OGC API - Environmental Data Retrieval Standard - Part 1 - Core

Table of Contents

Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: 2020-03-11

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

Internal reference number of this OGC® document: 19-086

Version: 0.0.4

Category: OGC® Implementation Specification

Editor: Mark Burgoyne

OGC API - Environmental Data Retrieval Standard - Part 1 - Core

Warning

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.

Document type: OGC®ImplementationSpecification
Document stage: Draft
Document language: English

License Agreement

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

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

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

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

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

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize

you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

Chapter 1. Introduction

i. Abstract

The OGC has extended their suite of standards to include Resource Oriented Architectures and Web APIs, these standards are based on a common foundation upon which all OGC APIs will be built. This document extends that foundation to define the Environmental Data Retrevial (EDR) API.

An Environmental Data Retrevial (EDR) API provides a family of lightweight interfaces to access Environmental Data resources. Each resource addressed by an EDR API maps to a defined query pattern. These patterns are described in the Requirement Class Core section.

The API-Common standard defines resources and access paths that are supported by all OGC APIs. These are listed in Table 1.

Table 1. Overview of Resources

Resource	Path	HTTP Method	Document Reference
Landing page	`/`	GET	API Landing Page
API definition	`/api`	GET	API Definition
Conformance classes	`/conformance`	GET	Declaration of Conformance Classes
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 the interrogate the API to determine its capabilities and retrieve information (metadata) about this distribution of the resource. This includes the API definition of the server(s) as well as metadata about the resources provided by those servers.

This standard extends the common Query operations for OGC APIs. Query operations allow resources or values extracted from those resources to be retrieved from the underlying data store. The information to be returned is based upon selection criteria (query string) provided by the client. This standard only defines simple query parameters which should be applicable to all resource types. Other OGC API standards may define additional query capabilities specific to their resource type.

ii. Keywords

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

ogcdoc, OGC document, property, geographic information, spatial data, spatial things, dataset, distribution, API, geojson, covJSON,, html, OpenAPI, AsyncAPI, REST, Common

iii. Preface

OGC Declaration

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 Inc. shall not be held responsible for identifying any or all such patent rights.

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

iv. Submitting organizations

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

- Met Office
- others TBD

v. Submitters

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

Name	**Affiliation**
Mark Burgoyne *(editor)*	Met Office
others	TBD

Chapter 2. Scope

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

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 this distribution of a resource. This includes the API definition of the server as well as metadata about the Environmental Data resources provided by the server.

Query operations allow Environmental Data resources to be retrieved from the underlying data store based upon simple selection criteria, defined by this standard and selected by the client.

Chapter 3. Conformance

Conformance with this standard shall be checked using the tests specified in Annex A (normative) 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 provides a common foundation for OGC API standards. Therefore, this standard should be viewed as an extension to API-Common. Conformance to this standard requires demonstrated conformance to the applicable Conformance Classes of API-Common.

This standard identifies six (6) 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 Requirements Class. The tests in Annex A are organized by Requirements Class. The Requirements Classes define the functional requirements which will be tested through the associated Conformance Class.

The requirements classes for OGC API-EDR are:

- Core

The *Core Requirements Class* is the minimal useful service interface for an OGC API. The requirements specified in this Requirements Class are mandatory for all implementations of API-EDR.

- Collections

NOTE	Validate that this description is accurate.

The *Collections Requirements Class* defines the requirements specific to the query and retrieval of Environmental Data Resources.

- Encodings
  - HTML
  - GeoJSON
  - CovJSON

Neither the nor *Core* nor *Collections* requirements class mandate a specific encoding or format for representing resources. The *HTML*, *GeoJSON* and *CovJSON* requirements classes specify representations for these resources in commonly used encodings for spatial data on the web.

None of these encodings are mandatory. An implementation of the *API-EDR* standard may decide to implement another encoding instead of, or in addition those listed. However, a common format does simplifies interoperability so support for *CovJSON* is highly recommended.

- OpenAPI 3.0

The *OGC API-Common* standard does not mandate any encoding or format for the formal definition of the API. However, the prefered option is the OpenAPI 3.0 specification. Therefore the EDR API's will be defined using *OpenAPI 3.0*.

Chapter 4. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to 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
- 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
- Herring, J.: **Simple Feature Access - Part 1: Common Architecture**, http://portal.opengeospatial.org/files/?artifact_id=25355
- Lott, R.: **Well-Known Text representation of Coordinate Reference Systems**, http://docs.opengeospatial.org/is/18-010r7/18-010r7.html

Chapter 5. Terms and Definitions

This document uses the terms defined in Sub-clause 5 of OGC API-Common Part 1 (OGC 19-072), which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

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

5.1. Conformance Module; Conformance Test Module

set of related tests, all within a single conformance test class (OGC 08-131)

```
NOTE: 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.
```

5.2. Conformance Class; Conformance Test Class

set of conformance test modules that must be applied to receive a single certificate of conformance (OGC 08-131)

```
NOTE: When no ambiguity is possible, the word _test_ may be left out, so conformance
test class maybe called a conformance class.
```

5.3. dataset

collection of data, published or curated by a single agent, and available for access or download in one or more formats (DCAT)

5.4. Distribution

represents an accessible form of a **dataset** (DCAT)

EXAMPLE: a downloadable file, an RSS feed or a web service that provides the data.

5.5. Executable Test Suite (ETS)

A set of code (e.g. Java and CTL) 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)

5.6. 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 08-131)

5.7. 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 08-131)

5.8. Requirements Class

aggregate of all requirement modules that must all be satisfied to satisfy a conformance test class (OGC 08-131)

5.9. Requirements Module

aggregate of requirements and recommendations of a specification against a single standardization target type (OGC 08-131)

5.10. Standardization Target

entity to which some requirements of a standard apply (OGC 08-131)

```
NOTE: The standardization target is the entity which may receive a certificate of
conformance for a requirements class.
```

Chapter 6. Conventions

6.1. Identifiers

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.

6.2. Link relations

To express relationships between resources, RFC 8288 (Web Linking) and registered link relation types are used.

6.3. Examples

Most of the examples provided in this standard are encoded in JSON. JSON was chosen because it is widely understood by implementers 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.

6.4. Schema

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

6.5. Use of HTTPS

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

6.6. API definition

6.6.1. General remarks

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

This standard specifies requirements and recommendations for APIs that share spatial 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.

6.6.2. 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.

6.6.3. References to OpenAPI components in normative statements

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

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

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

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

6.6.4. 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 1. 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.

6.6.5. Reusable OpenAPI components

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

CAUTION	During the development phase, these components use a base URL of "https://github.com/opengeospatial/Environmental-Data-Retrieval-API/master", but eventually they are expected to be available under the base URL "http://schemas.opengis.net/ogcapi-erd-1/1.0".

Chapter 7. Overview

7.1. General

The OGC API family of standards enable access to resources using the HTTP protocol and its' associated operations (GET, PUT, POST, etc.). OGC API-Common defines a set of features which are applicable to all OGC APIs. Other OGC standards extend API-Common with features specific to a resource type. This OGC API-EDR standard defines an API with two goals:

1. To allow service users to retrieve, in effect, transient resources over a discrete sampling geometry, created by the service in response to a standardized query pattern.
2. To provide 'building blocks' allowing the construction of more complex services

7.2. Resource Paths

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

NOTE	Add the rest of the API_EDR paths

Table 2. Environmental Data API Paths

Path Template	Relation	Resource
Common		
{root}/	none	Landing page
{root}/api	`service-desc` or `service-doc`	API Description (optional)
{root}/conformance	`conformance`	Conformance Classes
{root}/collections	`data`	Metadata describing the spatial collections available from this API.
{root}/collections/{coverageid}		Metadata describing the collection which has the unique identifier `{collectionid}`

Where:

- {root} = Base URI for the API server
- {collectionid} = an identifier for a specific collection

7.3. Dependencies

The OGC API-EDR standard is an extension of the OGC API-Common standard. Therefore, an implementation of API-EDR must first satisfy the appropriate Requirements Classes from API-Common. Table 3 Identifies the 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 API-EDR Sections to API-Common Requirements Classes

API-EDR Section	API-Common Requirements Class
API Landing Page	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core
API Definition	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core
Declaration of Conformance Classes	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core
Collections	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/collections
OpenAPI 3.0	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/oas30
GeoJSON	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/json
CovJSON	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/json
HTML	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/html

7.4. Overview

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

The `Core` Requirements Class defines the requirements for locating, understanding, and accessing environmental data resources. The `Core` Requirements Class is presented in five sections:

1. API Platform: a set of common capabilities
2. Collection Access: operations for accessing collections of environmental data.
3. Environmental Resources: operations for accessing environmental data resources
4. Query Resources: operations for accessing environmental data resources through queries
5. Parameters: parameters for use in the API-EDR operations.
6. General: general principles for use with this standard.

7.5. Dependencies

The OGC API-EDR standard is an extension of the OGC API-Common standard. Therefore, an implementation of API-EDR must first satisfy the appropriate Requirements Classes from API-Common.

Requirement 1	**/req/core/api-common**
The API implementation SHALL demonstrate conformance with the following Requirements Classes of the OGC API-Common version 1.0 Standard.	

A	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core
B	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/collections

7.6. Platform

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 Environmental Data resources.

7.6.1. API landing page

The landing page provides links to start exploration of the resources offered by an API. Its most important component is a list of links. OGC API-Common already requires some common links. Those links are sufficient for this standard.

