

OGC® DOCUMENT: 19-086R6

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
Implementation

DRAFT

Version: 1.1

Submission Date: 2022-12-01

Approval Date: YYYY-MM-DD

Publication Date: YYYY-MM-DD

Editor: Mark Burgoyne, David Blodgett, Chuck Heazel, Chris Little

Notice for Drafts: This document is not an OGC Standard. This document is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

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 NON-INFRINGEMENT 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

Copyright © 2022 Open Geospatial Consortium
To obtain additional rights of use, visit <http://www.ogc.org/legal/>

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

I. ABSTRACT	xiv
II. KEYWORDS	xv
III. PREFACE	xvi
IV. SECURITY CONSIDERATIONS	xvii
V. SUBMITTING ORGANIZATIONS	xviii
VI. SUBMITTERS	xviii
1. SCOPE	2
2. CONFORMANCE	4
2.1. Mandatory Requirements Classes	4
2.2. Optional Requirements Classes	5
3. NORMATIVE REFERENCES	8
4. TERMS AND DEFINITIONS	11
5. CONVENTIONS	15
5.1. Identifiers	15
5.2. Link relations	15
5.3. Media Types	16
5.4. Examples	17
5.5. Schema	17
5.6. Use of HTTPS	18
5.7. API definition	18
6. OVERVIEW	21
6.1. General	21
6.2. Resource Paths	21
7. DEPENDENCIES ON CORE AND COLLECTIONS REQUIREMENTS CLASSES OF OGC API – COMMON	25
7.1. Overview	25
7.2. Platform	26
8. QUERY, SPATIOTEMPORAL AND INFORMATION RESOURCES	33

8.1. Information Resources	33
8.2. Query Resources	34
9. GENERAL REQUIREMENTS	56
9.1. HTTP 1.1	56
9.2. HTTP Status Codes	56
9.3. Web Caching	57
9.4. Support for Cross-Origin Requests	58
9.5. Asynchronous queries	58
9.6. Coordinate Reference Systems	58
9.7. Encodings	59
9.8. Link Headers	60
9.9. OpenAPI 3.0	60
9.10. Security	61
ANNEX A (INFORMATIVE) REQUIREMENTS DETAIL	64
A.1. Introduction	64
A.2. Requirements Class “Core” in Detail	64
A.3. Requirements Class “Collections” in Detail	67
A.4. Requirements Class “Queries” for Position, Area, Cube, Trajectory, Corridor, Items, Locations, and Instances	73
A.5. Requirements Class “Query parameters” in Detail	83
A.6. Requirements Class “JSON” in Detail	105
A.7. Requirements Class “GeoJSON” in Detail	106
A.8. Requirements Class “EDR GeoJSON” in Detail	107
A.9. Requirements Class “CoverageJSON” in Detail	109
A.10. Requirements Class “HTML” in Detail	110
A.11. Requirements Class “OpenAPI 3.0” in Detail	110
ANNEX B (INFORMATIVE) ABSTRACT TEST SUITE (NORMATIVE)	114
B.1. Introduction	114
B.2. Conformance Class Core	114
B.3. Conformance Class Collections	118
B.4. Conformance Class JSON	125
B.5. Conformance Class GeoJSON	126
B.6. Conformance Class EDR GeoJSON	128
B.7. Conformance Class CoverageJSON	129
B.8. Conformance Class HTML	130
B.9. Conformance Class OpenAPI 3.0	131
B.10. Conformance Class Queries	133
ANNEX C (INFORMATIVE) COLLECTION RESPONSE METADATA (INFORMATIVE)	176
C.1. EDR Collection Object Structure	176
ANNEX D (INFORMATIVE) EXAMPLES (INFORMATIVE)	190
D.1. Example Landing Pages	190

D.2. API Description Examples	190
D.3. Conformance Examples	190
D.4. Collections Metadata Examples	191
D.5. Instance Metadata Examples	205
D.6. Location Query Metadata Examples	233
ANNEX E (INFORMATIVE) RELATIONSHIP WITH OTHER OGC STANDARDS	237
E.1. Introduction	237
E.2. Relationship between OGC API-EDR and OGC API-Features	237
E.3. Relationships between OGC API-EDR and Moving Features standards	237
E.4. Relationships between OGC API-EDR and Web Coverage Service and Coverage Implementation Schema	238
E.5. Relationship between OGC API-EDR and the OGC MetOcean Application profile of Web Coverage Service (WCS) 2.1	238
E.6. Relationships between OGC API-EDR, SOS and SensorThings API	239
ANNEX F (INFORMATIVE) GLOSSARY	241
ANNEX G (INFORMATIVE) REVISION HISTORY	246
BIBLIOGRAPHY	249

LIST OF TABLES

Table 1 – Overview of Resources	xiv
Table 2 – Environmental Data Retrieval API Paths	22
Table 3 – Mapping OGC API – EDR Sections to OGC API – Common Requirements Classes	26
Table 4 – Information Resource Paths	34
Table 5 – Query Types	34
Table 6 – Position query structure	35
Table 7 – Radius query structure	36
Table 8 – Area query structure	38
Table 9 – Cube query structure	39
Table 10 – Trajectory query structure	41
Table 11 – Corridor query structure	43
Table 12 – Items query structure	48
Table 13 – locations query structure	49
Table 14 – Custom query parameter structure	53
Table 15 – Typical HTTP status codes	56
Table B.1 – Conformance Class “Core”	114
Table B.2 – Abstract Test 1	115

Table B.3 – Abstract Test 2	115
Table B.4 – Abstract Test 3	115
Table B.5 – Schema and Tests for Landing Pages	116
Table B.6 – Abstract Test 4	116
Table B.7 – Abstract Test 5	116
Table B.8 – Abstract Test 6	117
Table B.9 – Abstract Test 7	117
Table B.10 – Conformance Class “Collections”	118
Table B.11 – Abstract Test 8	118
Table B.12 – Abstract Test 9	118
Table B.13 – Abstract Test 10	119
Table B.14 – Schema and Tests for Collections content	119
Table B.15 – Abstract Test 11	119
Table B.16 – Abstract Test 12	120
Table B.17 – Abstract Test 13	120
Table B.18 – Abstract Test 14	121
Table B.19 – Schema and Tests for Collection Entries	121
Table B.20 – Abstract Test 15	121
Table B.21 – Abstract Test 16	122
Table B.22 – Abstract Test 17	122
Table B.23 – Abstract Test 18	123
Table B.24 – Abstract Test 19	123
Table B.25 – Abstract Test 20	123
Table B.26 – Abstract Test 21	124
Table B.27 – Abstract Test 22	124
Table B.28 – Abstract Test 23	124
Table B.29 – Abstract Test 24	125
Table B.30 – Conformance Class “JSON”	125
Table B.31 – Abstract Test 25	126
Table B.32 – Abstract Test 26	126
Table B.33 – Conformance Class “GeoJSON”	126
Table B.34 – Abstract Test 27	127
Table B.35 – Abstract Test 28	127
Table B.36 – Conformance Class “EDR GeoJSON”	128
Table B.37 – Abstract Test 29	128
Table B.38 – Abstract Test 30	128
Table B.39 – Conformance Class “CoverageJSON”	129
Table B.40 – Abstract Test 31	129
Table B.41 – Abstract Test 32	130
Table B.42 – Conformance Class “HTML”	130
Table B.43 – Abstract Test 33	130

Table B.44 – Abstract Test 34	131
Table B.45 – Conformance Class “OpenAPI 3.0”	131
Table B.46 – Abstract Test 35	131
Table B.47 – Abstract Test 36	132
Table B.48 – Abstract Test 37	132
Table B.49 – Abstract Test 38	132
Table B.50 – Abstract Test 39	132
Table B.51 – Abstract Test 40	133
Table B.52 – Conformance Class “Queries”	133
Table B.53 – Abstract Test 41	134
Table B.54 – Abstract Test 42	134
Table B.55 – Abstract Test 43	134
Table B.56 – Abstract Test 44	135
Table B.57 – Abstract Test 45	135
Table B.58 – Abstract Test 46	136
Table B.59 – Abstract Test 47	136
Table B.60 – Abstract Test 48	136
Table B.61 – Abstract Test 49	137
Table B.62 – Abstract Test 50	137
Table B.63 – Abstract Test 51	137
Table B.64 – Abstract Test 52	138
Table B.65 – Abstract Test 53	138
Table B.66 – Abstract Test 54	138
Table B.67 – Abstract Test 55	139
Table B.68 – Abstract Test 56	139
Table B.69 – Abstract Test 57	140
Table B.70 – Abstract Test 58	140
Table B.71 – Abstract Test 59	140
Table B.72 – Abstract Test 60	141
Table B.73 – Abstract Test 61	141
Table B.74 – Abstract Test 62	142
Table B.75 – Abstract Test 63	142
Table B.76 – Abstract Test 64	142
Table B.77 – Abstract Test 65	143
Table B.78 – Abstract Test 66	143
Table B.79 – Abstract Test 67	143
Table B.80 – Abstract Test 68	144
Table B.81 – Abstract Test 69	144
Table B.82 – Abstract Test 70	144
Table B.83 – Abstract Test 71	145
Table B.84 – Abstract Test 72	145

Table B.85 – Abstract Test 73	146
Table B.86 – Abstract Test 74	146
Table B.87 – Abstract Test 75	146
Table B.88 – Abstract Test 76	146
Table B.89 – Abstract Test 77	147
Table B.90 – Abstract Test 78	148
Table B.91 – Abstract Test 79	148
Table B.92 – Abstract Test 80	148
Table B.93 – Abstract Test 81	149
Table B.94 – Abstract Test 82	149
Table B.95 – Abstract Test 83	149
Table B.96 – Abstract Test 84	150
Table B.97 – Abstract Test 85	150
Table B.98 – Abstract Test 86	150
Table B.99 – Abstract Test 87	151
Table B.100 – Abstract Test 88	151
Table B.101 – Abstract Test 89	152
Table B.102 – Abstract Test 90	152
Table B.103 – Abstract Test 91	152
Table B.104 – Abstract Test 92	152
Table B.105 – Abstract Test 93	153
Table B.106 – Abstract Test 94	153
Table B.107 – Abstract Test 95	153
Table B.108 – Abstract Test 96	154
Table B.109 – Abstract Test 97	154
Table B.110 – Abstract Test 98	154
Table B.111 – Abstract Test 99	155
Table B.112 – Abstract Test 100	155
Table B.113 – Abstract Test 101	156
Table B.114 – Abstract Test 102	156
Table B.115 – Abstract Test 103	156
Table B.116 – Abstract Test 104	157
Table B.117 – Abstract Test 105	157
Table B.118 – Abstract Test 106	158
Table B.119 – Abstract Test 107	158
Table B.120 – Abstract Test 108	158
Table B.121 – Abstract Test 109	158
Table B.122 – Abstract Test 110	159
Table B.123 – Abstract Test 111	159
Table B.124 – Abstract Test 112	159
Table B.125 – Abstract Test 113	160

Table B.126 – Abstract Test 114	160
Table B.127 – Abstract Test 115	160
Table B.128 – Abstract Test 116	160
Table B.129 – Abstract Test 117	161
Table B.130 – Abstract Test 118	161
Table B.131 – Abstract Test 119	161
Table B.132 – Abstract Test 120	162
Table B.133 – Abstract Test 121	162
Table B.134 – Abstract Test 122	162
Table B.135 – Abstract Test 123	163
Table B.136 – Abstract Test 124	163
Table B.137 – Abstract Test 125	164
Table B.138 – Abstract Test 126	164
Table B.139 – Abstract Test 127	164
Table B.140 – Abstract Test 128	165
Table B.141 – Abstract Test 129	165
Table B.142 – Abstract Test 130	165
Table B.143 – Abstract Test 131	166
Table B.144 – Abstract Test 132	166
Table B.145 – Abstract Test 133	166
Table B.146 – Abstract Test 134	167
Table B.147 – Abstract Test 135	167
Table B.148 – Abstract Test 136	167
Table B.149 – Abstract Test 137	168
Table B.150 – Abstract Test 138	168
Table B.151 – Abstract Test 139	169
Table B.152 – Abstract Test 140	169
Table B.153 – Schema and Tests for Collections content	169
Table B.154 – Abstract Test 141	170
Table B.155 – Abstract Test 142	170
Table B.156 – Abstract Test 143	170
Table B.157 – Abstract Test 144	171
Table B.158 – Abstract Test 145	171
Table B.159 – Abstract Test 146	171
Table B.160 – Abstract Test 147	172
Table B.161 – Abstract Test 148	172
Table B.162 – Abstract Test 149	172
Table B.163 – Abstract Test 150	173
Table B.164 – Abstract Test 151	173
Table B.165 – Abstract Test 152	174
Table B.166 – Abstract Test 153	174

Table C.1 – EDR Collection Object Structure	176
Table C.2 – Link Object	177
Table C.3 – Variables Object	178
Table C.4 – CRS Details Object	179
Table C.5 – Extent Object	180
Table C.6 – Spatial Object	180
Table C.7 – Temporal Object	181
Table C.8 – Vertical Object	181
Table C.9 – Data Queries Object	183
Table C.10 – EDR Query Object	183
Table C.11 – Parameter Object	185
Table C.12 – Unit Object	185
Table C.13 – Symbol Object	186
Table C.14 – Observed Property Object	186
Table C.15 – Measurement Type object	187
Table G.1 – Revision History	246

LIST OF FIGURES

Figure 1 – Landing Page Response Schema	27
Figure 2 – Landing Page Example	28
Figure 3 – Conformance Response Schema	31
Figure 4 – Conformance Information Example	31
Figure A.1	71
Figure A.4	86
Figure A.5	87
Figure A.6	88
Figure C.1	179
Figure C.2	179
Figure C.3	181
Figure C.4	182
Figure C.5	183
Figure C.6	187

LIST OF RECOMMENDATIONS

RECOMMENDATION 1	48
------------------------	----

RECOMMENDATION 2	57
RECOMMENDATION 3	58
RECOMMENDATION 4	59
RECOMMENDATION 5	59
RECOMMENDATION 6	59
RECOMMENDATION 7	60
REQUIREMENTS CLASS: OGC API – ENVIRONMENTAL DATA RETRIEVAL CORE	
REQUIREMENTS CLASS	64
REQUIREMENT A.1	65
REQUIREMENT A.2	65
REQUIREMENT A.3	65
REQUIREMENT A.4	65
REQUIREMENT A.5	66
REQUIREMENT A.6	66
REQUIREMENT A.7	67
REQUIREMENT A.8	67
REQUIREMENTS CLASS: COLLECTIONS REQUIREMENTS CLASS	67
REQUIREMENT A.9	68
REQUIREMENT A.10	68
REQUIREMENT A.11	68
REQUIREMENT A.12	69
REQUIREMENT A.13	69
REQUIREMENT A.14	70
REQUIREMENT A.15	71
REQUIREMENT A.16	71
REQUIREMENT A.17	71
REQUIREMENT A.18	72
REQUIREMENTS CLASS: QUERIES REQUIREMENTS CLASS	73
REQUIREMENT A.19	74
REQUIREMENT A.20	75
REQUIREMENT A.21	76
REQUIREMENT A.22	77
REQUIREMENT A.23	79

REQUIREMENT A.24	81
REQUIREMENT A.25	81
REQUIREMENT A.26	82
REQUIREMENT A.27	82
REQUIREMENT A.28	82
REQUIREMENT A.29	83
REQUIREMENT A.30	83
REQUIREMENT A.31	84
REQUIREMENT A.32	85
REQUIREMENT A.33	86
REQUIREMENT A.34	87
REQUIREMENT A.35	87
REQUIREMENT A.36	88
REQUIREMENT A.37	88
REQUIREMENT A.38	89
REQUIREMENT A.39	90
REQUIREMENT A.40	90
REQUIREMENT A.41	90
REQUIREMENT A.42	91
REQUIREMENT A.43	91
REQUIREMENT A.44	92
REQUIREMENT A.45	92
REQUIREMENT A.46	93
REQUIREMENT A.47	93
REQUIREMENT A.48	94
REQUIREMENT A.49	94
REQUIREMENT A.50	94
REQUIREMENT A.51	95
REQUIREMENT A.52	96
REQUIREMENT A.53	97
REQUIREMENT A.54	97
REQUIREMENT A.55	98
REQUIREMENT A.56	98

REQUIREMENT A.57	100
REQUIREMENT A.58	101
REQUIREMENT A.59	101
REQUIREMENT A.60	102
REQUIREMENT A.61	102
REQUIREMENT A.62	102
REQUIREMENT A.63	103
REQUIREMENT A.64	103
REQUIREMENT A.65	104
REQUIREMENT A.66	104
REQUIREMENTS CLASS: JSON REQUIREMENTS CLASS	105
REQUIREMENT A.67	105
REQUIREMENT A.68	106
REQUIREMENTS CLASS: GEOJSON REQUIREMENTS CLASS	106
REQUIREMENT A.69	106
REQUIREMENT A.70	107
REQUIREMENTS CLASS: EDR GEOJSON REQUIREMENTS CLASS	107
REQUIREMENT A.71	108
REQUIREMENT A.72	108
REQUIREMENTS CLASS: COVERAGEJSON REQUIREMENTS CLASS	109
REQUIREMENT A.73	109
REQUIREMENT A.74	109
REQUIREMENTS CLASS: HTML REQUIREMENTS CLASS	110
REQUIREMENT A.75	110
REQUIREMENT A.76	110
REQUIREMENT A.77	111
REQUIREMENT A.78	111
REQUIREMENT A.79	112
REQUIREMENT A.80	112
REQUIREMENT A.81	112
REQUIREMENT A.82	112
REQUIREMENTS CLASS: OPENAPI 3.0 REQUIREMENTS CLASS	111



ABSTRACT

The OGC Environmental Data Retrieval (EDR) Application Programming Interface (API) provides a family of lightweight query interfaces to access spatiotemporal 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: 1. Make it easier to access a wide range of data through a uniform, well-defined simple Web interface; 2. 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 requested data can be unambiguously specified by spatio-temporal coordinates.

