility"
                        }
                    },
                    "observedProperty": {
                        "id": "http://codes.wmo.int/common/quantity-kind/_
horizontalVisibility",
                        "label": "Visibility"
                    },
                    "measurementType": {
                        "method": "instantaneous",
                        "period": "PT0M"
                    }
                }
            }
        }
    ]
}
```


D.6. Location Query Metadata Examples

Example — Collection instance metadata response document: An example of the Locations metadata from a collection that supports the `Location` query pattern.

(link:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type]:

"items").

There is a link to the collections response itself (link relation type: “self”). Representations of this resource in other formats are referenced using link relation type “alternate”. Finally there are also links to the license information for the data (link relation type “license”).


```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "id": 3002,
      "geometry": {
        "type": "Point",
        "coordinates": [
          -0.854,
          60.749
        ]
      },
      "properties": {
        "name": "BALTASOUND",
        "datetime": "2020-03-30T19:00:00Z/2020-04-20T07:00:00Z",
        "detail": "https://oscar.wmo.int/surface/rest/api/stations/
station/1745/stationReport",
        "WIGOS Station Identifier": "0-20000-0-03002"
      }
    },
    {
      "type": "Feature",
      "id": 3005,
      "geometry": {
        "type": "Point",
        "coordinates": [
          -1.183,
          60.139
        ]
      },
      "properties": {
        "name": "LERWICK (S. SCREEN)",
        "datetime": "2020-03-30T19:00:00Z/2020-04-20T07:00:00Z",
        "detail": "https://oscar.wmo.int/surface/rest/api/stations/
station/1746/stationReport",
        "WIGOS Station Identifier": "0-20000-0-03005"
      }
    },
    {
      "type": "Feature",
      "id": 3008,
      "geometry": {
        "type": "Point",
        "coordinates": [
          -1.628,
          59.527
        ]
      },
      "properties": {
        "name": "FAIR ISLE",
        "datetime": "2020-03-30T19:00:00Z/2020-04-20T07:00:00Z",
        "detail": "https://oscar.wmo.int/surface/rest/api/stations/
station/1747/stationReport",
        "WIGOS Station Identifier": "0-20000-0-03008"
      }
    },
    {
      "type": "Feature",
```

```
      "id": 3017,
      "geometry": {
        "type": "Point",
        "coordinates": [
          -2.9,
          58.954
        ]
      },
      "properties": {
        "name": "KIRKWALL",
        "datetime": "2020-03-30T19:00:00Z/2020-04-20T07:00:00Z",
        "detail": "https://oscar.wmo.int/surface/rest/api/stations/
station/1750/stationReport",
        "WIGOS Station Identifier": "0-20000-0-03017"
      }
    },
    {
      "type": "Feature",
      "id": 3023,
      "geometry": {
        "type": "Point",
        "coordinates": [
          -7.397,
          57.358
        ]
      },
      "properties": {
        "name": "SOUTH UIST RANGE",
        "datetime": "2020-03-30T19:00:00Z/2020-04-20T07:00:00Z",
        "detail": "https://oscar.wmo.int/surface/rest/api/stations/
station/1751/stationReport",
        "WIGOS Station Identifier": "0-20000-0-03023"
      }
    }
  ]
```

E

ANNEX E (INFORMATIVE) RELATIONSHIP WITH OTHER OGC STANDARDS

E

ANNEX E (INFORMATIVE) RELATIONSHIP WITH OTHER OGC STANDARDS

E.1. Introduction

This Annex outlines the relationships, in terms of underlying conceptual models, overlaps, gaps, and target use cases and technologies, with other OGC standards.

E.2. Relationship between OGC API-EDR and OGC API-Features

The EDR API is completely compatible with OGC API — Features — Part 1: Core (OGC 17-069r3), in that it supports Collections and Items. It extends the Collection functionality by allowing 'Instances', a form of 'collection of collections'. The EDR API also supports the retrieval of spatiotemporal data by named location as well as coordinates.

E.3. Relationships between OGC API-EDR and Moving Features standards

There are four OGC Moving Features standards: conceptual model with XML encoding (OGC 18-075), access (OGC 16-120r3), CSV encoding (OGC 14-084r2), and JSON encoding (OGC 19-045r3). The Moving Features Standards are concerned with things that move along a trajectory, and simultaneously change their orientation through rigid body rotation. The concepts are defined in Unified Modeling Language (UML) and encoded in GML. The EDR API does not have the concept of orientation, or foliation or prisms. EDR API is OpenAPI defined, over HTTP(S), and not defined in UML.