Table 4. Dependencies

http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core

Operation

The `Landing Page` operation is defined in the `Core` conformance class of API-Common. No modifications are needed to support `Environmental` 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 API-Common.

Response

A successful response to the `Landing Page` operation is defined in API-Common. The schema for this resource is provided in Landing Page Response Schema.

Landing Page Response Schema

```
type: object
required:
  - links
properties:
  title:
    type: string
    example: Buildings in Bonn
  description:
    type: string
    example: Access to data about buildings in the city of Bonn via a Web API that
conforms to the OGC API Common specification.
  links:
    type: array
    items:
      $ref: link.yaml
```

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

Landing Page Example

```
Unresolved directive in clause_7_core.adoc -
include::examples/JSON/landingPage_example.json[]
```

Error situations

The requirements for handling unsuccessful requests are provided in [http-response]. General guidance on HTTP status codes and how they should be handled is provided in HTTP Status Codes.

7.6.2. 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.

Table 5. Dependencies

http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core

Operation

This operation is defined in the `Core` conformance class of API-Common. No modifications are needed to support `Environmental` 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 API-Common.

Response

A successful response to the API Definition request is a resource which documents the design of the API. API-Common leaves the selection of format for the API Definition response to the API implementor. However, the options are limited to those which have been defined in the API-Common standard. At this time OpenAPI 3.0 is the only option provided.

Error situations

The requirements for handling unsuccessful requests are provided in [http-response]. General guidance on HTTP status codes and how they should be handled is provided in HTTP Status Codes.

7.6.3. Declaration of conformance classes

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

Table 6. Dependencies

http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core

Operation

This operation is defined in the `Core` conformance class of API-Common. No modifications are needed to support `Environmental` 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 API-Common.

Response

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

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

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

Conformance Response Schema

```
type: object
required:
  - conformsTo
properties:
  conformsTo:
    type: array
    items:
      type: string
      example: "http://www.opengis.net/spec/OAPI_Common/1.0/req/core"
```

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

Conformance Information Example

```
Unresolved directive in clause_7_core.adoc -
include::examples/JSON/conformance_example.json[]
```

7.7. Environmental Resources

Environmental Resources can exposed using two path templates:

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

Where

{collectionId} = a unique identifier for a Spatial Resource collection.

{instanceId} = a text string identifying the *'Version'* or *'Instance'* of the choosen collection.

{queryType} = a text string identifying the type of geo-spatial query pattern performed by the API.

Information Resources associated with a specific collection should be accessed through the `/collections` path. Those which are not associated with a specific collection should use the `/{instanceId}/{queryType}` template.

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

The EDR API standard has identified an initial set of common `queryTypes` to implement, this list may change as the standard is used. The Table 4 provides a list of the proposed `queryTypes`.

Table 7. Information Resource Types

Query Type	API Standard
Corridor	/corridor
Cube	/cube
Items	/items
Point	/point
Polygon	/polygon
Trajectory	/trajectory
Instance	/instance

7.8. Query Resources

Query Resources are spatial queries which support the operation of the API or the access and use of the Spatial Resources. They are described in the QueryType specific Operations section.

Query Resources related to Spatial Resources can exposed using the path template:

- /collections/{collectionId}/{queryType}

The resources returned from each node in this template are described in [query-resource-paths].

Table 8. Information Resource Paths

Path Template	Resource
/collections	The root resource describing the spatial collections available from this API.
/collections/{collectionId}	Identifies a collection with the unique identifier `{collectionId}`
/collections/{collectionId}/{queryType}	Identifies an Information Resource of type {resourceType} associated with the `{collectionId}` collection.

The OGC API-Common standard does not define any Information Resource types. However Table 6 provides a mapping of the intial query types proposed for the EDR API.

Table 9. Query Types

Path Template	Query Type	Description
/collections/{collectionId}/position	Position	Return data for the requested point location
/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

Path Template	Query Type	Description
/collections/{collectionId}/corridor	Corridor	Return data within a spatial corridor
/collections/{collectionId}/items	Item	Items associated with the `{collectionId}` collection.
/collections/{collectionId}/locations	Locations	Location identifers associated with the `{collectionId}` collection.
/collections/{collectionId}/instances	Instances	List the available instances of the collection

7.9. Parameter Modules

Query parameters are used in URLs to define the resources which are returned on a GET request. The EDR standard defines the following as standard parameters for use in EDR API standards.

7.9.1. Parameter coords

Requirement 3	/req/edr/coords-definition

A	Each resource collection operation SHALL support a parameter `coords` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: coords`
	`in: query`
	`required: true`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

B	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

Requirement 4	/req/edr/coords-response
A	Only those resources that have a spatial geometry that intersects the area defined by the `coords` parameter SHALL be part of the result set.

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

C	The coordinate reference system of the values SHALL be interpreted as WGS84 longitude/latitude
	
	`WKT: GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]]`
	
	`EPSG: http://www.opengis.net/def/crs/OGC/1.3/CRS84`
	
	unless a different coordinate reference system is specified in a parameter `crs`.

7.9.2. Parameter datetime

Requirement 5	**/req/core/rec-datetime-parameter**
A	An EDR API SHALL support the Date-Time (datetime) parameter for `/collections` and `/collections/{collectionid}` requests.
B	Requests which include the Date-Time parameter SHALL comply with API-Common requirement `/req/core/rc-time-definition`.
C	Responses to Date-Time requests SHALL comply with API-Common requirement `/req/core/rc-time-response`.

The datetime parameter is defined in API-Common. The following information is provided as a convinience to the user.

"Intersects" means that the time (instant or period) specified in the parameter `datetime` includes a timestamp that is part of the temporal geometry of the resource (again, a time instant or period). For time periods this includes the start and end time.

Example 2. A date-time

February 12, 2018, 23:20:52 GMT:

`time=2018-02-12T23%3A20%3A52Z`

For resources with a temporal property that is a timestamp (like `lastUpdate` in the building

features), a date-time value would match all resources where the temporal property is identical.

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

Example 3. Intervals

February 12, 2018, 00:00:00 GMT to March 18, 2018, 12:31:12 GMT:

`datetime=2018-02-12T00%3A00%3A00Z%2F2018-03-18T12%3A31%3A12Z`

February 12, 2018, 00:00:00 UTC or later:

`datetime=2018-02-12T00%3A00%3A00Z%2F..`

March 18, 2018, 12:31:12 UTC or earlier:

`datetime=..%2F2018-03-18T12%3A31%3A12Z`

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

7.9.3. Parameter parametername

Requirement 6	**/req/edr/parameters-definition**

A	Each resource collection operation SHALL support a parameter `parametername`with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: parametername`
	`in: query`
	`required: true`
	`explode: false`
	`schema:`
	`  minItems: 1`
	`  type: array`
	`  items:`
	`  type: string`

Requirement 7	**/req/edr/parameters-response**
A	If the `parametername` parameter is provided, only those parameters named SHALL be returned. If the `parametername` parameter is not specified all parameters in the collection SHALL be returned.

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

Example 4. A single parameter

Only return values for the Maximum_temperature

```
parametername=Maximum_temperature
```

Example 5. Return multiple parameters

Values for the Maximum_temperature, Minimum_temperature and Total_precipitation

```
parametername=Maximum_temperature,Minimum_temperature,Total_precipitation
```

If none of the requested parameters exist in the collection a 403 message SHOULD be returned.

7.9.4. Parameter crs

Requirement 8	**/req/edr/outputCRS-definition**

A	Each resource collection operation SHALL support a parameter `crs` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: crs`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Requirement 9	**/req/edr/outputCRS-response**
A	If the `crs` parameter is provided, the returned information should be reprojected (if required) to the defined coordinate system. If the `crs` parameter is not specifed the data will be returned in its native projection.
B	The `crs` parameter SHALL consist of an identifier selected from the enumerated list of valid values supplied in the collections metadata.
C	if an unsupported `crs` value is requested a `403` error message SHOULD be returned .

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

7.9.5. Parameter outputformat

Requirement 10	**/req/edr/outputFormat-definition**

A	Each resource collection operation SHALL support a parameter `outputformat`with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: outputformat`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Requirement 11	**/req/edr/outputFormat-response**
A	If the `outputformat` parameter is provided, the returned information should be transformed to the defined data format.
B	The `outputformat` parameter SHALL consist of an string value based on an enumerated list of available options provided in the collections metadata.
C	If a unsupported `outputformat` value is requested a `403` error message should be returned.

Example 6. Return data as coverageJSON

```
outputformat=coverageJSON
```

If not specified the query will return data in the native format of the collection. If the requested format system does not match an entry in the defined list of valid outputFormats for the collection a 403 message SHOULD be returned.

7.10. QueryType specific Operations

The following describes the query parameters that are unique or specialised to the individual query types:

NOTE	The following include statements import files which in turn import files already imported above. That results in duplicate section identifiers.

7.10.1. Position

Parameters

Parameter coords

Requirement 12	/req/edr/coords-definition

A	Each resource collection operation SHALL support a parameter `coords` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: coords`
	`in: query`
	`required: true`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

B	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

Requirement 13	/req/edr/coords-response
A	Only those resources that have a spatial geometry that intersects the area defined by the `coords` parameter SHALL be part of the result set.
B	The coordinates SHALL consist of a Well Known Text (WKT) geometry string

C	The coordinate reference system of the values SHALL be interpreted as WGS84 longitude/latitude
	
	`WKT: GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]]`
	
	`EPSG: http://www.opengis.net/def/crs/OGC/1.3/CRS84`
	
	unless a different coordinate reference system is specified in a parameter `crs`.