The EDR API query patterns – 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 data resource accessed by an EDR API instance is a multidimensional dataset that could be accessed via an implementation of the Web Coverage Service (WCS) standard. In contrast to SOS and WCS, the EDR API is fully consistent with the patterns 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 instead provide useful building blocks to enable 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 infrastructures.

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

RESOURCE	PATH	HTTP METHOD	DOCUMENT REFERENCE
API definition	/api	GET	API Definition
Conformance classes	/conformance	GET	Declaration of Conformance Classes
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 query via the API to determine supported 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.

The OGC API-EDR standard extends the common query operations listed in Table 1 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.



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



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.



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 Survey (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 (<i>editor</i>)	Met Office
Chris Little (<i>editor</i>)	Met Office
Charles Heazel (<i>editor</i>)	HeazelTech LLC
David Blodgett (<i>editor</i>)	USGS
Tom Kralidis (<i>contributor</i>)	Meteorological Service of Canada

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 ones, 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

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 Standard 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 are 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 following queries: Position, Radius, Area, Cube, Trajectory, Corridor, Items, or Locations SHALL be implemented.

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



3

NORMATIVE REFERENCES

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>

Fielding, R., Reschke, J.: IETF RFC 7231, Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, <https://datatracker.ietf.org/doc/html/rfc7231#section-4.3.3>

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 Lott: OGC 18-010r7, *Geographic information – Well-known text representation of coordinate reference systems*. Open Geospatial Consortium (2019). <https://docs.ogc.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). <https://docs.ogc.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

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 a vertical extent

4.2. corridor

two parameter set of points around a trajectory

(Note: generalised from the ISO definition of 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

attribute, such as 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. spatiotemporal 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

5.1. 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 `application/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 schemas are typically represented using YAML encoding, a human-readable data-serialization language. This convention is for the ease of the user. This Standard does not prohibit the use of another schema language or encoding. Nor does the Standard specify 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 Standard generally refers to the HTTP protocol. This is not meant to exclude the use of HTTPS and is simply 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

So that developers can more easily learn how to use the API, good documentation is essential for any 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 implementation APIs that share spatiotemporal 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 EDR 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

6.1. 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:

1. To make it easier to access a wide range of data through a uniform, well-defined simple Web interface;
2. To allow clients to retrieve a subset of data created by the API in response to a standardized, coordinate orientated, query pattern;
3. 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 spatiotemporal data resource of a relatively persistent data store. The query and its response are transient resources, but if required could be made persistent for re-use.

The functionality provided by the 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. The OGC API-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 behavior for the HTTP GET and POST operations. Future versions may introduce additional methods as required, consistent with RFC 7231. The HTTP methods supported by an EDR service MUST be defined by the service OpenAPI description.

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

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 OGC API – Environmental Data Retrieval Core Requirements Class 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:

1. API Platform: a set of common capabilities
2. Collection Access: operations for accessing collections of spatio-temporal data.
3. Query Resources: operations for accessing spatio-temporal data resources through queries
4. Parameters: parameters for use in OGC API-EDR operations.
5. 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 that support exploration of the resources offered via the API. The most important component of a landing page 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:

1. 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-desc
        type: application/vnd.oai.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/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",
    "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:

1. Issue a GET request on the {root}/api path
2. 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 OGC 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 EDR 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:

1. Issue a GET request on the `{root}/conformance` path
2. 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 EDR 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, SPATIOTEMPORAL AND INFORMATION RESOURCES

QUERY, SPATIOTEMPORAL AND INFORMATION RESOURCES

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

A spatiotemporal data resource is a collection of spatiotemporal 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 spatiotemporal data resources (collections of spatiotemporal 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 spatiotemporal 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 supports 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 identifiers associated with the {collectionId} collection.
/collections/{collectionId}/instances	Instances	List the available instances of the collection

8.2.1. 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 and is at the discretion of the query service implementer.** The filter constraints are defined by the following query parameters:

Table 6 – Position query structure

Query Parameter	Type	Required	Description	Examples
coords <ul style="list-style-type: none"> definition rules 	WKT string	Yes	The coordinates are defined by a Point Well Known Text (WKT) string	<ul style="list-style-type: none"> coords=POINT(-3 51) coords= MULTIPOINT((-77 38.9),(2.35 48.85),(116.38 39.92),(149.1 -35.29),(-0.1 51.5))
z <ul style="list-style-type: none"> definition rules 	String	No	The vertical level to return data for (available options defined in the vertical attribute of the extent section in the collections metadata response)	<ul style="list-style-type: none"> z=850 z=1000,900,850,700 z=2/100
datetime <ul style="list-style-type: none"> definition rules 	String	No	Datetime range to return data (the available range is defined in the temporal attribute of the extent section in the collections metadata response)	<ul style="list-style-type: none"> datetime=2018-02-12T00:00Z datetime=2018-02-12T00:00/2018-03-18T12:31Z datetime=2018-02-12T00:00Z,2018-02-12T01:00Z,2018-02-14T12:00Z
parameter-name <ul style="list-style-type: none"> definition rules 	String	No	Comma delimited list of parameter names available options listed in the parameter_names section of the collections metadata response	<ul style="list-style-type: none"> parameter-name=Visibility,Air%20Temperature`
crs <ul style="list-style-type: none"> definition rules 	String	No	Coordinate reference system identifier for the coords values and output data (available options listed in the collections metadata response)	<ul style="list-style-type: none"> crs=EPSG:4326 crs=A_CUSTOM_LABEL
f <ul style="list-style-type: none"> definition rules 	String	No	Data format for the output data	<ul style="list-style-type: none"> f=GeoJSON f=netCDF4 f=CoverageJSON

(available options listed in the collections response), schemas describing JSON and XML outputs can be defined in the Open API documentation (see <https://swagger.io/docs/specification/data-models/>)

- f=CSV

If a client request has a **coords** value which includes a height value and defines a **z** query parameter, the **z** query parameter will be the requested height value

8.2.2. 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:

Table 7 – Radius query structure

Query Parameter	Type	Required	Description	Examples
coords <ul style="list-style-type: none"> • definition • rules 	WKT string	Yes	The coordinates are defined by a Point Well Known Text (WKT) string	<ul style="list-style-type: none"> • coords=POINT(-3 51) • coords= MULTIPOINT((-77 38.9),(2.35 48.85), (116.38 39.92),(149.1 -35.29),(-0.1 51.5))
z <ul style="list-style-type: none"> • definition • rules 	String	No	The vertical level to return data for (available options defined in the vertical attribute of the extent section in the collections metadata response)	<ul style="list-style-type: none"> • z=850 • z=1000,900,850,700 • z=2/100
datetime <ul style="list-style-type: none"> • definition • rules 	String	No	Datetime range to return data (the available range is defined in the temporal attribute of the extent section in the collections metadata response)	<ul style="list-style-type: none"> • datetime=2018-02-12T00:00Z • datetime=2018-02-12T00:00/2018-03-18T12:31Z • datetime=2018-02-12T00:00Z,2018-02-12T01:00Z,2018-02-14T12:00Z
parameter-name <ul style="list-style-type: none"> • definition • rules 	String	No	Comma delimited list of parameter names available options listed in the parameter_names section of the	<ul style="list-style-type: none"> • parameter-name=Visibility,Air %20Temperature`

				collections metadata response	
within				Defines radius of area around defined coordinates to include in the data selection	<ul style="list-style-type: none"> • within=20
<ul style="list-style-type: none"> • definition • rules 	String	Yes			
within-units				Distance units for the within parameter (available options defined in the within_units attribute of the radius data_query section in the collections metadata response)	<ul style="list-style-type: none"> • within-units=Miles
<ul style="list-style-type: none"> • definition • rules 	String	Yes			
crs				Coordinate reference system identifier for the coords values and output data (available options listed in the collections metadata response)	<ul style="list-style-type: none"> • crs=EPSG:4326 • crs=A_CUSTOM_LABEL
<ul style="list-style-type: none"> • definition • rules 	String	No			
f				Data format for the output data (available options listed in the collections response), schemas describing JSON and XML outputs can be defined in the Open API documentation (see https://swagger.io/docs/specification/data-models/)	<ul style="list-style-type: none"> • f=GeoJSON • f=netCDF4 • f=CoverageJSON • f=CSV
<ul style="list-style-type: none"> • definition • rules 	String	No			

If a client request has a **coords** value which includes a height value and defines a **z** query parameter the **z** query parameter will be the requested height value

8.2.3. Area query

The Area query returns data within the polygon defined by the **coords** parameter. **Logic for identifying the best match for the requested area will depend on the collection and is at the discretion of the query service implementer.** 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:

Table 8 – Area query structure

Query Parameter	Type	Required	Description	Examples
coords <ul style="list-style-type: none"> definition rules 	WKT string	Yes	The coordinates are defined by a Polygon Well Known Text (WKT) string	<ul style="list-style-type: none"> 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)) 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)
z <ul style="list-style-type: none"> definition rules 	String	No	The vertical level to return data for (available options defined in the vertical attribute of the extent section in the collections metadata response)	<ul style="list-style-type: none"> z=850 z=1000,900,850,700 z=2/100
datetime <ul style="list-style-type: none"> definition rules 	String	No	Datetime range to return data (the available range is defined in the temporal attribute of the extent section in the collections metadata response)	<ul style="list-style-type: none"> datetime=2018-02-12T00:00Z datetime=2018-02-12T00:00/2018-03-18T12:31Z datetime=2018-02-12T00:00Z,2018-02-12T01:00Z,2018-02-14T12:00Z
parameter-name <ul style="list-style-type: none"> definition rules 	String	No	Comma delimited list of parameter names available options listed in the parameter_names section of the collections metadata response	<ul style="list-style-type: none"> parameter-name=Visibility,Air%20Temperature`
crs <ul style="list-style-type: none"> definition rules 	String	No	Coordinate reference system identifier for the coords values and output data (available options listed in the collections metadata response)	<ul style="list-style-type: none"> crs=EPSG:4326 crs=A_CUSTOM_LABEL
f <ul style="list-style-type: none"> definition rules 	String	No	Data format for the output data (available options listed in the collections response), schemas describing JSON and XML outputs can be	<ul style="list-style-type: none"> f=GeoJSON f=netCDF4 f=CoverageJSON f=CSV

defined in the Open API documentation (see <https://swagger.io/docs/specification/data-models/>)

If a client request has a **coords** value which includes a height value and defines a **z** query parameter the **z** query parameter will be the requested height value

8.2.4. 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:

Table 9 – Cube query structure

Query Parameter	Type	Required	Description	Examples
bbox <ul style="list-style-type: none"> definition rules 	String	Yes	<p>The coordinates are defined by a BBOX string. Only data that has a geometry that intersects the area defined by the bbox are selected.</p> <ul style="list-style-type: none"> Lower left corner, coordinate axis 1 Lower left corner, coordinate axis 2 Upper right corner, coordinate axis 1 Upper right corner, coordinate axis 2 <p>bbox= minx,miny,maxx,maxy</p> <p>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 meters above</p>	<ul style="list-style-type: none"> bbox=-6.0,50.0,-4.35,52.0

				mean sea level, or below for negative values.	
z	<ul style="list-style-type: none"> definition rules 	String	No	The vertical level to return data for (available options defined in the vertical attribute of the extent section in the collections metadata response)	<ul style="list-style-type: none"> z=850 z=1000,900,850,700 z=2/100
datetime	<ul style="list-style-type: none"> definition rules 	String	No	Datetime range to return data (the available range is defined in the temporal attribute of the extent section in the collections metadata response)	<ul style="list-style-type: none"> datetime=2018-02-12T00:00Z datetime=2018-02-12T00:00/2018-03-18T12:31Z datetime=2018-02-12T00:00Z,2018-02-12T01:00Z,2018-02-14T12:00Z
parameter-name	<ul style="list-style-type: none"> definition rules 	String	No	Comma delimited list of parameter names available options listed in the parameter_names section of the collections metadata response	<ul style="list-style-type: none"> parameter-name=Visibility,Air%20Temperature`
crs	<ul style="list-style-type: none"> definition rules 	String	No	Coordinate reference system identifier for the coords values and output data (available options listed in the collections metadata response)	<ul style="list-style-type: none"> crs=EPSG:4326 crs=A_CUSTOM_LABEL
f	<ul style="list-style-type: none"> definition rules 	String	No	Data format for the output data (available options listed in the collections response), schemas describing JSON and XML outputs can be defined in the Open API documentation (see https://swagger.io/docs/specification/data-models/)	<ul style="list-style-type: none"> f=GeoJSON f=netCDF4 f=CoverageJSON f=CSV

If a client request has a **bbox** value which includes height values and defines a **z** query parameter the **z** query parameter will be the definition of the requested height value

8.2.5. Trajectory query

The Trajectory query returns data along the path defined by the coords parameter. **Logic for identifying the best matches for the requested trajectory will depend on the collection and is at the discretion of the query service implementer** . The results are further filtered by the constraints defined by the following query parameters:

Table 10 – Trajectory query structure

Query Parameter	Type	Required	Description	Examples
coords <ul style="list-style-type: none"> definition rules 	WKT string	Yes	<p>The coordinates are defined by one of the following Well Known Text (WKT) strings:</p> <ul style="list-style-type: none"> LINESTRING LINESTRINGZ LINESTRINGM LINESTRINGZM The 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 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 	<ul style="list-style-type: none"> A Simple 2D corridor 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 with multiple segments: coords=MULTILINESTRING(-3.53 50.72, -3.35 50.92), (-3.11 51.02, -2.85 51.42, -2.59 51.46) A 2D corridor with all waypoints at same time: 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, all waypoints at the same height : 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, all waypoints at the same time and height: 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 each waypoint at a different time: 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, each waypoint at a different time by at the same height: coords=LINESTRINGM(51.14 -2.98 1560507000, 51.36 -2.87

				<pre>1560507600, 51.03 -3.15 1560508200, 50.74 -3.48 1560508500, 50.9 -3.36 1560510240)&z=200</pre> <ul style="list-style-type: none"> • A 3D corridor, each waypoint at a different height: 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 each waypoint at a different height but the same 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)&datetime=2018-02-12T23:00:00Z • A 4D corridor, each waypoint at different heights and times: 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) 	
z	<ul style="list-style-type: none"> • definition • rules 	String	No	The vertical level to return data for (available options defined in the vertical attribute of the extent section in the collections metadata response)	<ul style="list-style-type: none"> • z=850 • z=1000,900,850,700 • z=2/100
datetime	<ul style="list-style-type: none"> • definition • rules 	String	No	Datetime range to return data (the available range is defined in the temporal attribute of the extent section in the collections metadata response)	<ul style="list-style-type: none"> • datetime=2018-02-12T00:00Z • datetime=2018-02-12T00:00/2018-03-18T12:31Z • datetime=2018-02-12T00:00Z,2018-02-12T01:00Z,2018-02-14T12:00Z
parameter-name	<ul style="list-style-type: none"> • definition • rules 	String	No	Comma delimited list of parameter names available options listed in the parameter_names section of the collections metadata response	<ul style="list-style-type: none"> • parameter-name=Visibility,Air%20Temperature
crs	<ul style="list-style-type: none"> • definition • rules 	String	No	Coordinate reference system identifier for the coords values and output data (available	<ul style="list-style-type: none"> • crs=EPSG:4326 • crs=A_CUSTOM_LABEL

			options listed in the collections metadata response
f	<ul style="list-style-type: none"> definition rules 	String	No Data format for the output data (available options listed in the collections response), schemas describing JSON and XML outputs can be defined in the Open API documentation (see https://swagger.io/docs/specification/data-models/) <ul style="list-style-type: none"> f=GeoJSON f=netCDF4 f=CoverageJSON f=CSV

If any of the following occurs:

- client request contains coords=LINestringZ(...) and a z query parameter
- client request contains coords=LINestringM(...) and a **datetime** query parameter
- client request contains coords=LINestringZM(...) and z or **datetime** query parameters

An error MUST be thrown by the server

8.2.6. Corridor query

The Corridor query returns data along and around the path defined by the coords parameter. **Logic for identifying the best match for the requested corridor will depend on the collection and is at the discretion of the query service implementer**. The results are further filtered by the constraints defined by the following query parameters:

Table 11 – Corridor query structure

Query Parameter	Type	Required	Description	Examples
coords <ul style="list-style-type: none"> definition rules 	WKT string	Yes	The coordinates are defined by one of the following Well Known Text (WKT) strings: <ul style="list-style-type: none"> LINestring LINestringZ LINestringM LINestringZM The Z in LINestringZ and 	<ul style="list-style-type: none"> 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 with multiple segments, on the surface of earth with no time or vertical dimensions: coords=MULTILINestring(-3.53 50.72, -3.35 50.92), (-3.

LINestringZM refers to the height value. If the specified CRS does not define the height units, the height units will default to metres above mean sea level

- 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

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.