Moving Features and EDR API do share a common conceptual definition, from ISO, of a Trajectory, but the Moving Features Standards encode trajectories in GML, CSV and Moving Features JSON, whereas the EDR API encodes trajectories in WKT. The Moving Features Standards support relationships between trajectories and other features, including other trajectories, the EDR API does not. Moving Features also explicitly supports concepts such as velocity, acceleration and distance along a trajectory, whereas the EDR API does not.

The Moving Features Standards consider trajectories as a primary resource to be queried, manipulated and processed, whereas in the EDR API, a trajectory is simply a query sampling pattern, encoded in WKT and ISO 8601 Date Time Format, into a spatiotemporal data resource.

E.4. Relationships between OGC API-EDR and Web Coverage Service and Coverage Implementation Schema

The primary messaging mechanism of the EDR API is JSON, including CoverageJSON, over HTTP(S). Implementations of the EDR API are described using the OpenAPI V3.0 specification. The target users are web-developers and end-users who are not geospatial experts. The target data resources are any dataset described as spatiotemporal, accessible by coordinates.

The EDR API is consistent with the Web Coverage Service (WCS) and Coverage Implementation Schema (CIS) standards but does not require the end user or developer to use the terms Domain and RangeSet. The EDR API can also be used to generate a single query against a collection of coverages, providing the data coordinate reference systems are consistent. The EDR API can support any of the WCS and CIS output formats if required. At the time of publication of version 1.0.0 of the EDR API, at least one EDR API implementation had been created by building on top of a WCS/CIS implementation.

The EDR API, with only a single form of spatiotemporal query, allows the retrieval of data from other data stores adhering to data models that are not coverages, such as features or observations.

E.5. Relationship between OGC API-EDR and the OGC MetOcean Application profile of Web Coverage Service (WCS) 2.1

The OGC API-EDR has developed out of the experiences of creating Part 0, Part 1 and Part 2 of the WCS 2.1 Met Ocean Application Profile, **ostensibly** for similar use cases, but for differing technology bases.

The primary messaging mechanism of the EDR API is JSON, including CoverageJSON, over HTTP(S). Implementations of the EDR API are described using the OpenAPI V3.0 specification.

The target users are web-developers and end-users who are not geospatial experts. The target data resources are any data described as spatiotemporal, accessible by coordinates, not just meteorological or oceanographic.

In contrast, the Met Ocean Application Profile of WCS 2.1 is designed primarily to support XML-encoded messaging, in particular, for GetCapabilities and GetCoverage requests. Responses returning coverages are modelled according to the CIS, which can be XML, JSON or JSON-LD. Developers and end-users are expected to be familiar with the geospatial terminology of coverages, and use the Profile with predominantly meteorological or oceanographic data.

The EDR API and the Met Ocean WCS Profile therefore support different use cases. Developers that are interested in extending their OWS or WCS solutions to support the Met Ocean domain are advised to use the Met Ocean Application Profile of WCS. Developers that are implementing Web APIs that make use of the OpenAPI specification are advised to use the EDR API.

E.6. Relationships between OGC API-EDR, SOS and SensorThings API

Both the OGC Sensor Observation Service (SOS) and the OGC SensorThings API enable access to observations made by sensors and transmitted over networks. As stated in Part 1 of the SensorThings API Standard “The main difference between the SensorThings API and the OGC SOS and Sensor Planning Service (SPS) is that the SensorThings API is designed specifically for the resource-constrained IoT devices and the Web developer community” (OGC 15-078r6).

Therefore, although the SensorThings API had overlaps with SOS within Web use cases, the OGC Membership acknowledged that there were some use cases within the IoT that could not be efficiently nor effectively addressed by the SOS. The same is true for the relationship between the EDR API and the SensorThings API.

SensorThings API follows OData’s specification for requesting entities. That means the entity control information, resource path usages, query options, the relevant JSON encodings, and batch-processing request follow OData 4.0. In contrast, the EDR API makes use of the OpenAPI V3.0 specification for describing resource paths, query options, JSON schema, and other aspects.

Further, the EDR API allows for retrieval of coverage data and HTML responses – both of which are not supported by the SensorThings API. Therefore, developers that are interested in IoT devices and OData, and do not have a need for HTML previews of content are advised to make use of the SensorThings API instead. Similarly developers that are interested in XML-encoded observations and sensor model descriptions are advised to make use of the SOS.

Similarly, an EDR or SensorThings API interface could be deployed on the same data set, so that users and developers that do not need the full details of observational, feature or coverage conceptual models and associated metadata could use the EDR API to hide the extra complexity, while users that do need all the details can use the SensorThings API to retrieve those.