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

POINT(x y)

A point at height `z` POINT(x y z)

And for a list of locations

MULTIPOINTx y),(x1 y1),(x2 y2),(x3 y3

And for a list of locations at defined heights

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

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

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

Example 7. Point

retrieve data for Greenwich, London

```
coords=POINT(0 51.48)
```

Example 8. Point for a specific height

retrieve data for Mount Everest

```
coords=POINT(86.95 27.99 8848)
```

Example 9. Multiple locations

retrieve data for a list of locations : 38.9N 77W, 48.85N 2.35E, 39.92N 116.38E, 35.29S 149.1E, 51.5N 0.1W

```
coords=MULTIPOINT  38.9 -77),(48.85 2.35),(39.92 116.38),(-35.29 149.1),(51.5 -0.1
```

Example 10. Multiple locations with specified heights

retrieve data for a list of locations : 38.9N 77W, 48.85N 2.35E, 39.92N 116.38E, 35.29S 149.1E, 51.5N 0.1W

```
coords=MULTIPOINT  -77.02  38.94  72.3),(-3.50  50.72  36.35),(1.43  43.63  135.63),(117.23
39.13 10.04),(144.96 -37.80 33.83
```

Parameter interpolation

Requirement 14	**/req/edr/interpolation-definition**

A	Each resource collection operation MAY support a parameter `interpolation` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: interpolation`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Requirement 15	**/req/edr/interpolation-response**
A	If the `interpolation` parameter is defined the result set SHALL to data point that is the best match based on the chosen algorithm.
B	The interpolation information is an interpolation method. The available interpolation methods should be part of the instance metadata: `interpolation = interpolation-method`
C	If not specified the query will use the default interpolation method for the chosen collection.
D	If the interpolation method value e does not match one of the listed methods a 403 error message should be returned.

Example 11. An interpolation

Use the defined nearest neighbour algorithm to get the best data match for the requested locations

`interpolation=nearest-neighbour`

Parameter interpolationz

If the collection supports interpolation of the data to a height not available in the underlying data and vertical interpolation is supported by the collection it will be defined as follows:

Requirement 16	**/req/edr/interpolationZ-definition**

A	Each resource collection operation MAY support a parameter `interpolationz`with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: interpolationz`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Requirement 17	**/req/edr/interpolationZ-response**
A	If the `interpolationz` parameter is defined the result set SHALL contain values derieved based on the chosen interpolation algorithm at the number of specifed z samples.

B	The interpolationz information is an interpolation method and an interval. The available interpolation methods should be part of the collection response and the samples value is a number :
	
	`interpolationz = "R" intervals "/"`
	`intervalinterpolation-method`

C	The interpolation information is an interpolation method. The available interpolation methods should be part of the instance metadata:
D	If not specified the query will use the default interpolation method for the chosen collection.
E	If the vertical coordinates only contain a single height value the R `intervals` MUST always be 1.
F	If the interpolation method value e does not match one of the listed methods a 403 error message should be returned.

Example 12. An interpolationz

Interpolate to the requested vertical height

```
interpolationz=R1/cubic_spline
```

Interpolate 6 values between the min and max heights supplied in the query

```
interpolationz=R6/cubic_spline
```

7.10.2. Radius

Parameters

Parameter coords

Requirement 18	/req/edr/coords-definition

A	Each resource collection operation SHALL support a parameter `coords` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: coords`
	`in: query`
	`required: true`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

B	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

Requirement 19	/req/edr/coords-response
A	Only those resources that have a spatial geometry that intersects the area defined by the `coords` parameter SHALL be part of the result set.
B	The coordinates SHALL consist of a Well Known Text (WKT) geometry string

C	The coordinate reference system of the values SHALL be interpreted as WGS84 longitude/latitude
	
	`WKT: GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]]`
	
	`EPSG: http://www.opengis.net/def/crs/OGC/1.3/CRS84`
	
	unless a different coordinate reference system is specified in a parameter `crs`.

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

POINT(x y)

A point at height `z` POINT(x y z)

And for a list of locations

MULTIPOINTx y),(x1 y1),(x2 y2),(x3 y3

And for a list of locations at defined heights

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

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

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

Example 13. Point

retrieve data for Greenwich, London

```
coords=POINT(0 51.48)
```

Example 14. Point for a specific height

retrieve data for Mount Everest

```
coords=POINT(86.95 27.99 8848)
```

Example 15. Multiple locations

retrieve data for a list of locations : 38.9N 77W, 48.85N 2.35E, 39.92N 116.38E, 35.29S 149.1E, 51.5N 0.1W

```
coords=MULTIPOINT  38.9 -77),(48.85 2.35),(39.92 116.38),(-35.29 149.1),(51.5 -0.1
```

Example 16. Multiple locations with specified heights

retrieve data for a list of locations : 38.9N 77W, 48.85N 2.35E, 39.92N 116.38E, 35.29S 149.1E, 51.5N 0.1W

```
coords=MULTIPOINT  -77.02  38.94  72.3),(-3.50  50.72  36.35),(1.43  43.63  135.63),(117.23
39.13 10.04),(144.96 -37.80 33.83
```

Parameter within

Requirement 20	**/req/edr/within-definition**
A	Each resource collection operation MAY support a parameter `within`with the following characteristics (using an OpenAPI Specification 3.0 fragment):

B	If the instance metadata does not provide `withinUnits` values the API SHALL NOT support `within` queries:
	`name: within`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Requirement 21	**/req/edr/within-response**
A	If the `within` parameter is provided, all selected information within the specified radius SHALL be part of the result set.
B	If a `withinunits` parameter is not provided, a 403 error WILL be returned.

Parameter withinunits

Requirement 22	**/req/edr/withinUnits-definition**
A	Each resource collection operation MAY support a parameter `withinunits`with the following characteristics (using an OpenAPI Specification 3.0 fragment):

B	A withinunits value MUST be one of the values defined in the instance metadata:
	
	`name: withinunits`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Requirement 23	**/req/edr/within-response**
A	The `withinunits` parameter defines the distance units of the `within` query parameter value .

7.10.3. Area

7.10.4. Parameters

7.10.5. Parameter coords

Requirement 24	**/req/edr/coords-definition**

A	Each resource collection operation SHALL support a parameter `coords` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: coords`
	`in: query`
	`required: true`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

B	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

Requirement 25	**/req/edr/coords-response**
A	Only those resources that have a spatial geometry that intersects the area defined by the `coords` parameter SHALL be part of the result set.
B	The coordinates SHALL consist of a Well Known Text (WKT) geometry string

C	The coordinate reference system of the values SHALL be interpreted as WGS84 longitude/latitude
	
	`WKT: GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]]`
	
	`EPSG: http://www.opengis.net/def/crs/OGC/1.3/CRS84`
	
	unless a different coordinate reference system is specified in a parameter `crs`.

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

```
The polygon is defined using a Well Known Text string following
```

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

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

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

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

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

```
`The coords parameter will only support 2D POLYGON`
```

Example 17. A polygon covering the UK

An area covering the UK in WGS84 (from 15°W to 5°E and from 60.95°S to 48.8°S)

```
coords=POLYGON  -15 48.8,-15 60.95,5 60.85,5 48.8,-15 48.8
```

Example 18. Multiple areas

Selecting data for two different regions

```
coords=MULTIPOLYGON  -15 48.8,-15 60.95,5 60.85,5 48.8,-15 48.8),(-6.1 50.3,-4.35 51.4,-2.6 51.6,-2.8 50.6,-5.3 49.9,-6.1,50.3
```

7.10.6. Parameter interpolationX

Requirement 26	**/req/edr/interpolationX-definition**

A	Each resource collection operation MAY support a parameter `interpolationx`with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: interpolationx`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Requirement 27	/req/edr/interpolationX-response
A	If the `interpolationx` parameter is defined the result set SHALL contain values derieved based on the chosen interpolation algorithm at the number of specifed x samples.
B	The interpolationx information is an interpolation method and an interval. The available interpolation methods should be part of the collection response and the samples value is a number :
C	The interpolation information is an interpolation method. The available interpolation methods should be part of the instance metadata:
D	If the interpolation method value does not match one of the listed methods a 403 error message should be returned. `interpolationx = "R"interval "/" interpolation-method`

If not specified the query will not perform any interpolation of data along the X axis. If the interpolationX method does not match an entry in the defined list of valid interpolationX methods for the collection a 403 message SHOULD be returned.

Example 19. An interpolationx

Use the defined linear algorithm to return 10 values between the minimum X value and the maximum X value

`interpolationx=R10/linear`

7.10.7. Parameter interpolationy

Requirement 28	/req/edr/interpolationY-definition

A	Each resource collection operation MAY support a parameter `interpolationy`with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: interpolationy`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Requirement 29	/req/edr/interpolationY-response
A	If the `interpolationy` parameter is defined the result set SHALL contain values derieved based on the chosen interpolation algorithm at the number of specifed y samples.
B	The interpolationy information is an interpolation method and an interval. The available interpolation methods should be part of the collection response and the samples value is a number. `interpolationy = "R"interval "/" interpolation-method`
C	The interpolation information is an interpolation method. The available interpolation methods should be part of the metadata returned by the API:
D	If not specified the query will use the default interpolation method for the chosen collection.
E	If the interpolation method value does not match one of the listed methods a 403 error message should be returned.