				<p>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)</p> <ul style="list-style-type: none"> 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 0.4 1560513600,-2.59 51.46 0.5 1560515400) 	
z	<ul style="list-style-type: none"> definition rules 	String	No	<p>The vertical level to return data for (available options defined in the vertical attribute of the extent section in the collections metadata response)</p>	<ul style="list-style-type: none"> z=850 z=1000,900,850,700 z=2/100
datetime	<ul style="list-style-type: none"> definition rules 	String	No	<p>Datetime range to return data (the available range is defined in the temporal attribute of the extent section in the collections metadata response)</p>	<ul style="list-style-type: none"> datetime=2018-02-12T00:00Z datetime=2018-02-12T00:00/2018-03-18T12:31Z datetime=2018-02-12T00:00Z,2018-02-12T01:00Z,2018-02-14T12:00Z
parameter-name	<ul style="list-style-type: none"> definition rules 	String	No	<p>Comma delimited list of parameter names available options listed in the parameter_names section of the collections metadata response</p>	<ul style="list-style-type: none"> parameter-name=Visibility,Air%20Temperature
resolution-x	<ul style="list-style-type: none"> definition rules 	String	No	<p>Defined if the user requires data at a different resolution from the native resolution of the data along the x-axis, it denotes the</p>	<ul style="list-style-type: none"> resolution-x=10

				number of intervals to retrieve data for along the x-axis	
resolution-y				Defined if the user requires data at a different resolution from the native resolution of the data along the y-axis, it denotes the number of intervals to retrieve data for along the y-axis	
<ul style="list-style-type: none"> • definition • rules 	String	No			<ul style="list-style-type: none"> • resolution-y=5
resolution-z				Defined if the user requires data at a different resolution from the native resolution of the data along the z-axis, it denotes the number of intervals to retrieve data for along the z-axis	
<ul style="list-style-type: none"> • definition • rules 	String	No			<ul style="list-style-type: none"> • resolution-z=100
corridor-width				The width value represents the whole width of the corridor where the trajectory supplied in the coords query parameter is the center point of the corridor	
<ul style="list-style-type: none"> • definition • rules 	String	Yes			<ul style="list-style-type: none"> • corridor-width=100
width-units				Distance units for the corridor-width parameter (available options defined in the width_units attribute of the corridor data_query section in the collections metadata response)	
<ul style="list-style-type: none"> • definition • rules 	String	Yes			<ul style="list-style-type: none"> • width-units=KM
corridor-height				The height value represents the whole height of the corridor where the trajectory supplied in the coords query parameter is the centre point of the corridor	
<ul style="list-style-type: none"> • definition • rules 	String	Yes			<ul style="list-style-type: none"> • corridor-height=100
height-units				Distance units for the corridor-height	
<ul style="list-style-type: none"> • definition 	String	Yes			<ul style="list-style-type: none"> • height-units=hPa

				parameter (available options defined in the height_units attribute of the corridor data_query section in the collections metadata response)	
				coordinate reference system identifier for the coords values and output data (available options listed in the collections metadata response)	<ul style="list-style-type: none"> • crs=EPSG:4326 • crs=A_CUSTOM_LABEL
crs	<ul style="list-style-type: none"> • definition • rules 	String	No		
				Data format for the output data (available options listed in the collections response), schemas describing JSON and XML outputs can be defined in the Open API documentation (see https://swagger.io/docs/specification/data-models/)	<ul style="list-style-type: none"> • f=GeoJSON • f=netCDF4 • f=CoverageJSON • f=CSV
f	<ul style="list-style-type: none"> • definition • rules 	String	No		

If any of the following occurs:

- Client request contains coords=LINestringZ(...) and a **z** query parameter
- Client request contains coords=LINestringM(...) and a **datetime** query parameter
- Client request contains coords=LINestringZM(...) and **z** or **datetime** query parameters

An error MUST be thrown by the server

8.2.7. Items query

The EDR API 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 a particular query has been cached for later use, such as a monitoring location. Or there may be a catalog of spatiotemporal sampling features such as domains of anomalies in a dataset. A GeoJSON-compatible JSON-Schema is specified to document an EDR API query endpoint and valid query parameters including time range, parameters, and spatial characteristics.

RECOMMENDATION 1

A:

STATEMENT If a collection using other EDR queries uses the `items` query, implementations **SHOULD** consider support for the [EDR GeoJSON Schema](#) as an encoding.

8.2.7.1. 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.

The filter constraints are defined by the following query parameters:

Table 12 – `Items` query structure

Query Parameter	Type	Required	Description	Examples
bbox <ul style="list-style-type: none"> definition rules 	String	Yes	<p>The coordinates are defined by a BBOX string. Only data that has a geometry that intersects the area defined by the <code>bbox</code> are selected.</p> <ul style="list-style-type: none"> Lower left corner, coordinate axis 1 Lower left corner, coordinate axis 2 Upper right corner, coordinate axis 1 Upper right corner, coordinate axis 2 <p><code>bbox=</code> <code>minx,miny,maxx,maxy</code> The X and Y coordinates are values in the coordinate system defined by the <code>crs</code> query parameter. If <code>crs</code> is not defined, the values will be assumed to be WGS84 longitude/latitude coordinates and heights will be assumed</p>	<ul style="list-style-type: none"> <code>bbox=-6.0,50.0,-4.35,52.0</code>

			to be in meters above mean sea level.	
datetime			Datetime range to return data (the available range is defined in the temporal attribute of the extent section in the collections metadata response)	<ul style="list-style-type: none"> datetime=2018-02-12T00:00Z datetime=2018-02-12T00:00/2018-03-18T12:31Z datetime=2018-02-12T00:00Z,2018-02-12T01:00Z,2018-02-14T12:00Z
limit			Maximum number of items to return	<ul style="list-style-type: none"> limit=10
<ul style="list-style-type: none"> definition rules 	String	No		
<ul style="list-style-type: none"> definition rules 	String	No		

Example — itemId: To retrieve an item, specify the required Item identifier:

`/collections/{collectionId}/items/{itemId}`

The following example returns 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`

The following example returns 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`

8.2.8. Locations query

The Locations query returns data for the named location. **Logic for identifying the best match for the coordinate will depend on the collection and is at the discretion of the query service implementer.** If a location id is not defined the API **MUST** return a GeoJSON features array of valid location identifiers.

The filter constraints are defined by the following query parameters:

Table 13 — locations query structure

Query Parameter	Type	Required	Description	Examples
locationId	String	No	A unique identifier for the required location, such as a GeoHash , a World Meteorological Organization (WMO)	<ul style="list-style-type: none"> EGLL Ottawa limit.broom.flip gbsvn

				station identifier or place name.	
datetime				Datetime range to return data (the available range is defined in the temporal attribute of the extent section in the collections metadata response)	<ul style="list-style-type: none"> datetime=2018-02-12T00:00Z datetime=2018-02-12T00:00/2018-03-18T12:31Z datetime=2018-02-12T00:00Z,2018-02-12T01:00Z,2018-02-14T12:00Z
parameter-name				Comma delimited list of parameter names available options listed in the parameter_names section of the collections metadata response	<ul style="list-style-type: none"> parameter-name=Visibility,Air%20Temperature`
crs				Coordinate reference system identifier for the coords values and output data (available options listed in the collections metadata response)	<ul style="list-style-type: none"> crs=EPSG:4326 crs=A_CUSTOM_LABEL
f				Data format for the output data (available options listed in the collections response), schemas describing JSON and XML outputs can be defined in the Open API documentation (see https://swagger.io/docs/specification/data-models/)	<ul style="list-style-type: none"> f=GeoJSON f=netCDF4 f=CoverageJSON f=CSV

8.2.9. Instances query

Having multiple versions or instances of the same collection, where the same information is reprocessed or regenerated is not unusual. Although these versions could be described as new collections the instance query type allows this data to be described as different views of the same collection.

8.2.9.1. Parameter instancelId

A unique identifier for the instance of the collection is specified as:

```
/collections/{collectionId}/instance/{instancelId}
```

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

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

8.2.9.2. Parameter queryType

The queryType options are exactly the same as those available for 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. The queryType structure is:

```
/collections/{collectionId}/instance/{instancelId}/{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 (instancelId = raw) for the Metar ((collectionId = metar) collection: /collections/metar/instance/raw/position

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

8.2.10. Custom dimension support

Support for dimensions other than the standard spatiotemporal dimensions can be provided by adding the custom dimension type to the EDR API extent object in the collections response. The custom dimension type allows for an array of custom query dimensions to be defined. Each custom dimension item is defined with the following characteristics (using an OpenAPI Specification 3.0 fragment)

Example 1 – Schema for custom query parameter metadata

```
type: object
required:
  - id
  - interval
  - reference
properties:
  id:
    type: string
  interval:
    type: array
    minItems: 1
    items:
      type: array
```

```

        minItems: 2
        items:
          anyOf:
            - type: string
            - type: number
        nullable: true
values:
  type: array
  minItems: 1
  items:
    anyOf:
      - type: string
      - type: number
    nullable: true
reference:
  type: string

```

Following the conventions used for the temporal or vertical extents, custom extent objects must provide the following information:

- id: Name of the custom dimension.
- interval: data value range of the dimension (minimum value, maximum value).
- values: defines all of the supported data values.
- reference: string which describes the custom dimension (this can be a link to a detailed description).

Example 2 – custom dimension definitions in a collections response

```

"extent": {
  "spatial": {
    "bbox": [[-180,-90,180,90]],
    "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\", 0.0174532925199433, AUTHORITY[\"EPSG\", \"9122\"]],
      AUTHORITY[\"EPSG\", \"4326\"]]",
  },
  "temporal": {
    "interval": [\"2017-06-14T00:00:00Z\", \"2017-06-14T12:00:00Z\"],
    "values": [\"2017-06-14T00:00:00Z\", \"2017-06-14T06:00:00Z\",
      \"2017-06-14T12:00:00Z\", \"2017-06-14T18:00:00Z\"],
    "trs": "TIMECRS[\"DateTime\", TDATUM[\"Gregorian Calendar\"],
      CS[TemporalDateTime, 1],
      AXIS[\"Time (T)\", future]]"
  },
  "custom": [
    {
      "id": "realisations",
      "interval": [[1, 50]],
      "values": ["R50/1/1"],
      "reference": "Ensemble members"
    },
    {
      "id": "icao_ids",
      "interval": [\"EB\", \"EB\"],
      "values": [\"EBAW\", \"EBBR\", \"EBBR\", \"EBKT\", \"EBLG\", \"EBOS\"],
    }
  ]
}

```

```

    "reference": "https://en.wikipedia.org/wiki/ICAO_airport_code"
  }
]
}

```

The `id` field can then be used as the name of a query parameter in any of the `query_types` supported by the collection. The query parameter value definition will follow the same conventions as the `datetime` and `z` query parameters by supporting single value, value lists and value ranges as valid inputs.

Custom query parameter definition

Table 14 – Custom query parameter structure

Query Parameter	Type	Required	Description	Examples
<i>custom_dimension_name</i> ¹ <ul style="list-style-type: none"> definition rules 	String	No	The value range to return data for (available options defined in the values attribute of the custom dimension definition in the extent section in the collections metadata response)	<ul style="list-style-type: none"> realisations=5 realisations=1,9,15,25 realisations=9/15

¹ The names of the custom dimension query parameters are defined as the `id` value within the custom attribute of the extent object in the collection metadata response.

Example 3 – Return the 5th, 10th, 15th and 20th ensemble members between 2022-05-10T00:00Z and 2022-05-10T06:00Z at height level 2.0: https://server.example/global_model/area?coords=POLYGON-12.602 59.367,-12.954 46.580,2.169 47.061,2.169 60.769,-12.602 59.367

`¶meter-name=Temperature`

`&datetime=2022-05-10T00:00Z/2022-05-10T06:00Z`

`&realisations=R4/5/5`

`&z=2.0`

`&crs=EPSG:4326`

`&f=CoverageJSON`

Example 4 – Only return data for EBBR EBOS between 2022-05-10T00:00Z and 2022-05-10T06:00Z from the available sites in the European area.: <https://server.example/observations/locations/europe?>

¶meter-name=wind_speed,wind_direction

&datetime=2022-05-10T00:00Z/2022-05-10T06:00Z

&icao_ids=EBBR,EBOS

&crs=EPSG:4326

&f=CoverageJSON



9

GENERAL REQUIREMENTS

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 15 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 15 – 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 <code>Retry-After</code> 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 <code>Location</code> 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 <code>WWW-Authenticate</code> 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 authorized 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 15, 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	
STATEMENT	A: The service SHOULD support entity tags and the associated headers as specified by HTTP/1.1.

9.4. Support for Cross-Origin Requests

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

RECOMMENDATION 3

A:

STATEMENT 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

Responding to queries synchronously is not always be possible. The EDR API standard does not specify how to handle asynchronous requests. Different services may propose different best practices.

For example, if the query requires handling requests asynchronously, one option of many, 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 SHOULD 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

A:

STATEMENT 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

A:

STATEMENT 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

A:

STATEMENT 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 URLs for resources to express links, such as for alternative representations of

the same resource. This document does not mandate any 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

A:

STATEMENT 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:

STATEMENT 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 as well as common codes such as 200 for successful GET requests and 400, 404 or 500 for error situations. See Clause 9.2.

Clients must 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 Standard does not mandate any specific security controls. However, the Standard 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.



ANNEX A (INFORMATIVE) REQUIREMENTS DETAIL



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	
OBLIGATION	requirement
TARGET TYPE	Web API
PREREQUISITES	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core http://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections
NORMATIVE STATEMENTS	Requirement A.-1 Requirement A.-2 Requirement A.-3 Requirement A.-4 Requirement A.-5 Requirement A.-6
Example 1	
Example 2	

REQUIREMENT A.1

STATEMENT A:
The server SHALL support the HTTP GET operation at the path /.

REQUIREMENT A.2

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

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

REQUIREMENT A.3

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

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

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

REQUIREMENT A.4

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

REQUIREMENT A.4

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

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

STATEMENT Note:
The f parameter SHOULD be used to satisfy this requirement.

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

REQUIREMENT A.5

STATEMENT A:
The list of Conformance Classes advertised by the API SHALL include:
<http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/core>

STATEMENT B:
<http://www.opengis.net/spec/ogcapi-common-2/1.0/conf/collections>

STATEMENT C:
<http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core>

REQUIREMENT A.6

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

STATEMENT 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/http HTTP

REQUIREMENT A.7

STATEMENT A:
The API SHALL conform to HTTP 1.1.

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

A.2.4. Requirement /req/core/crs84 CRS84

REQUIREMENT A.8

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 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

OBLIGATION	requirement
TARGET TYPE	Web API
PREREQUISITES	http://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core ISO 19107 ISO 19108 ISO 19111

REQUIREMENTS CLASS: COLLECTIONS REQUIREMENTS CLASS

ISO 19108
ISO 8601

NORMATIVE STATEMENTS

Requirement A.-1
Requirement A.-2
Requirement A.-3
Requirement A.-4
Requirement A.-5
Requirement A.-6
Requirement A.-7
Requirement A.-8
Requirement A.-9
Requirement A.-10

REQUIREMENT A.9

STATEMENT A:
The API SHALL support the HTTP GET operation at the path /collections.

REQUIREMENT A.10

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

STATEMENT B:
The content of that response SHALL be based upon the schema [collections.yaml](#).

REQUIREMENT A.11

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

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

REQUIREMENT A.12

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

STATEMENT B:
The content of that response SHALL be based upon the schema [collection.yaml](#).

STATEMENT 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.13

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

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

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

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

STATEMENT E:
Every Collection within a collections array MUST have a links parameter which must comply with the requirement [/req/core/rc-collection-info-links](#).

STATEMENT F:
Every Collection within a collections array MUST have an extent parameter which must comply with the requirement [/req/core/rc-extent](#).

REQUIREMENT A.13

G:

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

H:

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

STATEMENT **I:**

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

J:

STATEMENT 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:

STATEMENT Every Collection within a collections array **MUST** have an `parameter_names` attribute containing a list of parameters must comply with the requirement `/req/edr/rc-parameters`.

REQUIREMENT A.14

A:

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

STATEMENT

- 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.

STATEMENT **B:**

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

REQUIREMENT A.15

A:
STATEMENT 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.

B:
STATEMENT 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.16

A:
STATEMENT 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``).

Figure A.1

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

REQUIREMENT A.17

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

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

C:
STATEMENT 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.18

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

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

STATEMENT 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.

STATEMENT 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.

STATEMENT 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.

STATEMENT 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.
STATEMENT F: 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 have 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.

STATEMENT 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.

REQUIREMENT A.18

H:
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

OBLIGATION	requirement
TARGET TYPE	Web API
PREREQUISITE	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/collections
NORMATIVE STATEMENTS	Requirement A.-1 Requirement A.-2 Requirement A.-3 Requirement A.-4 Requirement A.-5 Requirement A.-6 Requirement A.-7

REQUIREMENTS CLASS: QUERIES REQUIREMENTS CLASS

Requirement A.-8
Requirement A.-9
Requirement A.-10
Requirement A.-11

A.4.2. Requirements for Position Queries

REQUIREMENT A.19

A:

STATEMENT 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:

STATEMENT For every collection identified in the feature collections response (path /collections), the server MAY support the HTTP POST operation at the path /collections/{collectionId}/position.

C:

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

D:

STATEMENT A position GET or POST operation MUST include a coords query parameter

E:

STATEMENT If the coords query parameter is not specified a HTTP 400 error should be generated

F:

STATEMENT A position GET or POST operation MAY include a z query parameter

G:

STATEMENT A position GET or POST operation SHOULD include a parameter-name query parameter

REQUIREMENT A.19

STATEMENT H:
A position GET or POST operation MAY include a datetime query parameter

STATEMENT I:
A position GET or POST operation SHOULD include a crs query parameter

STATEMENT J:
A position GET or POST operation SHOULD include an f query parameter

A.4.3. Requirements for Area Queries

REQUIREMENT A.20

STATEMENT 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.

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

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

STATEMENT D:
An area GET operation MUST include a coords query parameter

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

STATEMENT F:
An area GET or POST operation MAY include a z query parameter

REQUIREMENT A.20

STATEMENT G:
An area GET or POST operation SHOULD include a parameter-name query parameter

STATEMENT H:
An area GET or POST operation MAY include a date-time query parameter

STATEMENT I:
An area GET or POST operation SHOULD include a crs query parameter

STATEMENT J:
An area GET or POST operation SHOULD include a f query parameter

A.4.4. Requirements for Cube Queries

REQUIREMENT A.21

STATEMENT 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.

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

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

STATEMENT D:
A cube GET or POST operation MUST include a bbox query parameter

REQUIREMENT A.21

E:

STATEMENT A bbox query parameter SHOULD only consist of four values, any vertical values defined by bbox will be overridden by the values assigned to the z query parameter

F:

STATEMENT If the bbox query parameter is not specified a HTTP 400 error should be generated

G:

STATEMENT A cube GET or POST operation MUST include a z query parameter

H:

STATEMENT If the z query parameter is not specified a HTTP 400 error should be generated

I:

STATEMENT A cube GET or POST operation SHOULD include a parameter-name query parameter

J:

STATEMENT A cube GET or POST operation MAY include a datetime query parameter

K:

STATEMENT A cube GET or POST operation SHOULD include a crs query parameter

L:

STATEMENT A cube GET or POST operation SHOULD include an f query parameter

A.4.5. Requirements for Trajectory Queries

REQUIREMENT A.22

A:

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

REQUIREMENT A.22

B:

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

C:

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

D:

STATEMENT A trajectory GET or POST operation MUST include a coords query parameter

E:

STATEMENT If the coords query parameter is not specified a HTTP 400 error should be generated

F:

STATEMENT A trajectory GET or POST operation MAY include a z query parameter

G:

STATEMENT A trajectory GET or POST operation SHOULD include a parameter-name query parameter

H:

STATEMENT A trajectory GET or POST operation MAY include a datetime query parameter

I:

STATEMENT A trajectory GET or POST operation SHOULD include a crs query parameter

J:

STATEMENT A trajectory GET or POST operation SHOULD include an f query parameter

A.4.6. Requirements for Corridor Queries

REQUIREMENT A.23

A:

STATEMENT 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:

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

C:

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

D:

STATEMENT A corridor GET operation MUST include a coords query parameter

E:

STATEMENT If the coords query parameter is not specified a HTTP 400 error should be generated

F:

STATEMENT A corridor GET operation MUST include a corridor-width query parameter

G:

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

H:

STATEMENT A corridor GET operation MUST include a corridor-height query parameter

I:

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

REQUIREMENT A.23

STATEMENT J:
A corridor GET operation MUST include a width-units query parameter

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

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

STATEMENT M:
A corridor GET operation MUST include a height-units query parameter

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

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

STATEMENT P:
A corridor GET or POST operation MAY include a z query parameter

STATEMENT Q:
A corridor GET or POST operation SHOULD include a parameter-name query parameter

STATEMENT R:
A corridor GET or POST operation MAY include a datetime query parameter

STATEMENT S:

REQUIREMENT A.23

A corridor GET or POST operation SHOULD include a crs query parameter

STATEMENT T:
A corridor GET or POST operation SHOULD include a f query parameter

A.4.7. Requirements for Items Queries

REQUIREMENT A.24

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

A.4.8. Requirements for Locations Queries

REQUIREMENT A.25

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

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

REQUIREMENT A.25

D:

STATEMENT If a `locationId` is not specified a list of valid `locationId`'s MUST be returned with a description of their geospatial extent.

E:

STATEMENT A `locations` GET or POST operation SHOULD include a `parameter-name` query parameter

F:

STATEMENT A `locations` GET or POST operation MAY include a `datetime` query parameter

G:

STATEMENT A `locations` GET or POST operation SHOULD include a `crs` query parameter

H:

STATEMENT A `locations` GET or POST operation SHOULD include an `f` query parameter

A.4.9. Requirements for Instances Queries

REQUIREMENT A.26

A:

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

REQUIREMENT A.27

A:

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

REQUIREMENT A.28

A:

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

REQUIREMENT A.28

B:

STATEMENT 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.29

A:

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

B:

STATEMENT The content of that response SHALL be based upon the JSON schema [instances.yaml](#).

C:

STATEMENT 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 “Query parameters” in Detail

A.5.1. Requirements Class: Query parameters

Unresolved directive in sections/annex_requirements.adoc – include:../requirements/requirements_query_parameters.adoc[]

A.5.2. Requirement `/req/core/rc-bbox-definition` Parameter `bbox` definition

REQUIREMENT A.30

A:

STATEMENT The operation SHALL support a parameter `bbox` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
`name: bbox`

REQUIREMENT A.30