F

ANNEX F (INFORMATIVE) GLOSSARY

F

ANNEX F (INFORMATIVE) GLOSSARY

F.1. Abstract Test Suite (ATS)

A compendium of test assertions applicable to implementations of a standard. An ATS provides a basis for developing an Executable Test Suite to verify that the implementation under test conforms to all the relevant functional specifications.

F.2. Collection

body of resources that belong or are used together. An aggregate, set, or group of related resources. (OGC 20-024)

F.3. Conformance Module; Conformance Test Module

set of related tests, all within a single conformance test class (OGC 08-131r3)

Note 1 to entry: When no ambiguity is possible, the word `test` may be omitted. i.e. conformance test module is the same as conformance module. Conformance modules may be nested in a hierarchical way.

F.4. Conformance Class; Conformance Test Class

set of conformance test modules that must be applied to receive a single certificate of conformance (OGC 08-131r3)

Note 1 to entry: When no ambiguity is possible, the word *test* may be left out, so conformance test class maybe called a conformance class.

F.5. dataset

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

F.6. distribution

represents an accessible form of a **dataset** (DCAT)

Note 1 to entry: EXAMPLE: a downloadable file, an RSS feed or a web service that provides the data.

F.7. Executable Test Suite (ETS)

A set of code (e.g. Java and Compliance Test Language) that provides runtime tests for the assertions defined by the ATS. Test data required to do the tests are part of the ETS (OGC 08-134)

F.8. Recommendation

expression in the content of a document conveying that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required, or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited (OGC 08-131r3)

F.9. **Requirement**

expression in the content of a document conveying criteria to be fulfilled if compliance with the document is to be claimed and from which no deviation is permitted (OGC 08-131r3)

F.10. **Requirements Class**

aggregate of all requirement modules that must all be satisfied to satisfy a conformance test class (OGC 08-131r3)

F.11. **Requirements Module**

aggregate of requirements and recommendations of a specification against a single standardization target type (OGC 08-131r3)

F.12. **Spatial Resource**

resources usually considered as Geospatial Data. (OGC 19-072)

F.13. **Standardization Target**

entity to which some requirements of a standard apply (OGC 08-131r3)

Note 1 to entry: The standardization target is the entity which may receive a certificate of conformance for a requirements class.

F.14. **Web API**

API using an architectural style that is founded on the technologies of the Web. (W3C Data on the Web Best Practices)

G

ANNEX G (INFORMATIVE) REVISION HISTORY

G

ANNEX G (INFORMATIVE) REVISION HISTORY

Table G.1 — Revision History

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2019-10-31	October 2019 snapshot	C. Heazel	all	Baseline update
2020-06-04	June 2020 master	Mark Burgoyne	all	Resolved Trajectory pattern
2020-06-05	June 2020 branch	Chris Little	all	Increase alignment with Common
2020-07-15	July 2020 branch	Chris Little	all	Editorial consistency
2020-07-22	Restructure branch	Chris Little	all	Restructure 7,8,9,10
2020-07-22	Issues 106, 107	Dave Blodgett	all	Fix broken links
2020-10-20	Oct 2020 Master	Chris Little	Definitions	Added missing Cube definition
2021-01-14	Issue 170	Tom Kralidis	all	normalize query parameters as kebab-case
2021-01-20	master branch	Tom Kralidis	all	Editorial updates, requirement update for z parameter
2021-02-18	Master branch	Mark Burgoyne	all	Add dedicated width and height query parameters to the corridor query
2021-03-03	Master branch	Mark Burgoyne	all	Simplify Cube query
2021-03-03	Master branch	Chris Little	all	replace references to Environmental with spatio-temporal
2021-03-26	Master branch	Chris Little	Title Page	Capture r3 snapshot of document prior to Planning Committee Vote

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2021-04-20	Master branch	Mark Burgoyne	all	Finalize collection response metadata
2021-04-27	Master branch	Chris Little and others	all	Improved realism of examples, editorial niceties
2021-05-10	Master branch	Mark Burgoyne	all	Fix broken links
2021-09-09	1.0	G.Hobona	all	Conversion to metanorma asciidoc

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Open Geospatial Consortium: The Specification Model — A Standard for Modular specifications, OGC 08-131

2. W3C/OGC: Spatial Data on the Web Best Practices, W3C Working Group Note 28 September 2017, https://www.w3.org/TR/sdw-bp/

3. W3C: Data on the Web Best Practices, W3C Recommendation 31 January 2017, https://www.w3.org/TR/dwbp/

4. W3C: Data Catalog Vocabulary, W3C Recommendation 16 January 2014, https://www.w3.org/TR/vocab-dcat/

5. IANA: Link Relation Types, https://www.iana.org/assignments/link-relations/link-relations.xml