If not specified the query will not perform any interpolation of data along the Y axis. If the interpolationY method does not match an entry in the defined list of valid interpolationy methods for the collection a 403 message SHOULD be returned.

Example 20. An interpolationy

Use the defined nearest-neighbour algorithm to return 3 values between the minimum Y value and the maximum Y value

`interpolationy=R3/nearest-neighbour`

7.10.8. Parameter z

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

Example 21. A Z value

for instance if the 850hPa pressure level is being queried

z=850

`When not specified the API MUST return data from the lowest (i.e. nearest the surface) level of data in the collection instance (if the instance has a height dimension).`

7.10.9. Cube

Parameters

Parameter coords

Requirement 30	/req/edr/coords-definition

A	Each resource collection operation SHALL support a parameter `coords` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: coords`
	`in: query`
	`required: true`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

B	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

Requirement 31	/req/edr/coords-response
A	Only those resources that have a spatial geometry that intersects the area defined by the `coords` parameter SHALL be part of the result set.
B	The coordinates SHALL consist of a Well Known Text (WKT) geometry string

C	The coordinate reference system of the values SHALL be interpreted as WGS84 longitude/latitude
	
	`WKT: GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]]`
	
	`EPSG: http://www.opengis.net/def/crs/OGC/1.3/CRS84`
	
	unless a different coordinate reference system is specified in a parameter `crs`.

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

```
The cubes X Y coordinates are defined using Rectangular Polygon as Well Known Text
```

```
coords=POLYGON((x y,x1 y1,x2 y2, x3 y3, x y))
```

```
which are values in the coordinate system defined by the crs query parameter
if crs is not defined the values will be assumed to be WGS84 longitude/latitude
coordinates and heights will be assumed to be in metres above mean sea level
```

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

```
`If the WKT does not define a Rectangle the service will generate an error message`
```

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

Example 22. A cube covering the South West of the UK

```
coords=POLYGON  -6.0 50.0,-4.35 50.0,-4.35 52.0,,-6.0 52.0,-6.0 50.0
```

Parameter interpolationx

Requirement 32	**/req/edr/interpolationX-definition**

A	Each resource collection operation MAY support a parameter `interpolationx`with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: interpolationx`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Requirement 33	/req/edr/interpolationX-response
A	If the `interpolationx` parameter is defined the result set SHALL contain values derieved based on the chosen interpolation algorithm at the number of specifed x samples.
B	The interpolationx information is an interpolation method and an interval. The available interpolation methods should be part of the collection response and the samples value is a number :
C	The interpolation information is an interpolation method. The available interpolation methods should be part of the instance metadata:
D	If the interpolation method value does not match one of the listed methods a 403 error message should be returned. `interpolationx = "R"interval "/" interpolation-method`

If not specified the query will not perform any interpolation of data along the X axis. If the interpolationx method does not match an entry in the defined list of valid interpolationx methods for the collection a 403 message SHOULD be returned.

Example 23. An interpolationx

Use the defined linear algorithm to return 10 values between the minimum X value and the maximum X value

`interpolationx=R10/linear`

Parameter interpolationy

Requirement 34	/req/edr/interpolationY-definition

A	Each resource collection operation MAY support a parameter `interpolationy`with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: interpolationy`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Requirement 35	/req/edr/interpolationY-response
A	If the `interpolationy` parameter is defined the result set SHALL contain values derieved based on the chosen interpolation algorithm at the number of specifed y samples.
B	The interpolationy information is an interpolation method and an interval. The available interpolation methods should be part of the collection response and the samples value is a number. `interpolationy = "R"interval "/" interpolation-method`
C	The interpolation information is an interpolation method. The available interpolation methods should be part of the metadata returned by the API:
D	If not specified the query will use the default interpolation method for the chosen collection.
E	If the interpolation method value does not match one of the listed methods a 403 error message should be returned.

If not specified the query will not perform any interpolation of data along the Y axis. If the interpolationy method does not match an entry in the defined list of valid interpolationy methods for the collection a 403 message SHOULD be returned.

Example 24. An interpolationy

Use the defined nearest-neighbour algorithm to return 3 values between the minimum Y value and the maximum Y value

`interpolationy=R3/nearest-neighbour`

Parameter minz

Define the height of the bottom of the cube i.e. minz=level

The units will default to those of the coordinate reference system. If no CRS is defined the values will be assumed to be height above sea level in metres.

Example 25. A minz

if bottom of cube is at the 950hPa pressure level

minz=950

Parameter maxz

Define the height of the top of the cube i.e. maxz=level

The units will default to those of the coordinate reference system. If no CRS is defined the values will be assumed to be height above sea level in metres.

Example 26. A maxz

if top of cube is at the 650hPa pressure level

minz=650

Parameter interpolationz

Requirement 36	**/req/edr/interpolationZ-definition**

A	Each resource collection operation MAY support a parameter `interpolationz`with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: interpolationz`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Requirement 37	**/req/edr/interpolationZ-response**
A	If the `interpolationz` parameter is defined the result set SHALL contain values derieved based on the chosen interpolation algorithm at the number of specifed z samples.
B	The interpolationz information is an interpolation method and an interval. The available interpolation methods should be part of the collection response and the samples value is a number : `interpolationz = "R" intervals "/" intervalinterpolation-method`
C	The interpolation information is an interpolation method. The available interpolation methods should be part of the instance metadata:
D	If not specified the query will use the default interpolation method for the chosen collection.
E	If the vertical coordinates only contain a single height value the R `intervals` MUST always be 1.
F	If the interpolation method value e does not match one of the listed methods a 403 error message should be returned.

If not specified the query will not perform any interpolation of data along the Z axis. If the interpolationz method does not match an entry in the defined list of valid interpolationz methods for the collection a 403 message SHOULD be returned.

Example 27. An interpolationz

Use the defined cubic-spline algorithm to return 3 values between the minimum Z value and the maximum Z value

`interpolationz=R3/cubic-spline`

7.10.10. Trajectory (corridor)

Parameters

Parameter coords

Requirement 38	/req/edr/coords-definition

A	Each resource collection operation SHALL support a parameter `coords` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: coords`
	`in: query`
	`required: true`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

B	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

Requirement 39	/req/edr/coords-response
A	Only those resources that have a spatial geometry that intersects the area defined by the `coords` parameter SHALL be part of the result set.
B	The coordinates SHALL consist of a Well Known Text (WKT) geometry string

C	The coordinate reference system of the values SHALL be interpreted as WGS84 longitude/latitude
	
	`WKT: GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]]`
	
	`EPSG: http://www.opengis.net/def/crs/OGC/1.3/CRS84`
	
	unless a different coordinate reference system is specified in a parameter `crs`.

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

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

A 2D trajectory, on the surface of earth, at the same time: coords=LINESTRING(51.14 -2.98, 51.36 -2.87, 51.03 -3.15, 50.74 -3.48, 50.9 -3.36)

A 3D trajectory centre, on the surface of the earth but over a time range: 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 corrtrajectoryidor centre, through a 3D volume with height or depth, for the same time: coords=LINESTRINGZ(51.14 -2.98 0.1, 51.36 -2.87 0.2, 51.03 -3.15 0.3, 50.74 -3.48 0.4, 50.9 -3.36 0.5)

A 4D trajectory, through a 3D trajectory centre but over a time range: coords=LINESTRINGZM(51.14 -2.98 0.1 1560507000,51.36 -2.87 0.2 1560507600, 51.03 -3.15 0.3 1560508200, 50.74 -3.48 0.4 1560508500, 50.9 -3.36 0.5 1560510240)

where Z is the height value. `If the specified CRS does not define the height units the heights will units will default to metres above mean sea level`

and M is the time value defined as EPOC seconds since 1970

Example 28. A basic surface route

From Bristol to Exeter

`coords=LINESTRING(51.14 -2.98,51.36 -2.87,51.03 -3.15,50.74 -3.48,50.9 -3.36)`

Example 29. A basic surface route with defined time intervals

From Bristol to Exeter

```
coords=LINESTRING(51.14  -2.98  0  1560507000,51.36  -2.87  0  1560507600,51.03  -3.15  0
1560508200,50.74 -3.48 0 1560508500,50.9 -3.36 0 1560510240)
```

Parameter interpolation

Requirement 40	**/req/edr/interpolation-definition**

A	Each resource collection operation MAY support a parameter `interpolation` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: interpolation`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Requirement 41	**/req/edr/interpolation-response**
A	If the `interpolation` parameter is defined the result set SHALL to data point that is the best match based on the chosen algorithm.
B	The interpolation information is an interpolation method. The available interpolation methods should be part of the instance metadata: `interpolation = interpolation-method`
C	If not specified the query will use the default interpolation method for the chosen collection.
D	If the interpolation method value e does not match one of the listed methods a 403 error message should be returned.

Parameter interpolationz

Requirement 42	**/req/edr/interpolationZ-definition**

A	Each resource collection operation MAY support a parameter `interpolationz`with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: interpolationz`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

Requirement 43	**/req/edr/interpolationZ-response**
A	If the `interpolationz` parameter is defined the result set SHALL contain values derieved based on the chosen interpolation algorithm at the number of specifed z samples.