```
in: query
required: false
schema:
  oneOf:
    - items:
        minItems: 4
        maxItems: 4
        type: number
      type: array
    - items:
        minItems: 6
        maxItems: 6
        type: number
      type: array
style: form
explode: false
```

A.5.3. Requirement /req/core/rc-bbox-response Parameter bbox response

REQUIREMENT A.31

A:

STATEMENT 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:

STATEMENT 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:

STATEMENT 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):

- STATEMENT**
- 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.31

E:
STATEMEN The bounding box SHALL consist of four or six numbers, depending on whether the coordinate reference system includes a vertical axis (height or depth) and the coordinate reference system of the values SHALL be interpreted as the coordinate reference system that is specified in a parameter `crs`.

F:
STATEMEN If the `crs` query parameter is **not** defined, the bounding box SHALL consist of four or six numbers, depending on whether the coordinate reference system includes a vertical axis (height or depth) and the coordinate reference system of the values SHALL be interpreted as the default coordinate reference system specified for the query type.

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

A.5.4. Requirement `/req/edr/coords-definition` Parameter `coords` definition

REQUIREMENT A.32

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):

STATEMEN

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

REQUIREMENT A.32

B:
STATEMENT 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.5.5. Requirement /req/edr/point-coords-response Parameter coords response

REQUIREMENT A.33

A:
STATEMENT Only those resources that are representative of the requested geometry that is defined by the coords parameter SHALL be part of the result set.

B:
STATEMENT The coordinates SHALL consist of a Well Known Text (WKT) POINT or (if supported by the collection) MULTIPOINT geometry string.

C:
STATEMENT If an unsupported Well Known Text (WKT) geometry is requested a 400 error SHOULD be returned.

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

STATEMENT 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 A.4

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

A.5.6. Requirement /req/edr/polygon-coords-response Parameter coords response

REQUIREMENT A.34

A:

STATEMENT Only those resources that are representative of the requested geometry that is defined by the `coords` parameter SHALL be part of the result set.

B:

STATEMENT The coordinates SHALL consist of a Well Known Text (WKT) POLYGON or (if supported by the collection) MULTIPOLYGON geometry string.

C:

STATEMENT If an unsupported Well Known Text (WKT) geometry is requested a 400 error SHOULD be returned.

D:

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

STATEMENT

```
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 A.5

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

A.5.7. Requirement /req/edr/linestring-coords-response Parameter `coords` response

REQUIREMENT A.35

A:

STATEMENT Only those resources that are representative of the requested geometry that is defined by the `coords` parameter SHALL be part of the result set.

B:

STATEMENT The coordinates SHALL consist of a Well Known Text (WKT) LINESTRING or (if supported by the collection) MULTILINESTRING geometry string.

C:

STATEMENT If an unsupported Well Known Text (WKT) geometry is requested a 400 error SHOULD be returned.

REQUIREMENT A.35

D:

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

STATEMENT `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 A.6

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

A.5.8. Requirement /req/core/datetime-definition datetime definition

REQUIREMENT A.36

A:

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

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

A.5.9. Requirement /req/core/datetime-response datetime response

REQUIREMENT A.37

A:

STATEMENT 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:

STATEMENT 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.

REQUIREMENT A.37

C:

STATEMENT 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](#)):

```
STATEMENT 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:

STATEMENT The syntax of date-time is specified by [RFC 3339, 5.6](#).

F:

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

A.5.10. Requirement /req/edr/REQ_rc-parameter-name-definition Parameter parameter-name definition

REQUIREMENT A.38

A:

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

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

A.5.11. Requirement /req/edr/parameter-name-response Parameter parameter-name response

REQUIREMENT A.39

A:

STATEMENT 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:

STATEMENT 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.5.12. Requirement /req/edr/REQ_rc-crs-definition Parameter crs definition

REQUIREMENT A.40

A:

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

STATEMENT

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

A.5.13. Requirement /req/edr/REQ_rc-crs-response Parameter crs response

REQUIREMENT A.41

A:

STATEMENT 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 specified the data will be returned in its native projection.

B:

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

REQUIREMENT A.41

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

A.5.14. Requirement `/req/edr/rc-f-definition` Parameter `f` definition

REQUIREMENT A.42

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

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

A.5.15. Requirement `/req/edr/REQ_rc-f-response` Parameter `f` response

REQUIREMENT A.43

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

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

C:
STATEMENT If `f` value is not defined the returned data should be in the default format specified by the `default_output_format` attribute of the collections metadata response. If a `default_output_format` attribute is not specified, the data return SHOULD be in the native format of the source datastore.

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

A.5.16. Requirement /req/edr/z-definition Parameter z definition

REQUIREMENT A.44

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

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

A.5.17. Requirement /req/edr/z-response Parameter z response

REQUIREMENT A.45

A:
STATEMENT 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:
STATEMENT The z can be defined as a height range by specifying a min-level and max-level separated by a forward slash “/”

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

D:
STATEMENT An Arithmetic sequence using Recurring height intervals can be specified by Rnumber of intervals/min height/height interval

E:
STATEMENT 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.5.18. Requirement /req/edr/within-definition Parameter within definition

REQUIREMENT A.46

A:

STATEMENT 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:

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

A.5.19. Requirement /req/edr/REQ_rc-within-response Parameter within response

REQUIREMENT A.47

A:

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

REQUIREMENT A.47

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

A.5.20. Requirement `/req/edr/within-units-definition` Parameter `within-units` definition

REQUIREMENT A.48

A:
STATEMENT 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:

STATEMENT

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

A.5.21. Requirement `/req/edr/REQ_rc-within-units-response` Parameter `within-units` response

REQUIREMENT A.49

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

A.5.22. Requirement `/req/edr/resolution-x-definition` Parameter `resolution-x` definition

REQUIREMENT A.50

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

```
name: resolution-x
```


REQUIREMENT A.50

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

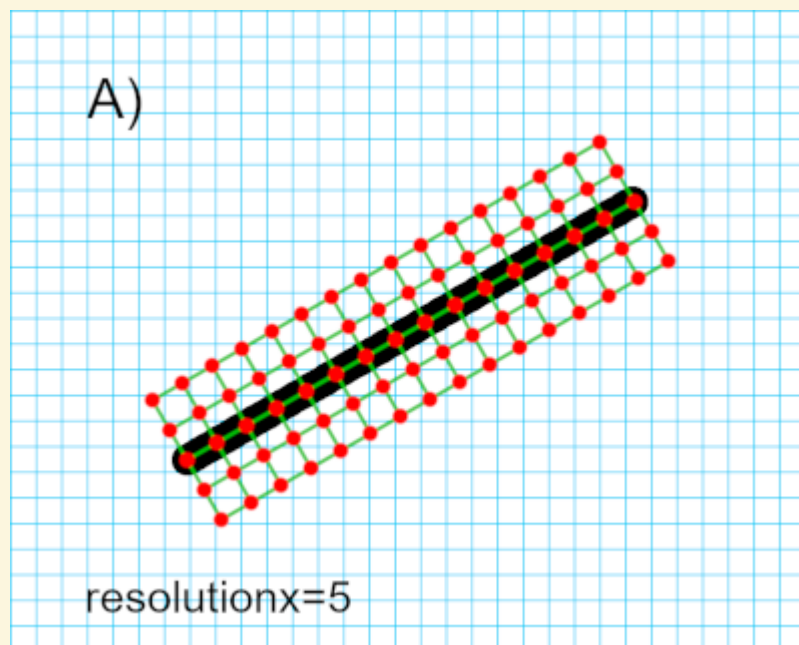
A.5.23. Requirement /req/edr/resolution-x-response Parameter resolution-x response

REQUIREMENT A.51

A:

STATEMENT 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.

Figure A.16

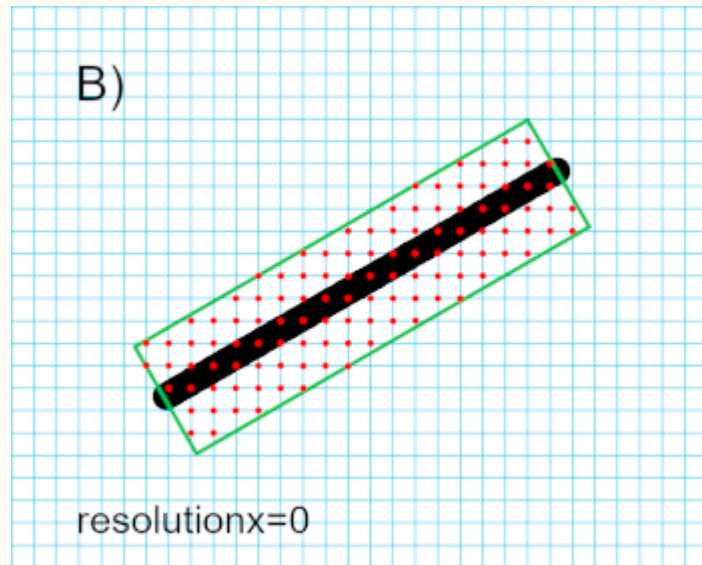


B:

STATEMENT 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.

Figure A.17

REQUIREMENT A.51



STATEMENT C:
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.5.24. Requirement /req/edr/cube-z-response Parameter z response for cube queries

REQUIREMENT A.52

STATEMENT 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.

STATEMENT B:
The z can be defined as a height range by specifying a min-level and max-level separated by a forward slash “/”

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

REQUIREMENT A.52

D:

STATEMENT An Arithmetic sequence using Recurring height intervals can be specified by Rnumber of intervals/min height/height interval

E:

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

A.5.25. Requirement /req/edr/resolution-y-definition Parameter resolution-y definition

REQUIREMENT A.53

A:

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

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

A.5.26. Requirement /req/edr/resolution-y-response Parameter resolution-y response

REQUIREMENT A.54

A:

STATEMENT 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:

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

C:

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

REQUIREMENT A.54

D:
STATEMENT 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.5.27. Requirement /req/edr/resolution-z-definition Parameter resolution-z definition

REQUIREMENT A.55

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

STATEMENT

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

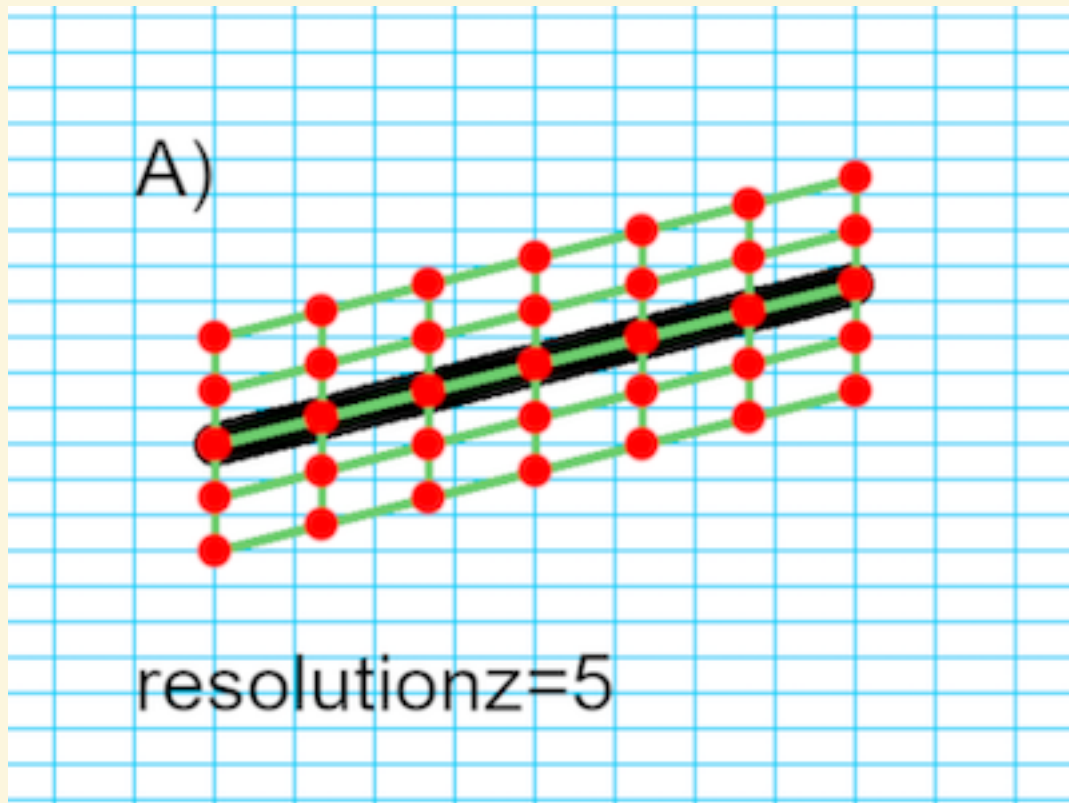
A.5.28. Requirement /req/edr/resolution-z-response Parameter resolution-z response

REQUIREMENT A.56

A:
STATEMENT 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.

Figure A.22

REQUIREMENT A.56

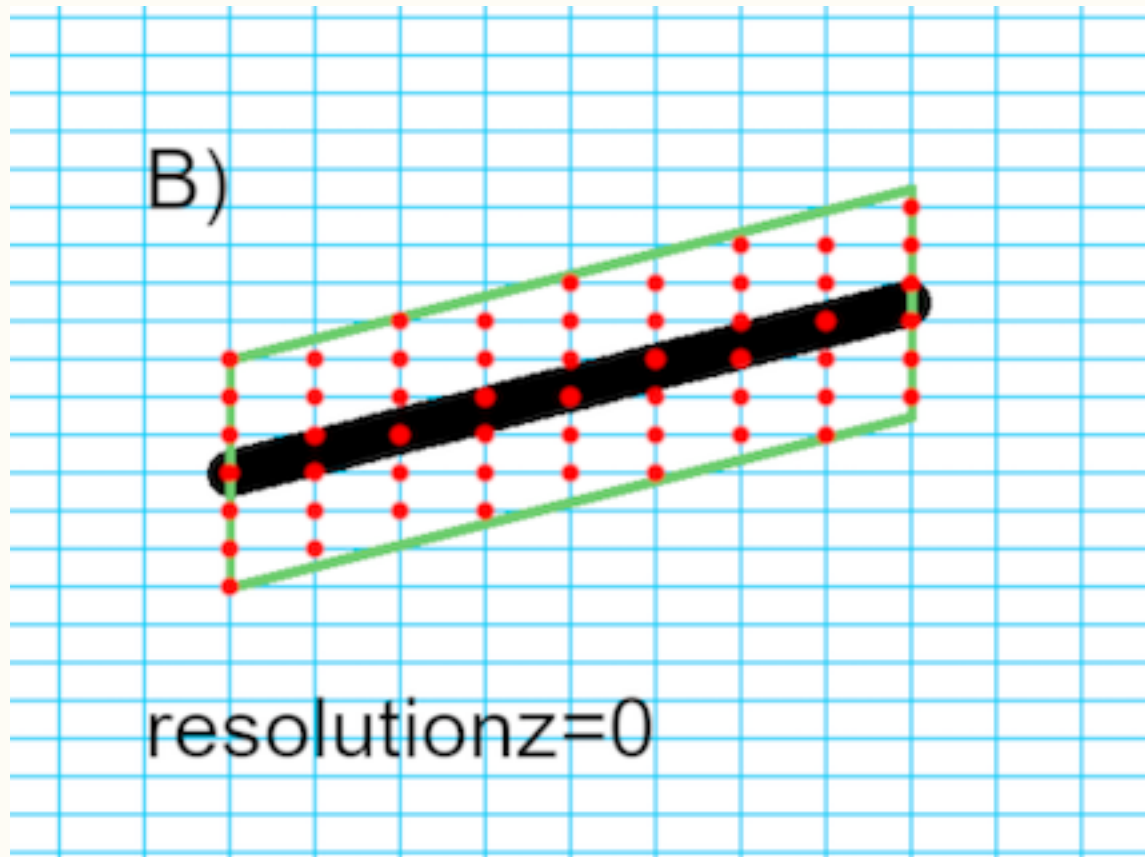


B:

STATEMENT 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.

Figure A.23

REQUIREMENT A.56



STATEMENT 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.5.29. Requirement /req/edr/REQ_rc-corridor-height-definition Parameter corridor-height definition

REQUIREMENT A.57

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

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

REQUIREMENT A.57

`explode: false`

A.5.30. Requirement /req/edr/REQ_rc-corridor-height-response Parameter corridor-height response

REQUIREMENT A.58

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

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

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

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

A.5.31. Requirement /req/edr/REQ_rc-height-units-definition Parameter height-units definition

REQUIREMENT A.59

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

A.5.32. Requirement /req/edr/height-units-response Parameter height-units response

REQUIREMENT A.60

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

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

A.5.33. Requirement /req/edr/corridor-width-definition Parameter corridor-width definition

REQUIREMENT A.61

A:
STATEMENT 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.5.34. Requirement /req/edr/REQ_rc-corridor-width-response Parameter corridor-width response

REQUIREMENT A.62

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

B:
STATEMENT The supported corridor-width width units will be supplied by the query metadata information.

REQUIREMENT A.62

`corridor-width = width`

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

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

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

A.5.35. Requirement `/req/edr/REQ_rc-width-units-definition` Parameter `width-units` definition

REQUIREMENT A.63

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

STATEMENT

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

A.5.36. Requirement `/req/edr/width-units-response` Parameter `width-units` response

REQUIREMENT A.64

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

STATEMENT

```
width-units = units
```

REQUIREMENT A.64

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

A.5.37. Requirement /req/edr/custom-dimension-definition Custom Parameter definition

REQUIREMENT A.65

A:
Each resource collection operation MAY support custom parameters with the following characteristics (using an OpenAPI Specification 3.0 fragment):

STATEMENT

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

A.5.38. Requirement /req/edr/custom-dimension-response Custom Parameter response

REQUIREMENT A.66

A:
STATEMENT If a the custom parameter is provided, only resources that have values that are valid for the range defined in the custom query parameter SHALL be part of the result set.

B:
STATEMENT The custom query parameter can be defined as a value range by specifying a min-value and max-value seperated by a forward slash "/"

C:
STATEMENT A list of query values can be defined be specifying a comma delimited list of values value1, value2, value3 etc

D:
STATEMENT An Arithmetic sequence using Recurring value intervals can be specified by Rnumber of intervals/min value/value interval

REQUIREMENT A.66

E:

STATEMENT If a custom dimension is defined in the collections response but not included in the query request all valid values should be returned with no subsetting by the custom dimension.

```
single-value = value
interval-closed = min-value "/" max-value
value-list = value1 "," value2 "," value3
repeating-interval = "R"number of intervals "/" min-value "/" value to
increment by
```

A.6. Requirements Class “JSON” in Detail

A.6.1. Requirements Class: JSON

REQUIREMENTS CLASS: JSON REQUIREMENTS CLASS

OBLIGATION	requirement
TARGET TYPE	Web API
PREREQUISITE	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/json
NORMATIVE STATEMENTS	Requirement A.-1 Requirement A.-2

REQUIREMENT A.67

A:

STATEMENT Every 200-response with the media type `application/json` SHALL be a payload encoded according to the [JSON Interchange Format](#).

B:

STATEMENT 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`.

REQUIREMENT A.67

C:

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

D:

STATEMENT 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

A:

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

- `application/json` for all resources.

A.7. Requirements Class “GeoJSON” in Detail

A.7.1. Requirements Class: GeoJSON

REQUIREMENTS CLASS: GEOJSON REQUIREMENTS CLASS

OBLIGATION	requirement
TARGET TYPE	Web API
PREREQUISITE	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
NORMATIVE STATEMENTS	Requirement A.-1 Requirement A.-2

REQUIREMENT A.69

STATEMENT A:
Every 200-response with the media type `application/geo+json` SHALL be

REQUIREMENT A.69

- a [GeoJSON FeatureCollection Object](#) for features, and
- a [GeoJSON Feature Object](#) for a single feature.

STATEMENT 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`.

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

STATEMENT 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.70

STATEMENT 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 “EDR GeoJSON” in Detail

A.8.1. Requirements Class: EDR GeoJSON

REQUIREMENTS CLASS: EDR GEOJSON REQUIREMENTS CLASS

OBLIGATION requirement

TARGET TYPE Web API

REQUIREMENTS CLASS: EDR GEOJSON REQUIREMENTS CLASS

PREREQUISITE	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
NORMATIVE STATEMENTS	Requirement A.-1 Requirement A.-2

A.8.2. Requirement /req/edr-geojson/content

REQUIREMENT A.71

STATEMENT A: Every 200-response with the media type `application/geo+json` SHALL be

- a [EDR GeoJSON FeatureCollection Object](#) for features, and
- a [EDR GeoJSON Feature Object](#) for a single feature.

STATEMENT 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`.

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

STATEMENT 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.72

STATEMENT 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.9. Requirements Class “CoverageJSON” in Detail

A.9.1. Requirements Class: CoverageJSON

REQUIREMENTS CLASS: COVERAGEJSON REQUIREMENTS CLASS

OBLIGATION	requirement
TARGET TYPE	Web API
PREREQUISITE	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
NORMATIVE STATEMENTS	Requirement A.-1 Requirement A.-2

REQUIREMENT A.73

A:
STATEMENT Every 200-response with the media type `application/prs.coverage+json` SHALL be

- a [CoverageJSON Object](#)

B:
STATEMENT 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.74

A:
STATEMENT 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.10. Requirements Class “HTML” in Detail

A.10.1. Requirements Class: HTML

REQUIREMENTS CLASS: HTML REQUIREMENTS CLASS	
OBLIGATION	requirement
TARGET TYPE	Web API
PREREQUISITE	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
NORMATIVE STATEMENTS	Requirement A.-1 Requirement A.-2

REQUIREMENT A.75

A:

Every 200-response of the server with the media type `text/html` SHALL be a [HTML 5 document](#) that **STATEMENT** 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.76

STATEMENT A:

Every 200-response of an operation of the server SHALL support the media type `text/html`.

A.11. Requirements Class “OpenAPI 3.0” in Detail

A.11.1. Requirements Class: OpenAPI 3.0

REQUIREMENTS CLASS: OPENAPI 3.0 REQUIREMENTS CLASS

OBLIGATION	requirement
TARGET TYPE	Web API
PREREQUISITES	http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/core http://www.opengis.net/spec/ogcapi-common-1/1.0/req/oas30 OpenAPI Specification 3.0.3
NORMATIVE STATEMENTS	Requirement A.-1 Requirement A.-2 Requirement A.-3 Requirement A.-4 Requirement A.-5 Requirement A.-6

A.11.2. Requirement /req/oas30/oas-impl OpenAPI 3.0 implementation

REQUIREMENT A.77

STATEMENT A:
The server SHALL implement all capabilities specified in the OpenAPI definition.

A.11.3. Requirement /req/oas30/oas-definition-1

REQUIREMENT A.78

STATEMENT 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.11.4. Requirement /req/oas30/oas-definition-2 OpenAPI 3.0 conformance

REQUIREMENT A.79

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

A.11.5. Requirement /req/oas30/completeness OpenAPI 3.0 Completeness

REQUIREMENT A.80

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

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

A.11.6. Requirement /req/oas30/exceptions-codes OpenAPI 3.0 Exception codes

REQUIREMENT A.81

A:
STATEMENT For error situations that originate from an API server, the API definition SHALL cover all applicable HTTP Status Codes.

A.11.7. Requirement /req/oas30/security OpenAPI 3.0 Security

REQUIREMENT A.82

A:
STATEMENT 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 Programming 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 appropriate, TLS.
Requirement	/req/core/http
Test Method	<ol style="list-style-type: none">1. All compliance tests shall be configured to use the HTTP 1.1 protocol exclusively.2. 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	<ol style="list-style-type: none">1. Issue an HTTP GET request to the URL {root}/2. Validate that a document was returned with a status code 2003. 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 required structure and contents.
Requirement	/req/core/root-success

Test Method	<p>Validate the landing page for all supported media types using the resources and tests identified in Table B.5</p> <p>For formats that require manual inspection, perform the following:</p> <ol style="list-style-type: none"> Validate that the landing page includes a “service-desc” and/or “service-doc” link to an API Definition Validate that the landing page includes a “conformance” link to the conformance class declaration 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

FORMAT	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	<ol style="list-style-type: none"> Construct a path for each API Definition link on the landing page Issue a HTTP GET request on each path Validate that a document was returned with a status code 200 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	<ol style="list-style-type: none"> 1. Construct a path for each “conformance” link on the landing page as well as for the {root}/conformance path. 2. Issue an HTTP GET request on each path 3. Validate that a document was returned with a status code 200 4. 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	<ol style="list-style-type: none"> 1. Validate the response document against OpenAPI 3.0 schema conf Classes.yaml 2. Validate that the document includes the conformance class “http://www.opengis.net/spec/ogcapi-edr-1/1.0/conf/core” 3. 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 http://www.opengis.net/def/crs/OGC/1.3/CRS84 coordinate reference system unless otherwise requested by the client.
Requirement	/req/core/crs84
Test Method	<ol style="list-style-type: none">1. Do not specify a coordinate reference system in any request. All spatial data should be in the http://www.opengis.net/def/crs/OGC/1.3/CRS84 reference system.2. 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	<ol style="list-style-type: none"> 1. Issue an HTTP GET request to the URL {root}/collections 2. Validate that a document was returned with a status code 200 3. 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	<ol style="list-style-type: none"> 1. Validate that all response documents comply with /req/core/rc-collection-info-links 2. In case the response includes a “crs” property, validate that it includes a valid Well Known Text definition 3. 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

FORMAT	SCHEMA DOCUMENT	TEST ID
HTML	collections.yaml	/conf/html/content
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	<p>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.</p> <ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ol style="list-style-type: none"> 1. Verify that all collections listed in the collections array of the Collections Metadata exist. 2. Verify that each collection entry includes an identifier. 3. Verify that each collection entry includes links in accordance with /conf/core/rc-collection-info-links. 4. Verify that each collection entry includes an extent property in accordance with /conf/core/rc-extent 5. Verify that the collection entry includes a data_queries property in accordance with /req/edr/rc-data-queries 6. Verify that if the collection data_queries entry includes a crs property, the property complies with /conf/edr/rc-crs 7. Verify that each collection entry includes a parameter_names property, and the property complies with /req/edr/rc-parameters 8. 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-data-queries
-------------------------	----------------------------------

Test Purpose	Validate that the <code>data_queries</code> section in the collection is correctly defined.
Requirement	<code>/req/edr/rc-data-queries</code>
Test Method	<ol style="list-style-type: none"> Verify that at least one of the following data query types are defined in the data query section: <ul style="list-style-type: none"> position radius area cube trajectory corridor items locations verify that all query types defined comply with <code>/req/edr/rc-common_query_type</code>

Table B.21 – Abstract Test 16

Abstract Test 16	<code>/conf/edr/rc-common-query-type</code>
Test Purpose	Validate variables property for a query data type in the <code>data_queries</code> section in the collection is correctly defined.
Requirement	<code>/req/edr/rc-common-query-type</code>
Test Method	<ol style="list-style-type: none"> Verify that the <code>data_queries</code> object has a <code>link</code> property Verify that the <code>link</code> property has a <code>href</code> property Verify that the <code>link</code> property has a <code>rel</code> property Verify that the <code>link</code> property has a <code>variables</code> property Verify that if the <code>data_queries</code> object is of type <code>position,area,trajectory</code> or <code>locations</code> the property complies with <code>/req/edr/rc-common-variables</code> Verify that if the <code>data_queries</code> object is of type <code>radius`</code> the property complies with <code>/req/edr/rc-radius-variables</code> Verify that if the <code>data_queries</code> object is of type <code>cube`</code> the property complies with <code>/req/edr/rc-cube-variables</code> Verify that if the <code>data_queries</code> object is of type <code>corridor`</code> the property complies with <code>/req/edr/rc-corridor-variables</code> Verify that if the <code>data_queries</code> object is of type <code>items`</code> the property complies with <code>/req/edr/rc-items-variables</code>

Table B.22 – Abstract Test 17

Abstract Test 17	<code>/conf/edr/rc-common-variables</code>
Test Purpose	Validate variables property for a query data type in the <code>data_queries</code> section in the collection is correctly defined.

Requirement	/req/edr/rc-common-variables
Test Method	<ol style="list-style-type: none"> 1. Verify that the <code>variables</code> object has a <code>query_type</code> property 2. Verify that the value of the <code>query_type</code> property matches the name of the data query type 3. If the <code>variables</code> object has an <code>output_formats</code> property verify that it is an array of strings 4. If the <code>variables</code> object has a <code>default_output_format</code> property verify that the value is in either the <code>output_formats</code> property of the <code>variables</code> object or the <code>output_formats</code> property of the parent collection. 5. If the <code>variables</code> object has a <code>crs_details</code> property verify that it is an array of <code>crs</code> objects, and each member object of the array has <code>crs_details</code> and <code>wkt</code> properties.

Table B.23 – Abstract Test 18

Abstract Test 18	/conf/edr/rc-radius-variables
Test Purpose	Validate <code>variables</code> property for a query data type in the <code>data_queries</code> section in the collection is correctly defined.
Requirement	/req/edr/rc-radius-variables
Test Method	<ol style="list-style-type: none"> 1. Verify that the <code>variables</code> property complies with /req/edr/rc-common-variables 2. Verify that the <code>variables</code> property has a <code>within_units</code> property 3. Verify that the <code>within_units</code> property is a string array

Table B.24 – Abstract Test 19

Abstract Test 19	/conf/edr/rc-cube-variables
Test Purpose	Validate <code>variables</code> property for a query data type in the <code>data_queries</code> section in the collection is correctly defined.
Requirement	/req/edr/rc-cube-variables
Test Method	<ol style="list-style-type: none"> 1. Verify that the <code>variables</code> property complies with /req/edr/rc-common-variables 2. Verify that the <code>variables</code> property has a <code>height_units</code> property 3. Verify that the <code>height_units</code> property is a string array

Table B.25 – Abstract Test 20

Abstract Test 20	/conf/edr/rc-corridor-variables
------------------	---------------------------------

Test Purpose	Validate variables property for a query data type in the data_queries section in the collection is correctly defined.
Requirement	/req/edr/rc-corridor-variables
Test Method	<ol style="list-style-type: none"> 1. Verify that the variables property complies with /req/edr/rc-common-variables 2. Verify that the variables property has a height_units property 3. Verify that the height_units property is a string array 4. Verify that the variables property has a width_units property 5. Verify that the width_units property is a string array

Table B.26 – Abstract Test 21

Abstract Test 21	/conf/edr/rc-items-variables
Test Purpose	Validate variables property for a query data type in the data_queries section in the collection is correctly defined.
Requirement	/req/edr/rc-items-variables
Test Method	<ol style="list-style-type: none"> 1. Verify that the variables object has a query_type property 2. Verify that the value of the query_type property is items

Table B.27 – Abstract Test 22

Abstract Test 22	/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	<ol style="list-style-type: none"> 1. Verify that all links include the rel and type link parameters.

B.3.4.3. Collection Links

Table B.28 – Abstract Test 23

Abstract Test 23	/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	<p>Verify that the response document includes:</p> <ol style="list-style-type: none"> 1. a link to this response document (relation: self), 2. a link to the response document in every other media type supported by the server (relation: alternate). <p>Verify that all links include the rel and type link parameters.</p>

B.3.4.4. Collection Parameters

Table B.29 – Abstract Test 24

Abstract Test 24	/conf/edr/rc-parameters
Test Purpose	Validate that each parameter in a collection is correctly defined.
Requirement	/req/edr/rc-parameters
Test Method	<ol style="list-style-type: none"> 1. Verify that all parameters listed in a collection have the required properties. 2. Verify that each parameter property has a unique name (in the collection). 3. Verify that each parameter property has a type property. 4. Verify that each parameter property has an observedProperty property. 5. Verify that the observedProperty property has a label property with correctly defined i18n compliant values. 6. Verify that the observedProperty property has an id property.

B.4. Conformance Class JSON

Table B.30 – 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.31 – Abstract Test 25

Abstract Test 25	/conf/json/definition
Test Purpose	Verify support for JSON
Requirement	/req/json/definition
Test Method	<ol style="list-style-type: none"> 1. A resource is requested with response media type of application/json 2. All 200-responses SHALL support the media type: application/json

B.4.2. JSON Content

Table B.32 – Abstract Test 26

Abstract Test 26	/conf/json/content
Test Purpose	Verify the content of a JSON document given an input document and schema.
Requirement	/req/json/content
Test Method	<ol style="list-style-type: none"> 1. Validate that the document is a JSON document. 2. Validate the document against the schema using a JSON Schema validator.

B.5. Conformance Class GeoJSON

Table B.33 – 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.34 – Abstract Test 27

Abstract Test 27	/conf/geojson/definition
Test Purpose	Verify support for JSON and GeoJSON
Requirement	/req/geojson/definition
Test Method	<ol style="list-style-type: none"> 1. A resource is requested with response media type of <code>application/geo+json</code> 2. All 200-responses SHALL support the following media types: <ul style="list-style-type: none"> • <code>application/geo+json</code> for resources that include feature content, and • <code>application/json</code> for all other resources.

B.5.2. GeoJSON Content

Table B.35 – Abstract Test 28

Abstract Test 28	/conf/geojson/content
Test Purpose	Verify the content of a GeoJSON document given an input document and schema.
Requirement	/req/geojson/content
Test Method	<ol style="list-style-type: none"> 1. Validate that the document is a GeoJSON document. 2. Validate the document against the schema using a JSON Schema validator. 3. Validate the document against the schema using a GeoJSON Schema validator.

B.6. Conformance Class EDR GeoJSON

Table B.36 – 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.37 – Abstract Test 29

Abstract Test 29	/conf/edr-geojson/definition
Test Purpose	Verify support for the EDR GeoJSON Schema
Requirement	/req/edr-geojson/definition
Test Method	<ol style="list-style-type: none">1. A resource is requested with response media type of <code>application/json</code> and adheres to the EDR Feature Collection GeoJSON Schema.2. All 200-responses SHALL support the following media types:<ul style="list-style-type: none">• <code>application/json</code> for resources

B.6.2. EDR GeoJSON Content

Table B.38 – Abstract Test 30

Abstract Test 30	/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	<ol style="list-style-type: none"> 1. Validate that the document is an EDR GeoJSON document. 2. Validate the document against one of the EDR GeoJSON schemas: <ul style="list-style-type: none"> • FeatureCollection: edrFeatureCollectionGeoJSON.yaml • Feature: featureGeoJSON.yaml • GeometryCollection: geometrycollectionGeoJSON.yaml using a JSON Schema validator.
-------------	---

B.7. Conformance Class CoverageJSON

Table B.39 – 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.40 – Abstract Test 31

Abstract Test 31	/conf/covjson/definition
Test Purpose	Verify support for CoverageJSON
Requirement	/req/covjson/definition
Test Method	<ol style="list-style-type: none"> 1. A resource is requested with response media type of <code>application/prs.coverage+json</code> 2. All 200-responses SHALL support the following media types: <ul style="list-style-type: none"> • <code>application/prs.coverage+json</code> for resources

B.7.2. CoverageJSON Content

Table B.41 – Abstract Test 32

Abstract Test 32	/conf/covjson/content
Test Purpose	Verify the content of a CoverageJSON document given an input document and schema.
Requirement	/req/covjson/content
Test Method	<ol style="list-style-type: none"> 1. Validate that the document is a CoverageJSON document. 2. Validate the document against the coverageJSON.yaml schema using a JSON Schema validator.

B.8. Conformance Class HTML

Table B.42 – 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.43 – Abstract Test 33

Abstract Test 33	/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 <code>text/html</code>

B.8.2. HTML Content

Table B.44 – Abstract Test 34

Abstract Test 34	/conf/html/content
Test Purpose	Verify the content of an HTML document given an input document and schema.
Requirement	/req/html/content
Test Method	<ol style="list-style-type: none">1. Validate that the document is an HTML 5 document2. Manually inspect the document against the schema.

B.9. Conformance Class OpenAPI 3.0

Table B.45 – 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.46 – Abstract Test 35

Abstract Test 35	/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.47 – Abstract Test 36

Abstract Test 36	/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.48 – Abstract Test 37

Abstract Test 37	/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	<ol style="list-style-type: none"> 1. Verify that an OpenAPI definition in JSON is available using the media type <code>application/vnd.oai.openapi+json;version=3.0</code> and link relation <code>service-desc</code> 2. Verify that an HTML version of the API definition is available using the media type <code>text/html</code> and link relation <code>service-doc</code>.

Table B.49 – Abstract Test 38

Abstract Test 38	/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.50 – Abstract Test 39

Abstract Test 39	/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	<ol style="list-style-type: none"> 1. Construct a path from each URL template including all server URL options and all enumerated path parameters. 2. 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.51 – Abstract Test 40

Abstract Test 40	/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	<ol style="list-style-type: none"> 1. Identify all authentication protocols supported by the API. 2. Validate that each authentication protocol is described in the Open API document by a Security Schema Object and its use specified by a Security Requirement Object.

B.10. Conformance Class Queries

Table B.52 – 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.53 – Abstract Test 41

Abstract Test 41	/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	<ol style="list-style-type: none">1. No query parameters are specified2. Validate that a document was returned with a status code 400.

Table B.54 – Abstract Test 42

Abstract Test 42	/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	<ol style="list-style-type: none">1. coords query parameter is not specified2. Validate that a document was returned with a status code 400.

Table B.55 – Abstract Test 43

Abstract Test 43	/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 or MULTIPOINT Well Known Text value.
Requirement	/req/queries/position
Test Method	<ol style="list-style-type: none">1. Check coords query parameter is a valid Well Known Text Point or MultiPoint value2. Validate that a document was returned with a status code 400.

Table B.56 – Abstract Test 44