B	The interpolationz information is an interpolation method and an interval. The available interpolation methods should be part of the collection response and the samples value is a number :
	
	`interpolationz = "R" intervals "/"`
	`intervalinterpolation-method`

C	The interpolation information is an interpolation method. The available interpolation methods should be part of the instance metadata:
D	If not specified the query will use the default interpolation method for the chosen collection.
E	If the vertical coordinates only contain a single height value the R `intervals` MUST always be 1.
F	If the interpolation method value e does not match one of the listed methods a 403 error message should be returned.

7.10.11. Items

Parameters

Due to the concept of `Collections` in EDR the item query is a powerful and flexible tool, but retains only simple options. It could be used to rerun a previously run query but with a replacement for one or more of the input parameters; this would be achieved by using the id field value from the successful output from a previous query as the input id query parameter of item query, then the API user could override one or more of the original query parameters by just entering the query parameters that they want to change (i.e. a trajectory query where the same route is used every day with the same interpolation options, crs and outputFormat and the user just needs to enter the new time range).

Again this is dependant on the data publisher implementing support for the functionality.

Example 30. id

return information for the requested id

id=gfs-query-2020-01-31

As in the EDR concept a `Collection` is a black box it would be possible to overload the id query parameter to perform different actions, as it is the `Collection` black box that interprets how it handles the parameter; for instance in the Metar collection it could be combined with the parametername query parameter and the time parameter to return all Metars for ICAO id KIAD for the requested time period

.

return information for the requested id

/collections/metar/raw/item?id=KIAD&parametername=icao_id&time=2020-01-31T00:00:00Z/2020-02-01T04:00:00Z

This example does have a flaw in that the EDR API provides no mechanism for the user to discover the available icao id's in the metadata (the id's are available in the query results but not in any of the available metadata outputs).

7.10.12. Locations

Parameters

With the locations query a location is defined by a unique identifer, this is a string value. It can be anything as long as it is unique for the required location, for instance a GeoHash `gbsvn` or a WMO station id like `03772`, what3words like `bolt.lime.metro` or place name like `Devon`. The metadata in returned by the API must supply a geospatial definition for the identifier.

Example 31. locid

return all available data for the metar collection for the requested location identifier, where the location is defined by the Heathrow METAR id

/collections/metar/locations/EGLL

Example 32. subset by time and parameter

The API can allow user to subset the required data by time and parameter and also supports requesting data format changes (if supported by the data provider)

/collections/metar/locations/EGLL?time=2020-03-30T00:00Z/2020-03-T12:00Z&parametername=temperature

7.10.13. Instances

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

Parameters

Parameter instanceId

A unique identifier for the instance of the collection

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

1. Return the Raw data instance metadata (instanceId = raw) for the Metar ((collectionId = metar) collection

```
/collections/metar/instance/raw
```

1. Return the Level 1 Quality controlled data instance (instanceId = qc_lvl_1) metadata for the Metar (collectionId = metar) collection

```
/collections/metar/instance/qc_lvl_1
```

Parameter queryType

The queryType options are exactly the same as those available to collections that don't have multiple instances and support the same query parameters and functionality. See the <query-resource-table>> for the mappings of the intial query types proposed for the EDR API.

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

see the QueryType specific Operations section for details of the query parameters supported by the queryTypes.

1. A point query on an Raw data instance(instanceId = raw) for the Metar ((collectionId = metar) collection

```
/collections/metar/instance/raw/point
```

1. A trajectory query on an Raw data instance(instanceId = raw) for the Metar ((collectionId = metar) collection

```
/collections/metar/instance/raw/trajectory
```

7.11. General Requirements

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

7.11.1. 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.

Requirement 44	**/req/core/http**
A	The API SHALL conform to HTTP 1.1.
B	If the API supports HTTPS, then the API SHALL also conform to HTTP over TLS.

7.11.2. HTTP Status Codes

Table 7 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 10. Typical HTTP status codes

Status code	Description
`200`	A successful request.
`202`	A successful request, but the response is still being generated. The response will include a `Retry-After` header field giving a recommendation in seconds for the client to retry.
`304`	An entity tag was provided in the request and the resource has not been changed since the previous request.
`308`	The server cannot process the data through a synchronous request. The response includes a `Location` header field which contains the URI of the location the result will be available at once the query is complete Asynchronous queries.
`400`	The server cannot or will not process the request due to an apparent client error. For example, a query parameter had an incorrect value.
`401`	The request requires user authentication. The response includes a `WWW-Authenticate` header field containing a challenge applicable to the requested resource.
`403`	The server understood the request, but is refusing to fulfill it. While status code `401` indicates missing or bad authentication, status code `403` indicates that authentication is not the issue, but the client is not authorised to perform the requested operation on the resource.
`404`	The requested resource does not exist on the server. For example, a path parameter had an incorrect value.
`405`	The request method is not supported. For example, a POST request was submitted, but the resource only supports GET requests.

Status code	Description
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 7, too.

7.11.3. 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 1	**/rec/core/etag**
A	The service SHOULD support entity tags and the associated headers as specified by HTTP/1.1.

7.11.4. Support for Cross-Origin Requests

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

Recommendation 2	**/rec/core/cross-origin**
A	If the server is intended to be accessed from the browser, cross-origin requests SHOULD be supported. Note that support can also be added in a proxy layer on top of the server.

Two common mechanisms to support cross-origin requests are:

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

7.11.5. Asynchronous queries

It will not always be possible to respond to queries synchronously, if the query requires handling asynchronously the system should 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.

Recommendation 3	**/rec/core/asynchronous-query**
A	If a requests will require significant time to execute, asynchronous queries SHOULD be supported.
B	User SHOULD be redirected to a URI location where the result will be available once completed using a HTTP statuts code of 308 with a header field of *Location* containing the URI.
C	The result URI SHOULD return a HTTP statuts code of 202 with a header field of *Retry_after* with a recommendation in seconds of how long to wait before retrying the request.
D	Once the data is available the result URI SHOULD return a HTTP statuts code of 200 with the data.

- Asynchrony

7.11.6. Coordinate Reference Systems

As discussed in Chapter 9 of the W3C/OGC 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 geographic data and it is important to be clear about the coordinate reference system that coordinates are in.

For the reasons discussed in the Best Practices, EDR APIs MUST always support WGS84 longitude and latitude as a coordinate reference system.

Requirement 45	**/req/collections/crs84**
A	Unless the client explicitly requests a different coordinate reference system, all spatial geometries SHALL be in the CRS84 (WGS 84 longitude/latitude) coordinate reference system for geometries without height information and CRS84h (WGS 84 longitude/latitude plus ellipsoidal height) for geometries with height information.

The implementations compliant with the Core are not required to support publishing geometries in coordinate reference systems other than http://www.opengis.net/def/crs/OGC/1.3/CRS84.

7.11.7. Encodings

While the OAPI Common standard does not specify any mandatory encoding, the following encodings are recommended. See Clause 7 (Overview) for a discussion of this issue.

HTML encoding recomendation:

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

GeoJSON encoding recomendation:

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

CoverageJSON encoding recomendation:

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

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

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

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

As clients simply need to dereference the URI of the link, the implementation details and the mechanism how the encoding is included in the URI of the link are not important. Developers interested in the approach of a particular implementation, for example, to manipulate ("hack") in

the browser address bar, 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.

7.11.8. Link Headers

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

Chapter 8. Requirements classes for encodings

8.1. Overview

This clause specifies two pre-defined requirements classes for encodings to be used by an OGC API implementation. These encodings are commonly used encodings for spatial data on the web:

- HTML
- GeoJSON

Neiter of these encodings are mandatory and an implementation of the Core requirements class may implement either, both, or none of them. Clause 7 (Overview) includes a discussion about recommended encodings.

8.2. Requirement Class "HTML"

Geographic information that is only accessible in formats like GeoJSON or GML has two issues:

- The data is not discoverable using the most common mechanism for discovering information, that is the search engines of the Web,
- The data can not be viewed directly in a browser - additional tools are required to view the data.

Therefore, sharing data on the Web should include publication in HTML. To be consistent with the Web, it should be done in a way that enables users and search engines to access all data.

This is discussed in detail in W3C Best Practice. This standard therefore recommends supporting HTML as an encoding.

The HTML conformance class is defined in API-Common. The API-EDR standard implements that HTML conformance class without modification.

Requirements Class	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/html	
Target type	Web API
Dependency	Conformance Class "Core"
Dependency	API-Common HTML
Dependency	HTML5
Dependency	Schema.org

Requirement 46	**/req/html/api-common**

Extends	/req/core/api-common
The API SHALL demonstrate conformance with the following Requirements Class of the OGC API-Common version 1.0 Standard.	
A	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/html

Requirement 47	/req/html/conformance
The list of Conformance Classes advertised by the API SHALL include:	
A	http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/html

8.3. Requirement Class "GeoJSON"

GeoJSON is a commonly used format that is simple to understand and well supported by tools and software libraries. Since most Web developers are comfortable with using a JSON-based format, supporting GeoJSON is recommended if the resource can be represented in GeoJSON for the intended use.

The GeoJSON conformance class is defined in API-Common. The API-EDR standard extends that GeoJSON conformance class.

Requirements Class	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/json	
Target type	Web API
Dependency	Requirements Class "API-Common Core"
Dependency	API-Common GeoJSON
Dependency	GeoJSON
Pre-conditions	1) The API advertises conformance to the JSON Conformance Class 2) The client negotiates use of the JSON or GeoJSON encoding.