Abstract Test 44	/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	<ol style="list-style-type: none"> 1. Test with valid query parameters 2. Validate that a document was returned with a status code 200. <p>Repeat these tests using the following parameter tests:</p> <p>Coordinates</p> <ul style="list-style-type: none"> • Parameter /req/edr/coords-definition • Response /req/edr/coords-response <p>VerticalLevel</p> <ul style="list-style-type: none"> • Parameter /req/edr/z-definition • Response /req/edr/z-response <p>Parameters</p> <ul style="list-style-type: none"> • Parameter /req/edr/REQ_rc-parameter-name-definition • Response /req/edr/parameter-name-response <p>DateTime</p> <ul style="list-style-type: none"> • Parameter /req/core/datetime-definition • Response /req/core/datetime-response <p>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.</p>

Table B.57 – Abstract Test 45

Abstract Test 45	/conf/edr/rc-coords-definition
Test Purpose	Validate that the coords query parameters are constructed correctly.
Requirement	/req/edr/coords-definition
Test Method	<p>Verify that the coords query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: coords in: query required: true schema: type: string style: form explode: false </pre>

Use a coords value in all requests:

- A valid Well-Known Text (WKT) representation of geometry string

Table B.58 – Abstract Test 46

Abstract Test 46	/conf/edr/rc-coords-response
Test Purpose	Validate that the coords query parameters are processed correctly.
Requirement	/req/edr/coords-response
Test Method	1. Verify that only resources that have a spatial geometry that intersects the coordinates are returned as part of the result set.
Test Method	1. Verify coords values are valid for the specified coordinate reference system 2. Verify that the coordinate reference system of the geometries is 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.59 – Abstract Test 47

Abstract Test 47	/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): <code>name: z in: query required: false schema: type: string style: form explode: false</code>

Table B.60 – Abstract Test 48

Abstract Test 48	/conf/edr/rc-z-response
Test Purpose	Validate that the vertical level query parameters are processed correctly.
Requirement	/req/edr/z-response

Test Method	<ol style="list-style-type: none"> 1. Verify that only resources that have a vertical geometry that intersects the vertical information in the z parameter were included in the result set 2. Validate that the vertical level parameter complies with the syntax described in /req/edr/z-response.
-------------	---

Table B.61 – Abstract Test 49

Abstract Test 49	/conf/core/datetime-definition
Test Purpose	Validate that the datetime query parameters are constructed correctly.
Requirement	/req/core/datetime-definition
Test Method	<p>Verify that the datetime query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: datetime in: query required: false schema: type: string style: form explode: false </pre>

Table B.62 – Abstract Test 50

Abstract Test 50	/conf/core/datetime-response
Test Purpose	Validate that the datetime query parameters are processed correctly.
Requirement	/req/core/datetime-response
Test Method	<ol style="list-style-type: none"> 1. Verify that only resources that have a temporal geometry that intersects the temporal information in the datetime parameter were included in the result set. 2. Verify that all resources in the collection that are not associated with a temporal geometry are included in the result set. 3. Validate that the datetime parameter complies with the syntax described in /req/core/datetime-response.

Table B.63 – Abstract Test 51

Abstract Test 51	/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	<p>Verify that the parameter-name query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: parameter-name in: query required: false schema: type: string style: form explode: false </pre>

Table B.64 – Abstract Test 52

Abstract Test 52	/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	<ol style="list-style-type: none"> 1. Verify that only resources for the requested parameters were included in the result set 2. Validate that the parameter-name parameter complies with the syntax described in /req/edr/parameter-name-response.

Table B.65 – Abstract Test 53

Abstract Test 53	/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	<p>Verify that the crs query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: crs in: query required: false schema: type: string style: form explode: false </pre>

Table B.66 – Abstract Test 54

Abstract Test 54	/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	1. Verify that the geometry of the resources returned are valid for the requested coordinate reference system
Test Method	1. Verify that all crs values defined in the collections metadata are supported by the collection
Test Method	1. Verify that all crs values not defined in the collections metadata will generate a HTTP 400 error 2. Validate that the crs parameter complies with the syntax described in /req/edr/REQ_rc-crs-response.

Table B.67 – Abstract Test 55

Abstract Test 55	/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): <pre> name: f in: query required: false schema: type: string style: form explode: false </pre>

Table B.68 – Abstract Test 56

Abstract Test 56	/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	1. Verify that the response is returned in the requested data format
Test Method	1. Verify that all output format values defined in the collections metadata are supported by the collection 2. Validate that the f parameter complies with the syntax described in /req/edr/REQ_rc-f-response.

B.10.1.2. Area

Table B.69 – Abstract Test 57

Abstract Test 57	/conf/area
Test Purpose	Validate that an error is returned by an Area query if no query parameters are specified.
Requirement	/req/edr/rc-area
Test Method	<ol style="list-style-type: none">1. No query parameters are specified2. Validate that a document was returned with a status code 400.

Table B.70 – Abstract Test 58

Abstract Test 58	/conf/area
Test Purpose	Validate that an error is returned by an Area query when the coords query parameter is not specified.
Requirement	/req/edr/rc-area
Test Method	<ol style="list-style-type: none">1. coords query parameter is not specified2. Validate that a document was returned with a status code 400.

Table B.71 – Abstract Test 59

Abstract Test 59	/conf/area
Test Purpose	Validate that an error is returned by an Area query when the coords query parameter does not contain a valid POLYGON or MULTIPOLYGON Well Known Text value.
Requirement	/req/edr/rc-area
Test Method	<ol style="list-style-type: none">1. Check coords query parameter is a valid Well Known Text Polygon or MultiPolygon value2. Validate that a document was returned with a status code 400.

Table B.72 – Abstract Test 60

Abstract Test 60	/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	<ol style="list-style-type: none"> 1. Test with valid query parameters 2. Validate that a document was returned with a status code 200. <p>Repeat these tests using the following parameter tests:</p> <p>Coordinates</p> <ul style="list-style-type: none"> • Parameter /req/edr/coords-definition • Response /req/edr/coords-response <p>VerticalLevel</p> <ul style="list-style-type: none"> • Parameter /req/edr/z-definition • Response /req/edr/z-response <p>Parameters</p> <ul style="list-style-type: none"> • Parameter /req/edr/REQ_rc-parameter-name-definition • Response /req/edr/parameter-name-response <p>DateTime</p> <ul style="list-style-type: none"> • Parameter /req/core/datetime-definition • Response /req/core/datetime-response <p>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.</p>

Table B.73 – Abstract Test 61

Abstract Test 61	/conf/edr/rc-coords-definition
Test Purpose	Validate that the coords query parameters are constructed correctly.
Requirement	/req/edr/coords-definition
Test Method	<p>Verify that the coords query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: coords in: query required: true schema: type: string style: form explode: false </pre>

Use a coords value in all requests:

- A valid Well-Known Text (WKT) representation of geometry string

Table B.74 – Abstract Test 62

Abstract Test 62	/conf/edr/rc-coords-response
Test Purpose	Validate that the coords query parameters are processed correctly.
Requirement	/req/edr/coords-response
Test Method	1. Verify that only resources that have a spatial geometry that intersects the coordinates are returned as part of the result set.
Test Method	1. Verify coords values are valid for the specified coordinate reference system 2. Verify that the coordinate reference system of the geometries is 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.75 – Abstract Test 63

Abstract Test 63	/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): <code>name: z in: query required: false schema: type: string style: form explode: false</code>

Table B.76 – Abstract Test 64

Abstract Test 64	/conf/edr/rc-z-response
Test Purpose	Validate that the vertical level query parameters are processed correctly.
Requirement	/req/edr/z-response

Test Method	<ol style="list-style-type: none"> 1. Verify that only resources that have a vertical geometry that intersects the vertical information in the z parameter were included in the result set 2. Validate that the vertical level parameter complies with the syntax described in /req/edr/z-response.
-------------	---

Table B.77 – Abstract Test 65

Abstract Test 65	/conf/core/datetime-definition
Test Purpose	Validate that the datetime query parameters are constructed correctly.
Requirement	/req/core/datetime-definition
Test Method	<p>Verify that the datetime query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: datetime in: query required: false schema: type: string style: form explode: false </pre>

Table B.78 – Abstract Test 66

Abstract Test 66	/conf/core/datetime-response
Test Purpose	Validate that the datetime query parameters are processed correctly.
Requirement	/req/core/datetime-response
Test Method	<ol style="list-style-type: none"> 1. Verify that only resources that have a temporal geometry that intersects the temporal information in the datetime parameter were included in the result set. 2. Verify that all resources in the collection that are not associated with a temporal geometry are included in the result set. 3. Validate that the datetime parameter complies with the syntax described in /req/core/datetime-response.

Table B.79 – Abstract Test 67

Abstract Test 67	/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	<p>Verify that the parameter-name query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: parameter-name in: query required: false schema: type: string style: form explode: false </pre>

Table B.80 – Abstract Test 68

Abstract Test 68	/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	<ol style="list-style-type: none"> 1. Verify that only resources for the requested parameters were included in the result set 2. Validate that the parameter-name parameter complies with the syntax described in /req/edr/parameter-name-response.

Table B.81 – Abstract Test 69

Abstract Test 69	/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	<p>Verify that the crs query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: crs in: query required: false schema: type: string style: form explode: false </pre>

Table B.82 – Abstract Test 70

Abstract Test 70	/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	1. Verify that the geometry of the resources returned are valid for the requested coordinate reference system
Test Method	1. Verify that all crs values defined in the collections metadata are supported by the collection
Test Method	1. Verify that all crs values not defined in the collections metadata will generate a HTTP 400 error 2. Validate that the crs parameter complies with the syntax described in /req/edr/REQ_rc-crs-response.

Table B.83 – Abstract Test 71

Abstract Test 71	/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): <pre> name: f in: query required: false schema: type: string style: form explode: false </pre>

Table B.84 – Abstract Test 72

Abstract Test 72	/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	1. Verify that the response is returned in the requested data format
Test Method	1. Verify that all output format values defined in the collections metadata are supported by the collection 2. Validate that the f parameter complies with the syntax described in /req/edr/REQ_rc-f-response.

B.10.1.3. Cube

Table B.85 – Abstract Test 73

Abstract Test 73	/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	<ol style="list-style-type: none">1. No query parameters are specified2. Validate that a document was returned with a status code 400.

Table B.86 – Abstract Test 74

Abstract Test 74	/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	<ol style="list-style-type: none">1. bbox query parameter is not specified2. Validate that a document was returned with a status code 400.

Table B.87 – Abstract Test 75

Abstract Test 75	/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	<ol style="list-style-type: none">1. Check bbox query parameter is valid2. Validate that a document was returned with a status code 400.

Table B.88 – Abstract Test 76

Abstract Test 76	/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	<ol style="list-style-type: none"> 1. Test with valid query parameters 2. Validate that a document was returned with a status code 200. <p>Repeat these tests using the following parameter tests:</p> <p>bbox</p> <ul style="list-style-type: none"> • Parameter /req/core/rc-bbox-definition • Response /req/core/rc-bbox-response <p>VerticalLevel</p> <ul style="list-style-type: none"> • Parameter /req/edr/z-definition • Response /req/edr/cube-z-response <p>Parameters</p> <ul style="list-style-type: none"> • Parameter /req/edr/REQ_rc-parameter-name-definition • Response /req/edr/parameter-name-response <p>DateTime</p> <ul style="list-style-type: none"> • Parameter /req/core/datetime-definition • Response /req/core/datetime-response <p>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.</p>

Table B.89 – Abstract Test 77

Abstract Test 77	/conf/edr/rc-coords-definition
Test Purpose	Validate that the coords query parameters are constructed correctly.
Requirement	/req/edr/coords-definition
Test Method	<p>Verify that the coords query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre>name: coords in: query required: true schema: type: string style: form explode: false</pre> <p>Use a coords value in all requests:</p> <ul style="list-style-type: none"> • A valid Well-Known Text (WKT) representation of geometry string

Table B.90 – Abstract Test 78

Abstract Test 78	/conf/edr/rc-coords-response
Test Purpose	Validate that the coords query parameters are processed correctly.
Requirement	/req/edr/coords-response
Test Method	1. Verify that only resources that have a spatial geometry that intersects the coordinates are returned as part of the result set.
Test Method	1. Verify coords values are valid for the specified coordinate reference system 2. Verify that the coordinate reference system of the geometries is 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.91 – Abstract Test 79

Abstract Test 79	/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): <pre>name: z in: query required: false schema: type: string style: form explode: false</pre>

Table B.92 – Abstract Test 80

Abstract Test 80	/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	1. Verify that only resources that have a vertical geometry that intersects the vertical information in the z parameter were included in the result set

2. Validate that the vertical level parameter complies with the syntax described in /req/edr/cube-z-response.

Table B.93 – Abstract Test 81

Abstract Test 81	/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): <pre>name: datetime in: query required: false schema: type: string style: form explode: false</pre>

Table B.94 – Abstract Test 82

Abstract Test 82	/conf/core/datetime-response
Test Purpose	Validate that the datetime query parameters are processed correctly.
Requirement	/req/core/datetime-response
Test Method	<ol style="list-style-type: none"> 1. Verify that only resources that have a temporal geometry that intersects the temporal information in the datetime parameter were included in the result set. 2. Verify that all resources in the collection that are not associated with a temporal geometry are included in the result set. 3. Validate that the datetime parameter complies with the syntax described in /req/core/datetime-response.

Table B.95 – Abstract Test 83

Abstract Test 83	/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.96 – Abstract Test 84

Abstract Test 84	/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	<ol style="list-style-type: none"> 1. Verify that only resources for the requested parameters were included in the result set 2. Validate that the parameter-name parameter complies with the syntax described in /req/edr/parameter-name-response.

Table B.97 – Abstract Test 85

Abstract Test 85	/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	<p>Verify that the crs query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: crs in: query required: false schema: type: string style: form explode: false </pre>

Table B.98 – Abstract Test 86

Abstract Test 86	/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	1. Verify that the geometry of the resources returned are valid for the requested coordinate reference system
Test Method	1. Verify that all crs values defined in the collections metadata are supported by the collection
Test Method	1. Verify that all crs values not defined in the collections metadata will generate a HTTP 400 error 2. Validate that the crs parameter complies with the syntax described in /req/edr/REQ_rc-crs-response.

Table B.99 – Abstract Test 87

Abstract Test 87	/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): <pre> name: f in: query required: false schema: type: string style: form explode: false </pre>

Table B.100 – Abstract Test 88

Abstract Test 88	/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	1. Verify that the response is returned in the requested data format
Test Method	1. Verify that all output format values defined in the collections metadata are supported by the collection 2. Validate that the f parameter complies with the syntax described in /req/edr/REQ_rc-f-response.

B.10.1.4. Trajectory

Table B.101 – Abstract Test 89

Abstract Test 89	/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	<ol style="list-style-type: none"> 1. No query parameters are specified 2. 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 is not specified.
Requirement	/req/edr/rc-trajectory
Test Method	<ol style="list-style-type: none"> 1. coords query parameter is not specified 2. 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 an error is returned by a Trajectory query when the coords query parameter does not contain a valid LINESTRING or MULTILINESTRING Well Known Text value.
Requirement	/req/edr/rc-trajectory
Test Method	<ol style="list-style-type: none"> 1. Check coords query parameter is a valid Well Known Text LineString or MultiLineString value 2. Validate that a document was returned with a status code 400.

Table B.104 – Abstract Test 92

Abstract Test 92	/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 or MULTILINESTRINGM Well Known Text value.

Requirement	/req/edr/rc-trajectory
Test Method	<ol style="list-style-type: none"> 1. Check coords query parameter with time parameter is a valid Well Known Text LINESTRINGM or MULTILINESTRINGM value, the M coordinate must be a valid Epoch value (as known as UNIX time) 2. Validate that a document was returned with a status code 400.

Table B.105 – Abstract Test 93

Abstract Test 93	/conf/trajectory
Test Purpose	Validate that an error is returned by a Trajectory query when the coords query parameter is a LINESTRINGZ or MULTILINESTRINGZ coordinate and the z query parameter is specified
Requirement	/req/edr/rc-trajectory
Test Method	<ol style="list-style-type: none"> 1. Check coords query parameter that the system throws an error when a vertical level is specified in both the coords and z parameters 2. Validate that a document was returned with a status code 400.

Table B.106 – Abstract Test 94

Abstract Test 94	/conf/trajectory
Test Purpose	Validate that an error is returned by a Trajectory query when the coords query parameter is a LINESTRINGZM or MULTILINESTRINGZM coordinate and the z query parameter is specified
Requirement	/req/edr/rc-trajectory
Test Method	<ol style="list-style-type: none"> 1. Check coords query parameter that the system throws an error when a vertical level is specified in both the coords and z parameters 2. Validate that a document was returned with a status code 400.

Table B.107 – Abstract Test 95

Abstract Test 95	/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 or MULTILINESTRINGZM Well Known Text value.
Requirement	/req/edr/rc-trajectory
Test Method	<ol style="list-style-type: none"> 1. Check coords query parameter with time parameter is a valid Well Known Text LINESTRINGZM or MULTILINESTRINGZM value, the Z

	coordinate must be a within the range of vertical levels advertised in the Collection metadata
	2. Validate that a document was returned with a status code 400.

Table B.108 – Abstract Test 96

Abstract Test 96	/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 or MULTILINESTRINGZ Well Known Text value.
Requirement	/req/edr/rc-trajectory
Test Method	<ol style="list-style-type: none"> 1. Check coords query parameter with time parameter is a valid Well Known Text LINESTRINGZ or MULTILINESTRINGZ value, the Z coordinate must be a within the range of vertical levels advertised in the Collection metadata 2. Validate that a document was returned with a status code 400.

Table B.109 – Abstract Test 97

Abstract Test 97	/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	<ol style="list-style-type: none"> 1. If time values are specified in the coords query parameter check that they are within the range of time values defined in the Collection metadata. 2. Validate that a document was returned with a status code 400.

Table B.110 – Abstract Test 98

Abstract Test 98	/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	<ol style="list-style-type: none"> 1. Test with valid query parameters 2. Validate that a document was returned with a status code 200. <p>Repeat these tests using the following parameter tests: Coordinates</p>

- 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.111 – Abstract Test 99

Abstract Test 99	/conf/edr/rc-coords-definition
Test Purpose	Validate that the coords query parameters are constructed correctly.
Requirement	/req/edr/coords-definition
Test Method	<p>Verify that the coords query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: coords in: query required: true schema: type: string style: form explode: false </pre> <p>Use a coords value in all requests:</p> <ul style="list-style-type: none"> • A valid Well-Known Text (WKT) representation of geometry string

Table B.112 – Abstract Test 100

Abstract Test 100	/conf/edr/rc-coords-response
Test Purpose	Validate that the coords query parameters are processed correctly.
Requirement	/req/edr/coords-response

Test Method	1. Verify that only resources that have a spatial geometry that intersects the coordinates are returned as part of the result set.
Test Method	1. Verify coords values are valid for the specified coordinate reference system 2. Verify that the coordinate reference system of the geometries is 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.113 – Abstract Test 101

Abstract Test 101	/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): <pre>name: parameter-name in: query required: false schema: type: string style: form explode: false</pre>

Table B.114 – Abstract Test 102

Abstract Test 102	/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	1. Verify that only resources for the requested parameters were included in the result set 2. Validate that the parameter-name parameter complies with the syntax described in /req/edr/parameter-name-response.

Table B.115 – Abstract Test 103

Abstract Test 103	/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	<p>Verify that the crs query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: crs in: query required: false schema: type: string style: form explode: false </pre>