Requirement 48	/req/json/api-common
Extends	/req/core/api-common
The API SHALL demonstrate conformance with the following Requirements Class of the OGC API-Common version 1.0 Standard.	
A	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/geojson

Requirement 49	/req/json/conformance
The list of Conformance Classes advertised by the API SHALL include:	

A	http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/json

NOTE The following schema names are from API-Features and need to be updated.

Templates for the definition of the schemas for the GeoJSON responses in JSON Schema definitions are available at collections.yaml and collectionInfo.yaml.

These are generic schemas that do not include any application schema information about specific resource types or their properties.

Chapter 9. Requirements class "OpenAPI 3.0"

9.1. Basic requirements

Requirements Class	
http://www.opengis.net/spec/ogcapi-edr-1/1.0/req/oas30	
Target type	Web API
Dependency	Conformance Class "Core"
Dependency	OGC API-Common Standard 1.0
Dependency	OpenAPI Specification 3.0.2

The OpenAPI 3.0 Requirements Class is applicable to API-EDR as well. So an implementation of API-EDR which supports OpenAPI 3.0 as an API Description format must also comply with the API-Common oas30 Conformance Class.

Requirement 50	**/req/oas30/oas-common**
Extends	/req/core/api-common
A	The API SHALL demonstrate conformance with the following Requirements Class of the OGC API-Common version 1.0 Standard. http://www.opengis.net/spec/ogcapi-common-1/1.0/req/oas30.

Implementations must also advertise conformance with this Requirements Class.

Requirement 51	**/req/oas30/conformance**
The list of Conformance Classes advertised by the API SHALL include:	
A	http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/oas30

Two example OpenAPI documents are included in Annex B.

9.2. Complete definition

Requirement 52	**/req/oas30/completeness**
A	The OpenAPI definition SHALL specify for each operation all HTTP Status Codes and Response Objects that the API uses in responses.
B	This includes the successful execution of an operation as well as all error situations that originate from the server.

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

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

9.3. Exceptions

Requirement 53	**/req/oas30/exceptions-codes**
A	For error situations that originate from an API server, the API definition SHALL cover all applicable HTTP Status Codes.

Example 33. An exception response object definition

```
description: An error occurred.
content:
  application/json:
    schema:
      $ref:
https://raw.githubusercontent.com/opengeospatial/OAPI/openapi/schemas/exception.ya
ml
  text/html:
    schema:
      type: string
```

9.4. Security

Requirement 54	**/req/oas30/security**
A	For cases, where the operations of the API are access-controlled, the security scheme(s) and requirements SHALL be documented in the OpenAPI definition.

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.

Chapter 10. Media Types

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

- `application/geo+json` for feature collections and features, and
- `application/json` for all other resources.

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

- `application/gml+xml;version=3.2` for any 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, and
- `application/xml` for all other resources.

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

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

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

Annex A: Abstract Test Suite (Normative)

A.1. Introduction

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

A.2. Conformance Class Core

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

A.2.1. General Tests

HTTP

Abstract Test 1	**/ats/core/http**
Test Purpose	Validate that the resource paths advertised through the API conform with HTTP 1.1 and, where approprate, TLS.
Requirement	/req/core/http
Test Method	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.

A.2.2. Landing Page {root}/

Abstract Test 2	**/ats/core/root-op**
Test Purpose	Validate that a landing page can be retrieved from the expected location.

Requirement	/req/core/root-op
Test Method	1. Issue an HTTP GET request to the URL {root}/ 2. Validate that a document was returned with a status code 200 3. Validate the contents of the returned document using test /ats/core/root-success.

Abstract Test 3	**/ats/core/root-success**
Test Purpose	Validate that the landing page complies with the require structure and contents.
Requirement	/req/core/root-success
Test Method	Validate the landing page for all supported media types using the resources and tests identified in Table 8 For formats that require manual inspection, perform the following: a. Validate that the landing page includes a "service-desc" and/or "service-doc" link to an API Definition b. Validate that the landing page includes a "conformance" link to the conformance class declaration c. 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 11. Schema and Tests for Landing Pages

Format	Schema Document	Test ID
HTML	landingPage.json	/ats/html/content
JSON	landingPage.json	/ats/geojson/content

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

Abstract Test 4	**/ats/core/api-definition-op**
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	1. Construct a path for each API Definition link on the landing page 2. Issue a 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 /ats/core/api-definition-success.

Abstract Test 5	**/ats/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.

A.2.4. Conformance Path {root}/conformance

Abstract Test 6	**/ats/core/conformance-op**
Test Purpose	Validate that a Conformance Declaration can be retrieved from the expected location.
Requirement	/req/core/conformance-op
Test Method	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 /ats/core/conformance-success.

Abstract Test 7	**/ats/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	1. Validate the response document against OpenAPI 3.0 schema confClasses.yaml 2. Validate that the document includes the conformance class "http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/core" 3. Validate that the document list all OGC API conformance classes that the API implements.

A.3. Conformance Class Collections

Conformance Class	
http://www.opengis.net/spec/ogcapi-common/1.0/conf/collections	
Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi_common/1.0/req/collections
Dependency	Conformance Class "OAPI Core"

A.3.1. General Tests

CRS 84

Abstract Test 8	**/ats/collections/crs84**
Test Purpose	Validate that all spatial geometries provided through the API are in the CRS84 spatial reference system unless otherwise requested by the client.
Requirement	/req/collections/crs84
Test Method	1. Do not specify a coordinate reference system in any request. All spatial data should be in the CRS84 reference system. 2. Validate retrieved spatial data using the CRS84 reference system.

A.3.2. Feature Collections {root}/collections

Abstract Test 9	**/ats/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	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 /ats/collections/rc-md-success.

Abstract Test 10	**/ats/collections_rc-md-success**
Test Purpose	Validate that the Collections content complies with the required structure and contents.
Requirement	/req/collections/rc-md-success, /req/collections/crs84
Test Method	1. Validate that all response documents comply with /ats/collections/rc-md-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 9

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 12. Schema and Tests for Collections content

Format	Schema Document	Test ID
HTML	collections.json	/ats/html/content
JSON	collections.json	/ats/geojson/content

A.3.3. Feature Collection {root}/collections/{collectionId}

Abstract Test 11	**/ats/collections/src-md-op**
Test Purpose	Validate that the Collection content can be retrieved from the expected location.

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

Abstract Test 12	**/ats/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.

A.3.4. Features {root}/collections/{collectionId}/items

NOTE	This test is too Feature centric. Will need to be greatly reduced in scope.

Abstract Test 13	**/ats/collections/rc-op**
Test Purpose	Validate that resources can be identified and extracted from a Collection using query parameters.
Requirement	/req/collections/rc-op

Test Method	1. For every resource collection identified in Collections, issue an HTTP GET request to the URL `/collections/{collectionId}/items` where `{collectionId}` is the `id` property for a Collection described in the Collections content. 2. Validate that a document was returned with a status code 200. Repeat these tests using the following parameter tests: **Bounding Box:** • Parameter /ats/collections/rc-bbox-definition • Response /ats/collections/rc-bbox-response **DateTime:** • Parameter /ats/collections/rc-time-definition • Response /ats/collections/rc-time-response Execute requests with combinations of the "bbox" and "datetime" query parameters and verify that only features are returned that match both selection criteria.

Abstract Test 14	**/ats/collections/rc-bbox-definition**
Test Purpose	Validate that the bounding box query parameters are constructed correctly.
Requirement	/req/collections/rc-bbox-definition

Test Method	Verify that the `bbox` query parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):
	
	`name: bbox`
	`in: query`
	`required: false`
	`schema:`
	`  type: array`
	`  minItems: 4`
	`  maxItems: 6`
	`  items:`
	`  type: number`
	`style: form`
	`explode: false`
	
	Use a bounding box with four numbers in all requests:
	
	• Lower left corner, WGS 84 longitude
	• Lower left corner, WGS 84 latitude
	• Upper right corner, WGS 84 longitude
	• Upper right corner, WGS 84 latitude

Abstract Test 15	**/ats/collections/rc-bbox-response**
Test Purpose	Validate that the bounding box query parameters are processed corrrectly.
Requirement	/req/collections/rc-bbox-response
Test Method	1. Verify that only resources that have a spatial geometry that intersects the bounding box are returned as part of the result set. 2. Verify that the `bbox` parameter matched all resources in the collection that were not associated with a spatial geometry (this is only applicable for datasets that include resources without a spatial geometry). 3. Verify that the coordinate reference system of the geometries is WGS 84 longitude/latitude ("http://www.opengis.net/def/crs/OGC/1.3/CRS84" or "http://www.opengis.net/def/crs/OGC/0/CRS84h") since no parameter `bbox-crs` was specified in the request.

Abstract Test 16	**/ats/collections/rc-time-definition**
Test Purpose	Validate that the dateTime query parameters are constructed correctly.
Requirement	/req/collections/rc-time-definition

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

Abstract Test 17	**/ats/collections/rc-time-response**
Test Purpose	Validate that the dataTime query parameters are processed correctly.
Requirement	/req/collections/rc-time-response
Test Method	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 dateime parameter complies with the syntax described in /req/collections/rc-time-response.

Abstract Test 18	**/ats/collections/rc-response**
Test Purpose	Validate that the Resource Collection complies with the require structure and contents.
Requirement	/req/collections/rc-response

Test Method	The test method is specific to the resource type returned.

A.3.5. Second Tier Tests

These tests are invoked by other tests.

Extent

Abstract Test 19	**/ats/coollections/rc-md-extent**
Test Purpose	Validate that the `extent` property if it is present
Requirement	/req/collections/rc-md-extent
Test Method	1. Verify that the `extent` provides bounding boxes that include all spatial geometries 2. Verify that if the `extent` provides time intervals that include all temporal geometries in this collection. 3. A temporal extent of `null` indicates an open time interval.

Items

Abstract Test 20	**/ats/collections/rc-md-items**
Test Purpose	Validate that each collection provided by the server is described in the Collections Metadata.
Requirement	/req/collections/rc-md-items
Test Method	1. Verify that there is an entry in the `collections` array of the Collections Metadata for each feature collection provided by the API. 2. Verify that each collection entry includes an identifier. 3. Verify that each collection entry includes links in accordance with /collections/rc-md-items-links. 4. Verify that if the collection entry includes an extent property, that that property complies with /collections/rc-md-extent 5. Validate each collection entry for all supported media types using the resources and tests identified in Table 10

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 13. Schema and Tests for Collection Entries

Format	Schema Document	Test ID
HTML	collectionInfo.json	/ats/html/content
JSON	collectionInfo.json	/ats/geojson/content

Abstract Test 21	**/ats/collections/rc-md-items-links**
Test Purpose	Validate that each Feature Collection metadata entry in the Collections Metadata document includes all required links.
Requirement	/req/collections/rc-md-items-links
Test Method	1. Verify that each Collection item in the Collections Metadata document includes a `link` property for each supported encoding. 2. Verify that the `links` properties of the collection includes an item for each supported encoding with a link to the features resource (relation: `items`). 3. Verify that all links include the `rel` and `type` link parameters.

Links

Abstract Test 22	**/ats/collections/rc-md-links**
Test Purpose	Validate that the required links are included in the Collections Metadata document.
Requirement	/req/collections/rc-md-links
Test Method	Verify that the response document includes: 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`). Verify that all links include the `rel` and `type` link parameters.

A.4. Conformance Class GeoJSON

Conformance Class
http://www.opengis.net/spec/ogcapi-common/1.0/conf/geojson

Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi_common/1.0/req/geojson
Dependency	Conformance Class "OAPI Core"

A.4.1. GeoJSON Definition

Abstract Test 23	**/ats/geojson/definition**
Test Purpose	Verify support for JSON and GeoJSON
Requirement	/req/geojson/definition
Test Method	1. A resource is requested with response media type of `application/geo+json` 2. All `200`-responses SHALL support the following media types: ◦ `application/geo+json` for resources that include feature content, and ◦ `application/json` for all other resources.

A.4.2. GeoJSON Content

Abstract Test 24	**/ats/geojson/content**
Test Purpose	Verify the content of a GeoJSON document given an input document and schema.
Requirement	/req/geojson/content
Test Method	1. Validate that the document is a GeoJSON document. 2. Validate the document against the schema using an JSON Schema validator.

A.5. Conformance Class HTML

Conformance Class	
http://www.opengis.net/spec/ogcapi-common/1.0/conf/html	
Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi_common/1.0/req/html
Dependency	Conformance Class "OAPI Core"

A.5.1. HTML Definition

Abstract Test 25	**/ats/html/definition**
Test Purpose	Verify support for HTML
Requirement	/req/html/definition
Test Method	Verify that every `200`-response of every operation of the API where HTML was requested is of media type `text/html`

A.5.2. HTML Content

Abstract Test 26	**/ats/html/content**
Test Purpose	Verify the content of an HTML document given an input document and schema.
Requirement	/req/html/content
Test Method	1. Validate that the document is an HTML 5 document 2. Manually inspect the document against the schema.

A.6. Conformance Class OpenAPI 3.0

Conformance Class	
http://www.opengis.net/spec/ogcapi-common/1.0/conf/oas3	
Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi_common/1.0/req/oas3
Dependency	Conformance Class "OAPI Core"

Abstract Test 27	**/ats/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.

Abstract Test 28	**/ats/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.

Abstract Test 29	**/ats/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	1. Verify that an OpenAPI definition in JSON is available using the media type `application/vnd.oai.openapi+json;version=3.0` and link relation `service-desc` 2. Verify that an HTML version of the API definition is available using the media type `text/html` and link relation `service-doc`.

Abstract Test 30	**/ats/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.

Abstract Test 31	**/ats/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	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.

Abstract Test 32	**/ats/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	1. Identify all authentication protocols supported by the API. 2. Validate that each authentication protocol is described inthe OpenAPI document by a Security Schema Object and its' use specified by a Security Requirement Object.

Annex B: Examples (Informative)

B.1. Example Landing Pages

Example 34. 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", "type": "application/openapi+json;version=3.0", "title":
"the API definition" },
    { "href": "http://data.example.org/conformance",
      "rel": "conformance", "type": "application/json", "title": "OGC conformance
classes implemented by this API" },
    { "href": "http://data.example.org/collections",
      "rel": "data", "type": "application/json", "title": "Metadata about the
resource collections" }
  ]
}
```

B.2. API Description Examples

NOTE include::examples/tbd.adoc[]

B.3. Conformance Examples

Example 35. Conformance Response

This example response in JSON is for an OGC API Features that supports OpenAPI 3.0 for the API definition and HTML and GeoJSON as encodings for resources.

```
{
  "conformsTo": [
    "http://www.opengis.net/spec/ogcapi-features-1/1.0/req/core",
    "http://www.opengis.net/spec/ogcapi-features-1/1.0/req/oas30",
    "http://www.opengis.net/spec/ogcapi-features-1/1.0/req/html",
    "http://www.opengis.net/spec/ogcapi-features-1/1.0/req/geojson"
  ]
}
```

B.4. Collections Metadata Examples

Example 36. Collections metadata response document

This collection metadata example response in JSON is for a service with collections "Metar", "GFS height above ground parameters" and "DEM". It includes links to the collection resource in all formats that are supported by the API (link relation type: "items").

There is a link to the feature collections response itself (link relation type: "self").

Representations of this resource in other formats are referenced using link relation type "alternate".

An additional link is to a GML application schema for the dataset - using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "describedBy".

A bulk download of all the features in the dataset is referenced using link relation type "enclosure"

Finally there are also links to the license information for the building data (using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "license").

Reference system information is not provided as the service provides geometries only in the default system (spatial: WGS 84 longitude/latitude; temporal: Gregorian calendar).


```
{
  "links": [
    { "href": "http://data.example.org/collections.json",
      "rel": "self", "type": "application/json", "title": "this document" },
    { "href": "http://data.example.org/collections.html",
      "rel": "alternate", "type": "text/html", "title": "this document as HTML" },
    { "href": "http://schemas.example.org/1.0/foobar.xsd",
      "rel": "describedBy", "type": "application/xml", "title": "XML schema for
Acme Corporation data" }
  ],
  "collections": [
    {
      "id": "metar",
      "title": "Metars",
      "description": "Metar observations",
      "extent": {
        "spatial": [ -180.0, -89.9, 180, 89.9 ],
        "temporal": [ "2020-01-30T12:00:00Z/2020-01-13T11:00:00Z" ]
      },
      "keywords": ["Temperature","Dewpoint temperature","Wind speed","Maximum wind
gust speed","Wind direction","visibility","Pressure","Pressure reduced to mean sea
level","Raw Observation", "id"]
      "links": [
```

```
        { "href": "http://data.example.org/collections/metars",
          "rel": "describedBy", "type": "application/json",
          "title": "Metar collection metadata" },
        { "href": "https://data.example.org/collection/metar/link-for-more-
details",
          "rel": "describedBy", "type": "source",
          "title": "Further information about the Metar data" },
        { "href": "https://data.example.org/collection/metar/link-for-publisher-
details",
          "rel": "describedBy", "type": "publisher",
          "title": "Further information about the data publisher" },
        { "href": "http://data.example.org/collections/metars/position",
          "rel": "data", "type": "position",
          "title": "Point query for the Metar collection" },
        { "href": "http://data.example.org/collections/metars/area",
          "rel": "data", "type": "area",
          "title": "Polygon query for the Metar collection" },
        { "href": "http://data.example.org/collections/metars?outputFormat=html",
          "rel": "describedBy", "type": "text/html",
          "title": "Metadata for the Metar collection" }
        { "href": "https://data.example.org/collection/metar/licence-doc",
          "rel": "describedBy", "type": "licence",
          "title": "Licence for the Metar collection" }
        { "href": "https://data.example.org/collection/metar/rights-doc",
          "rel": "describedBy", "type": "rights",
          "title": "Usage terms for the Metar collection" }

      ]
    },
    {
      "id": "gfs_time-height_above_ground-lat-lon",
      "title": "GFS time, height_above_ground lat-lon",
      "description": "GFS parameters based on heights above ground level",
      "extent": {
        "spatial": [ -180.0, -89.9, 180, 89.9 ],
        "temporal": [ "2020-02-01T00:00:00Z/2020-02-12T00:00:00Z" ],
        "vertical": ["2.0","10.0","80.0"]
      },

"keywords":["Maximum_temperature_height_above_ground_Mixed_intervals_Maximum","Min
imum_temperature_height_above_ground_Mixed_intervals_Minimum"],
      "links": [
        { "href": "https://data.example.org/collection/gfs_time-
height_above_ground-lat-lon/licence-doc",
          "rel": "describedBy", "type": "licence",
          "title": "Licence for the gfs_time-height_above_ground-lat-lon
collection" }
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/link-for-more-details",
          "rel": "describedBy", "type": "source",
          "title": "Further information about the gfs data" },
```