Table B.116 – Abstract Test 104

Abstract Test 104	/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	<ol style="list-style-type: none"> 1. Verify that the geometry of the resources returned are valid for the requested coordinate reference system
Test Method	<ol style="list-style-type: none"> 1. Verify that all crs values defined in the collections metadata are supported by the collection
Test Method	<ol style="list-style-type: none"> 1. Verify that all crs values not defined in the collections metadata will generate a HTTP 400 error 2. Validate that the crs parameter complies with the syntax described in /req/edr/REQ_rc-crs-response.

Table B.117 – Abstract Test 105

Abstract Test 105	/conf/edr/rc-f-definition
Test Purpose	Validate that the f query parameter is constructed correctly.
Requirement	/req/edr/rc-f-definition
Test Method	<p>Verify that the f query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: f in: query required: false schema: type: string style: form explode: false </pre>

Table B.118 – Abstract Test 106

Abstract Test 106	/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	1. Verify that the response is returned in the requested data format
Test Method	1. Verify that all output format values defined in the collections metadata are supported by the collection 2. Validate that the f parameter complies with the syntax described in /req/edr/REQ_rc-f-response.

B.10.1.5. Corridor

Table B.119 – Abstract Test 107

Abstract Test 107	/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	1. No query parameters are specified 2. 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 not specified.
Requirement	/req/edr/rc-corridor
Test Method	1. coords query parameter is not specified 2. 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 corridor-width query parameter is not specified.
Requirement	/req/edr/rc-corridor
Test Method	<ol style="list-style-type: none"> 1. corridor-width query parameter is not specified 2. 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 corridor-height query parameter is not specified.
Requirement	/req/edr/rc-corridor
Test Method	<ol style="list-style-type: none"> 1. corridor-height query parameter is not specified 2. 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 width-units query parameter is not specified.
Requirement	/req/edr/rc-corridor
Test Method	<ol style="list-style-type: none"> 1. width-units query parameter is not specified 2. 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 height-units query parameter is not specified.
Requirement	/req/edr/rc-corridor
Test Method	<ol style="list-style-type: none"> 1. height-units query parameter is not specified 2. 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 <code>corridor</code> query when the <code>coords</code> query parameter does not contain a valid <code>LINESTRING</code> or <code>MULTILINESTRING</code> Well Known Text value.
Requirement	<code>/req/edr/rc-corridor</code>
Test Method	<ol style="list-style-type: none"> 1. Check <code>coords</code> query parameter is a valid Well Known Text <code>LineString</code> or <code>MultiLineString</code> value 2. 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 <code>corridor</code> query when the <code>coords</code> query parameter does not contain a valid <code>LINESTRINGM</code> or <code>MULTILINESTRINGM</code> Well Known Text value.
Requirement	<code>/req/edr/rc-corridor</code>
Test Method	<ol style="list-style-type: none"> 1. Check <code>coords</code> query parameter with <code>time</code> parameter is a valid Well Known Text <code>LINESTRINGM</code> or <code>MULTILINESTRINGM</code> value, the <code>M</code> coordinate must be a valid Epoch value (as known as UNIX time) 2. 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 an error is returned by a <code>corridor</code> query when the <code>coords</code> query parameter is a <code>LINESTRINGZ</code> or <code>MULTILINESTRINGZ</code> coordinate and the <code>z</code> query parameter is specified
Requirement	<code>/req/edr/rc-corridor</code>
Test Method	<ol style="list-style-type: none"> 1. Check <code>coords</code> query parameter that the system throws an error when a vertical level is specified in both the <code>coords</code> and <code>z</code> parameters 2. Validate that a document was returned with a status code 400.

Table B.128 – Abstract Test 116

Abstract Test 116	/conf/corridor
--------------------------	-----------------------

Test Purpose	Validate that an error is returned by a <code>corridor</code> query when the <code>coords</code> query parameter is a valid <code>LINESTRINGZM</code> or <code>MULTILINESTRINGZM</code> coordinate and the <code>z</code> query parameter is specified
Requirement	<code>/req/edr/rc-corridor</code>
Test Method	<ol style="list-style-type: none"> 1. Check <code>coords</code> query parameter that the system throws an error when a vertical level is specified in both the <code>coords</code> and <code>z</code> parameters 2. Validate that a document was returned with a status code 400.

Table B.129 – Abstract Test 117

Abstract Test 117	<code>/conf/corridor</code>
Test Purpose	Validate that an error is returned by a <code>corridor</code> query when the <code>coords</code> query parameter does not contain a valid <code>LINESTRINGZM</code> or <code>MULTILINESTRINGZM</code> Well Known Text value.
Requirement	<code>/req/edr/rc-corridor</code>
Test Method	<ol style="list-style-type: none"> 1. Check <code>coords</code> query parameter with time parameter is a valid Well Known Text <code>LINESTRINGZM</code> or <code>MULTILINESTRINGZM</code> value, the <code>Z</code> coordinate must be a within the range of vertical levels advertised in the Collection metadata 2. Validate that a document was returned with a status code 400.

Table B.130 – Abstract Test 118

Abstract Test 118	<code>/conf/corridor</code>
Test Purpose	Validate that an error is returned by a <code>corridor</code> query when the <code>coords</code> query parameter does not contain a valid <code>LINESTRINGZ</code> or <code>MULTILINESTRINGZ</code> Well Known Text value.
Requirement	<code>/req/edr/rc-corridor</code>
Test Method	<ol style="list-style-type: none"> 1. Check <code>coords</code> query parameter with time parameter is a valid Well Known Text <code>LINESTRINGZ</code> or <code>MULTILINESTRINGZ</code> value, the <code>Z</code> coordinate must be a within the range of vertical levels advertised in the Collection metadata 2. Validate that a document was returned with a status code 400.

Table B.131 – Abstract Test 119

Abstract Test 119	<code>/conf/corridor</code>
Test Purpose	Validate that an error is returned by a <code>corridor</code> query when the <code>coords</code> query parameter contains invalid time coordinates

Requirement	/req/edr/rc-corridor
Test Method	<ol style="list-style-type: none"> 1. 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 2. Validate that a document was returned with a status code 400.

Table B.132 – Abstract Test 120

Abstract Test 120	/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	<ol style="list-style-type: none"> 1. Specify a width-units value that is not listed in the collection response 2. Validate that a document was returned with a status code 400.

Table B.133 – Abstract Test 121

Abstract Test 121	/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	<ol style="list-style-type: none"> 1. Specify a height-units value that is not listed in the collection response 2. Validate that a document was returned with a status code 400.

Table B.134 – Abstract Test 122

Abstract Test 122	/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	<ol style="list-style-type: none"> 1. Test with valid query parameters 2. Validate that a document was returned with a status code 200. <p>Repeat these tests using the following parameter tests:</p> <p>Coordinates</p> <ul style="list-style-type: none"> • Parameter /req/edr/coords-definition • Response /req/edr/coords-response

<p>VerticalLevel</p> <ul style="list-style-type: none"> • Parameter /req/edr/z-definition • Response /req/edr/z-response <p>Parameters</p> <ul style="list-style-type: none"> • Parameter /req/edr/REQ_rc-parameter-name-definition • Response /req/edr/parameter-name-response <p>DateTime</p> <ul style="list-style-type: none"> • Parameter /req/core/datetime-definition • Response /req/core/datetime-response <p>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.</p>

Table B.135 – Abstract Test 123

Abstract Test 123	/conf/edr/rc-coords-definition
Test Purpose	Validate that the coords query parameters are constructed correctly.
Requirement	/req/edr/coords-definition
Test Method	<p>Verify that the coords query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: coords in: query required: true schema: type: string style: form explode: false </pre> <p>Use a coords value in all requests:</p> <ul style="list-style-type: none"> • A valid Well-Known Text (WKT) representation of geometry string

Table B.136 – Abstract Test 124

Abstract Test 124	/conf/edr/rc-coords-response
Test Purpose	Validate that the coords query parameters are processed correctly.
Requirement	/req/edr/coords-response
Test Method	1. Verify that only resources that have a spatial geometry that intersects the coordinates are returned as part of the result set.
Test Method	1. Verify coords values are valid for the specified coordinate reference system

2. Verify that the coordinate reference system of the geometries is 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.137 – Abstract Test 125

Abstract Test 125	<code>/conf/edr/REQ_rc-corridor-width-definition</code>
Test Purpose	Validate that the <code>corridor-width</code> query parameter is constructed correctly.
Requirement	<code>/req/edr/corridor-width-definition</code>
Test Method	Verify that the <code>corridor-width</code> query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment): <pre>name: corridor-width in: query required: true schema: type: string style: form explode: false</pre>

Table B.138 – Abstract Test 126

Abstract Test 126	<code>/conf/collections/REQ_rc-corridor-width-response</code>
Test Purpose	Validate that the <code>corridor-width</code> query parameters are processed correctly.
Requirement	<code>/req/edr/REQ_rc-corridor-width-response</code>
Test Method	<ol style="list-style-type: none"> 1. Verify that a 400 error will be generated if <code>corridor-width</code> is not specified 2. Validate that the <code>corridor-width</code> parameter complies with the syntax described in <code>/req/edr/REQ_rc-corridor-width-response</code>.

Table B.139 – Abstract Test 127

Abstract Test 127	<code>/conf/edr/REQ_rc-corridor-height-definition</code>
Test Purpose	Validate that the <code>corridor-height</code> query parameter is constructed correctly.
Requirement	<code>/req/edr/REQ_rc-corridor-height-definition</code>

Test Method	<p>Verify that the <code>corridor-height</code> query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: corridor-height in: query required: true schema: type: string style: form explode: false </pre>
-------------	---

Table B.140 – Abstract Test 128

Abstract Test 128	<code>/conf/collections/REQ_rc-corridor-height-response</code>
Test Purpose	Validate that the <code>corridor-height</code> query parameters are processed correctly.
Requirement	<code>/req/edr/REQ_rc-corridor-height-response</code>
Test Method	<ol style="list-style-type: none"> 1. Verify that a 400 error will be generated if <code>corridor-height</code> is not specified 2. Validate that the <code>corridor-height</code> parameter complies with the syntax described in <code>/req/edr/REQ_rc-corridor-height-response</code>.

Table B.141 – Abstract Test 129

Abstract Test 129	<code>/conf/edr/REQ_rc-width-units-definition</code>
Test Purpose	Validate that the <code>width-units</code> query parameter is constructed correctly.
Requirement	<code>/req/edr/REQ_rc-width-units-definition</code>
Test Method	<p>Verify that the <code>width-units</code> query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: width-units in: query required: true schema: type: string style: form explode: false </pre>

Table B.142 – Abstract Test 130

Abstract Test 130	<code>/conf/collections/REQ_rc-width-units-response</code>
Test Purpose	Validate that the <code>width-units</code> query parameters are processed correctly.

Requirement	/req/edr/width-units-response
Test Method	<ol style="list-style-type: none"> 1. Verify that units not listed in the metadata will generate an error message 2. Validate that the width-units parameter complies with the syntax described in /req/edr/width-units-response.

Table B.143 – Abstract Test 131

Abstract Test 131	/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	<p>Verify that the within-units query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: height-units in: query required: true schema: type: string style: form explode: false </pre>

Table B.144 – Abstract Test 132

Abstract Test 132	/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	<ol style="list-style-type: none"> 1. Verify that height units not listed in the metadata will generate an error message 2. Validate that the height-units parameter complies with the syntax described in /req/edr/height-units-response.

Table B.145 – Abstract Test 133

Abstract Test 133	/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	<p>Verify that the parameter-name query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: parameter-name in: query required: false schema: type: string style: form explode: false </pre>
-------------	---

Table B.146 – Abstract Test 134

Abstract Test 134	/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	<ol style="list-style-type: none"> 1. Verify that only resources for the requested parameters were included in the result set 2. Validate that the parameter-name parameter complies with the syntax described in /req/edr/parameter-name-response.

Table B.147 – Abstract Test 135

Abstract Test 135	/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	<p>Verify that the crs query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: crs in: query required: false schema: type: string style: form explode: false </pre>

Table B.148 – Abstract Test 136

Abstract Test 136	/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	1. Verify that the geometry of the resources returned are valid for the requested coordinate reference system
Test Method	1. Verify that all crs values defined in the collections metadata are supported by the collection
Test Method	1. Verify that all crs values not defined in the collections metadata will generate a HTTP 400 error 2. Validate that the crs parameter complies with the syntax described in /req/edr/REQ_rc-crs-response.

Table B.149 – Abstract Test 137

Abstract Test 137	/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): <pre> name: f in: query required: false schema: type: string style: form explode: false </pre>

Table B.150 – Abstract Test 138

Abstract Test 138	/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	1. Verify that the response is returned in the requested data format
Test Method	1. Verify that all output format values defined in the collections metadata are supported by the collection 2. 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.151 – Abstract Test 139

Abstract Test 139	/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	<ol style="list-style-type: none"> 1. Issue an HTTP GET request to the URL {root}/collections/{collectionId}/instances 2. Validate that a document was returned with a status code 200 3. Validate the contents of the returned document using test /conf/instances_rc-md-success.

Table B.152 – Abstract Test 140

Abstract Test 140	/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	<ol style="list-style-type: none"> 1. Validate that all response documents comply with /req/core/rc-collection-info-links 2. 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” 3. 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.153 – Schema and Tests for Collections content

FORMAT	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.154 – Abstract Test 141

Abstract Test 141	/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	<ol style="list-style-type: none"> 1. 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. 2. Validate that an Instance of a Collection was returned with a status code 200 3. Validate the contents of the returned document using test /conf/instances/src-md-success.

Table B.155 – Abstract Test 142

Abstract Test 142	/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.156 – Abstract Test 143

Abstract Test 143	/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	<ol style="list-style-type: none"> 1. No query parameters are specified 2. 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.157 – Abstract Test 144

Abstract Test 144	/conf/locations
Test Purpose	Validate that an error is returned by a Locations query when the locationId is invalid.
Requirement	/req/edr/rc-locations
Test Method	<ol style="list-style-type: none"> 1. Check that invalid locationId values return an error message 2. Validate that a document was returned with a status code 404.

Table B.158 – Abstract Test 145

Abstract Test 145	/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	<ol style="list-style-type: none"> 1. Test with valid query parameters 2. Validate that a document was returned with a status code 200. <p>Repeat these tests using the following parameter tests:</p> <p>Parameters</p> <ul style="list-style-type: none"> • Parameter /req/edr/REQ_rc-parameter-name-definition • Response /req/edr/parameter-name-response <p>DateTime</p> <ul style="list-style-type: none"> • Parameter /req/core/datetime-definition • Response /req/core/datetime-response <p>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.</p>

Table B.159 – Abstract Test 146

Abstract Test 146	/conf/core/datetime-definition
Test Purpose	Validate that the datetime query parameters are constructed correctly.
Requirement	/req/core/datetime-definition

Test Method	<p>Verify that the <code>datetime</code> query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: datetime in: query required: false schema: type: string style: form explode: false </pre>
-------------	--

Table B.160 – Abstract Test 147

Abstract Test 147	<code>/conf/core/datetime-response</code>
Test Purpose	Validate that the <code>datetime</code> query parameters are processed correctly.
Requirement	<code>/req/core/datetime-response</code>
Test Method	<ol style="list-style-type: none"> 1. Verify that only resources that have a temporal geometry that intersects the temporal information in the <code>datetime</code> parameter were included in the result set. 2. Verify that all resources in the collection that are not associated with a temporal geometry are included in the result set. 3. Validate that the <code>datetime</code> parameter complies with the syntax described in <code>/req/core/datetime-response</code>.

Table B.161 – Abstract Test 148

Abstract Test 148	<code>/conf/collections/REQ_rc-parameter-name-definition</code>
Test Purpose	Validate that the <code>parameter-name</code> query parameters are constructed correctly.
Requirement	<code>/req/edr/REQ_rc-parameter-name-definition</code>
Test Method	<p>Verify that the <code>parameter-name</code> query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: parameter-name in: query required: false schema: type: string style: form explode: false </pre>

Table B.162 – Abstract Test 149

Abstract Test 149	<code>/conf/edr/rc-parameter-name-response</code>
-------------------	---

Test Purpose	Validate that the parameter-name query parameters are processed correctly.
Requirement	/req/edr/parameter-name-response
Test Method	<ol style="list-style-type: none"> 1. Verify that only resources for the requested parameters were included in the result set 2. Validate that the parameter-name parameter complies with the syntax described in /req/edr/parameter-name-response.

Table B.163 – Abstract Test 150

Abstract Test 150	/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	<p>Verify that the crs query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: crs in: query required: false schema: type: string style: form explode: false </pre>

Table B.164 – Abstract Test 151

Abstract Test 151	/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	<ol style="list-style-type: none"> 1. Verify that the geometry of the resources returned are valid for the requested coordinate reference system
Test Method	<ol style="list-style-type: none"> 1. Verify that all crs values defined in the collections metadata are supported by the collection
Test Method	<ol style="list-style-type: none"> 1. Verify that all crs values not defined in the collections metadata will generate a HTTP 400 error 2. Validate that the crs parameter complies with the syntax described in /req/edr/REQ_rc-crs-response.

Table B.165 – Abstract Test 152

Abstract Test 152	/conf/edr/rc-f-definition
Test Purpose	Validate that the f query parameter is constructed correctly.
Requirement	/req/edr/rc-f-definition
Test Method	<p>Verify that the f query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):</p> <pre> name: f in: query required: false schema: type: string style: form explode: false </pre>

Table B.166 – Abstract Test 153

Abstract Test 153	/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	<ol style="list-style-type: none"> 1. Verify that the response is returned in the requested data format
Test Method	<ol style="list-style-type: none"> 1. Verify that all output format values defined in the collections metadata are supported by the collection 2. Validate that the f parameter complies with the syntax described in /req/edr/REQ_rc-f-response.



ANNEX C (INFORMATIVE) COLLECTION RESPONSE METADATA (INFORMATIVE)



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	<p>A bounding box is provided as four numbers:</p> <ul style="list-style-type: none"> • 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	<p>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".</p> <p>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.</p>

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	Array of ISO 8601 Date Array	Yes	An array of ISO 8601 Date Array, each ISO8601 Date Array should contain two values first being the minimum date time and second the maximum date time for information in the collection (see https://en.wikipedia.org/wiki/ISO_8601)
values	ISO 8601 Date Array	Yes	An array of ISO 8601 datestrings which details the time intervals available in the collection, each member of 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 <u>Well Known Text</u> 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 level values array, each Level value Array should contain two values first being the minimum vertical level and second the maximum vertical level for information in the collection
values	String Array	Yes	Array of height values supported by the collection.
vrs	String	Yes	Follows the conventions defined by the <u>Well Known Text</u> 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”: [[“2021-04-22T00:00:00Z”,“2021-05-03T12:00:00Z”]],
    “values”: [“R82/2021-04-22T00:00:00Z/PT3H”,
    “R2/2021-05-02T12:00:00Z/PT12H”],
    “trs”: “TIMECRS[\"DateTime\",TDATUM[\"Gregorian Calendar\"],
    CS[TemporalDateTime,1],AXIS[\"Time (T)\",future]]”
  },
  “vertical”: {
    “interval”: [[“1829.0”,“3658.0”]],
    “values”: [“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
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/sampleddata/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/sampleddata/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 Dictionary (Python) or HashMap(Java).

C.1.11. Parameter Object

Table C.11 – Parameter Object

FIELD NAME	TYPE	REQUIRED	DESCRIPTION
id	String	No	parameter id
type	String	Yes	Always 'Parameter'
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
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	<p>Duration of calculation. For time durations, this follows the ISO8601 Duration standard.</p> <ul style="list-style-type: none"> • A negative sign before a duration value (i.e. -PT10M) infers that the time start starts at the specified 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 measurementType 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"
    }
  }
}

```




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-desc", "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 resources 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 (link relation type “license”) and also the terms and conditions of service (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": "hrly_obs",
      "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.example.org/services/data/
terms-and-conditions---datapoint#termsofservice",
    "hreflang": "en",
    "rel": "restrictions",
    "type": "text/html"
  }
],
"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"]
    ],
    "values": [
      "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": [
  "http://www.opengis.net/def/crs/OGC/1.3/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/ObservedVariableAtmosp
here/_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": "3_hrly_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",
        "Probability 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"]
        ],
        "values": [
            "2020-06-23T18:00:00Z/2020-07-04T21:00:00Z"
        ],
        "trs": "TIMECRS[\"DateTime\", TDATUM[\"Gregorian Calendar
\"], CS[TemporalDateTime, 1], AXIS[\"Time (T)\", future]]"
    }
},
"data_queries" : {
    "position": {
        "link": {
            "href": "http://www.example.org/edr/collections/3_hrly_
forecast/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": "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/3_hrly_
forecast/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"
],
"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/3_hrly_
forecast/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"
            ],
            "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\"]"
                }
            ]
        }
    },
    "instances": {
        "link": {
            "href": "http://www.example.org/edr/collections/3_hrly_
forecast/instances",
            "hreflang": "en",
            "rel": "data",
            "templated": false,
            "variables": {
                "title": "Instances query",
                "description": "Instances query",
                "query_type": "instances"
            }
        }
    },
    "crs": [
        "http://www.opengis.net/def/crs/OGC/1.3/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/ObservedVariableAtmosp
here/_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"
}
},
"Probability 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": "Probability 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 ([link relation type](#) “license”) and also the terms and conditions of service ([link relation type](#) “restrictions”).

```

{
  "links": [
    {
      "href": "http://www.example.org/edr/collections/3_hrly_fcst/
instances/",
      "hreflang": "en",
      "rel": "self",
      "type": "application/json"
    },
    {
      "href": "http://www.example.org/edr/collections/3_hrly_fcst/
instances/?f=html",
      "hreflang": "en",
      "rel": "alternate",
      "type": "text/html"
    },
    {
      "href": "http://www.example.org/edr/collections/3_hrly_fcst/
instances/?f=xml",
      "hreflang": "en",
      "rel": "alternate",
      "type": "application/xml"
    }
  ]
}

```

```

    },
    {
      "href": "http://www.example.org/terms-and-conditions---
datapoint#termsofservice",
      "hreflang": "en",
      "rel": "restrictions",
      "type": "text/html",
      "title": ""
    },
    {
      "href": "http://www.example.org/terms-and-conditions---
datapoint#datalicence",
      "hreflang": "en",
      "rel": "license",
      "type": "text/html",
      "title": ""
    },
    {
      "href": "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://www.example.org/edr/collections/3_hrly_
fcst/instances/2020-06-30T10:00:00Z",
          "hreflang": "en",
          "rel": "self",
          "type": "application/json"
        },
        {
          "href": "http://www.example.org/edr/collections/3_hrly_
fcst/instances/2020-06-30T10:00:00Z?f=html",
          "hreflang": "en",
          "rel": "alternate",
          "type": "text/html"
        },
        {
          "href": "http://www.example.org/edr/collections/3_hrly_
fcst/instances/2020-06-30T10:00:00Z?f=xml",
          "hreflang": "en",

```



```

    },
    "radius": {
      "link": {
        "href": "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://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://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": [
  "http://www.opengis.net/def/crs/OGC/1.3/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://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/ObservedVariableAtmosp
here/_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"
        }
    },
    "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://www.example.org/edr/collections/3_hrly_
fcst/instances/2020-06-30T09:00:00Z",
            "hreflang": "en",
            "rel": "self",
            "type": "application/json"
        },
        {
            "href": "http://www.example.org/edr/collections/3_hrly_
fcst/instances/2020-06-30T09:00:00Z?f=html",
            "hreflang": "en",
            "rel": "alternate",
            "type": "text/html"
        },
        {
            "href": "http://www.example.org/edr/collections/3_hrly_
fcst/instances/2020-06-30T09:00:00Z?f=xml",
            "hreflang": "en",
            "rel": "alternate",
            "type": "application/xml"
        }
    ],
    "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"]
    ],
    "values": [
        "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://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://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://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://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": [
        "http://www.opengis.net/def/crs/OGC/1.3/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://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/ObservedVariableAtmosp
here/_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"
}
}
}
},
{
  "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://www.example.org/edr/collections/3_hrly_
fcst/instances/2020-06-30T08:00:00Z",
            "hreflang": "en",
            "rel": "self",
            "type": "application/json"
        },
        {
            "href": "http://www.example.org/edr/collections/3_hrly_
fcst/instances/2020-06-30T08:00:00Z?f=html",
            "hreflang": "en",
            "rel": "alternate",
            "type": "text/html"
        },
        {
            "href": "http://www.example.org/edr/collections/3_hrly_
fcst/instances/2020-06-30T08:00:00Z?f=xml",
            "hreflang": "en",
            "rel": "alternate",
            "type": "application/xml"
        }
    ],
    "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"]
            ],
            "values": [
                "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://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://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://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://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": [
  "http://www.opengis.net/def/crs/OGC/1.3/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://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/ObservedVariableAtmosp
here/_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"
  }
}
},
{
  "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://www.example.org/edr/collections/3_hrly_
fcst/instances/2020-06-30T07:00:00Z",
      "hreflang": "en",
      "rel": "self",
      "type": "application/json"
    },
    {
      "href": "http://www.example.org/edr/collections/3_hrly_
fcst/instances/2020-06-30T07:00:00Z?f=html",
      "hreflang": "en",
      "rel": "alternate",
      "type": "text/html"
    }
  ]
}
}
}

```

```

        "href": "http://www.example.org/edr/collections/3_hrly_
fcst/instances/2020-06-30T07:00:00Z?f=xml",
        "hreflang": "en",
        "rel": "alternate",
        "type": "application/xml"
    },
    ],
    "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"]
            ],
            "values": [
                "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://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://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://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://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": [
    "http://www.opengis.net/def/crs/OGC/1.3/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://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/ObservedVariableAtmosp
here/_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": {

```



```

        -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"  
  }  
]  
}
```



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.



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)



ANNEX G (INFORMATIVE) REVISION HISTORY



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
2021-04-20	Master branch	Mark Burgoyne	all	Finalize collection response metadata

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
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
2022-05-12	Master branch	Mark Burgoyne	all	Add missing requirements and tests for data_queries metadata
2022-06-01	1.0.1	G.Hobona	all	Final OGC Staff review



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>