```
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/link-for-publisher-details",
          "rel": "describedBy", "type": "publisher",
          "title": "Further information about the data publisher" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances",
          "rel": "data", "type": "application/json",
          "title": "List of instances of the GFS collection" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/rights-doc",
          "rel": "describedBy", "type": "rights",
          "title": "Usage terms for the GFS collection" }
      ]
    },
    {
      "id": "dem",
      "title": "DEM",
      "description": "Global Digital Elevation Model height data",
      "extent": {
        "spatial": [ -180.0, -89.9, 180, 89.9 ]
      },
      "keywords":["Height above Mean Sea Level"]
      "links": [
        { "href": "http://data.example.org/collections/dem",
          "rel": "describedBy", "type": "application/json",
          "title": "Metadata fore the dem collection" },
        { "href": "https://data.example.org/collection/dem/licence-doc",
          "rel": "describedBy", "type": "licence",
          "title": "Licence for the dem collection" }
        { "href": "http://data.example.org/collections/dem/link-for-more-details",
          "rel": "describedBy", "type": "source",
          "title": "Further information about the dem data" },
        { "href": "http://data.example.org/collections/dem/link-for-publisher-
details",
          "rel": "describedBy", "type": "publisher",
          "title": "Further information about the data publisher" },
        { "href": "http://data.example.org/collections/dem/position",
          "rel": "data", "type": "position",
          "title": "Position query on the DEM collection" },
        { "href": "http://data.example.org/collections/dem/area",
          "rel": "data", "type": "area",
          "title": "Area query on the DEM collection" },
        { "href": "http://data.example.org/collections/dem/trajectory",
          "rel": "data", "type": "trajectory",
          "title": "Trajectory query on the DEM collection" },
        { "href": "http://data.example.org/collections/dem/rights-doc",
          "rel": "describedBy", "type": "rights",
          "title": "Usage terms for the DEM collection" }


      ]
```

```
    }

  ]
}
```

B.5. Collection Metadata Examples

Example 37. Collection instance metadata response document

This collection metadata example response in JSON lists the available instances for the Metar collection. It includes links to the collection resource in all formats that are supported by the API (link relation type: "items").

There is a link to the feature collections response itself (link relation type: "self").

Representations of this resource in other formats are referenced using link relation type "alternate".

An additional link is to a GML application schema for the dataset -using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "describedBy".

A bulk download of all the features in the dataset is referenced using link relation type "enclosure"

Finally there are also links to the license information for the building data (using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "license").

Reference system information is not provided as the service provides geometries only in the default system (spatial: WGS 84 longitude/latitude; temporal: Gregorian calendar).

```
{
  "links": [
    { "href": "http://data.example.org/collections.json",
      "rel": "self", "type": "application/json", "title": "this document" },
    { "href": "http://data.example.org/collections.html",
      "rel": "alternate", "type": "text/html", "title": "this document as HTML" },
    { "href": "http://schemas.example.org/1.0/foobar.xsd",
      "rel": "describedBy", "type": "application/xml", "title": "XML schema for
Acme Corporation data" }
  ],
  "collections": [
    {
      "id": "raw",
      "title": "Raw",
      "description": "Raw Metar observations",
      "extent": {
```

```
        "spatial": [ -180.0, -89.9, 180, 89.9 ],
        "temporal": [ "2020-01-30T12:00:00Z/2020-01-13T11:00:00Z" ]
      },
      "dataDetails": {
          "@context": {
              "@version": 1.1,
              "xsd": "http://www.w3.org/2001/XMLSchema#",
              "dc": "http://purl.org/dc/terms/",
              "dcam":"http://purl.org/dc/dcam/"
          },

          "dc:accessRights": {},
          "dcam:domainIncludes":
["temperature","wind_speed","wind_direction","maximum_wind_gust_speed","speci_visi
bility","pressure","pressure_msl","raw_observation","id"]

      },
      "links": [
        { "href": "http://data.example.org/collections/metars",
          "rel": "describedBy", "type": "application/json",
          "title": "Metadata for Metar data" },
        { "href": "http://data.example.org/collections/metars?outputFormat=html",
          "rel": "describedBy", "type": "text/html",
          "title": "Metadata for Metar data" },
        { "href": "https://data.example.org/collection/metar/licence-doc",
          "rel": "describedBy", "type": "metadata",
          "title": "Licence for Metar data" },
        { "href": "https://data.example.org/collection/metar/link-for-more-
details",
          "rel": "describedBy", "type": "source",
          "title": "Further information about the Metar data" },
        { "href": "https://data.example.org/collection/metar/link-for-publisher-
details",
          "rel": "describedBy", "type": "publisher",
          "title": "Further information about the data publisher" },
        { "href": "http://data.example.org/collections/metars/position",
          "rel": "data", "type": "position",
          "title": "Position query for the Metar collection" }
        { "href": "http://data.example.org/collections/metars/locations",
          "rel": "describedBy", "type": "locations",
          "title": "List location identifiers for the Metar collection" }

      ]
    }
  ]
}
```

Example 38. Collection metadata response document

This collection metadata example response in JSON lists the available instances of the DEM

collection. It includes links to the collection resource in all formats that are supported by the API (link relation type: "items").

There is a link to the feature collections response itself (link relation type: "self").

Representations of this resource in other formats are referenced using link relation type "alternate".

An additional link is to a GML application schema for the dataset - using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "describedBy".

A bulk download of all the features in the dataset is referenced using link relation type "enclosure"

Finally there are also links to the license information for the building data (using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "license").

Reference system information is not provided as the service provides geometries only in the default system (spatial: WGS 84 longitude/latitude; temporal: Gregorian calendar).

```
{
  "links": [
    { "href": "http://data.example.org/collections.json",
      "rel": "self", "type": "application/json", "title": "this document" },
    { "href": "http://data.example.org/collections.html",
      "rel": "alternate", "type": "text/html", "title": "this document as HTML" },
    { "href": "http://schemas.example.org/1.0/foobar.xsd",
      "rel": "describedBy", "type": "application/xml", "title": "XML schema for
Acme Corporation data" }
  ],
  "collections": [
    {
      "id": "data",
      "title": "data",
      "description": "Global Digital Elevation Model height data",
      "extent": {
        "spatial": [ -180.0, -89.9, 180, 89.9 ]
      },
      "links": [
        { "href": "https://data.example.org/collection/dem/link-for-more-details",
          "rel": "describedBy", "type": "source",
          "title": "Further information about the DEM data" },
        { "href": "https://data.example.org/collection/dem/link-for-publisher-
details",
          "rel": "describedBy", "type": "publisher",
          "title": "Further information about the data publisher" },
        { "href": "http://data.example.org/collections/dem/position",
          "rel": "data", "type": "position",
          "title": "Point query for the DEM collection" },
        { "href": "http://data.example.org/collections/dem/area",
          "rel": "data", "type": "area",
          "title": "Area query for the DEM collection" },
        { "href": "http://data.example.org/collections/dem?outputFormat=html",
          "rel": "describedBy", "type": "text/html",
          "title": "Metadata for the dem collection" },
        { "href": "https://data.example.org/collection/dem/licence-doc",
          "rel": "describedBy", "type": "licence",
          "title": "Licence for the dem collection" }
      ]
    }

  ]
}
```

B.6. Instance Metadata Examples

Example 39. Collection instance metadata response document

This collection metadata example response in JSON lists the available instances for the gfs_time-height_above_ground-lat-lon gfs collection. It includes links to the collection resource in all formats that are supported by the API (link relation type: "items").

There is a link to the feature collections response itself (link relation type: "self").

Representations of this resource in other formats are referenced using link relation type "alternate".

An additional link is to a GML application schema for the dataset -using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "describedBy".

A bulk download of all the features in the dataset is referenced using link relation type "enclosure"

Finally there are also links to the license information for the building data (using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "license").

Reference system information is not provided as the service provides geometries only in the default system (spatial: WGS 84 longitude/latitude; temporal: Gregorian calendar).


```
{
  "links": [
    { "href": "http://data.example.org/collections.json",
      "rel": "self", "type": "application/json", "title": "this document" },
    { "href": "http://data.example.org/collections.html",
      "rel": "alternate", "type": "text/html", "title": "this document as HTML" },
    { "href": "http://schemas.example.org/1.0/foobar.xsd",
      "rel": "describedBy", "type": "application/xml", "title": "XML schema for
Acme Corporation data" },
    { "href": "https://data.example.org/collection/gfs_time-height_above_ground-
lat-lon/link-for-more-details",
      "rel": "describedBy", "type": "source",
      "title": "Further information about the gfs_time-height_above_ground-lat-lon
collection" },
    { "href": "https://data.example.org/collection/gfs_time-height_above_ground-
lat-lon/link-for-publisher-details",
      "rel": "describedBy", "type": "publisher",
      "title": "Further information about the data publisher" },
    { "href": "https://data.example.org/collection/gfs_time-height_above_ground-
lat-lon/licence-doc",
      "rel": "describedBy", "type": "licence",
      "title": "Licence for the the gfs_time-height_above_ground-lat-lon latest
run instance of the collection" }
  ],
  "instances": [
    {
      "id": "latest",
```

```
      "title": "Latest",
      "description": "GFS parameters based on heights above ground level from the
latest model run",
      "extent": {
        "spatial": [ -180.0, -89.9, 180, 89.9 ],
        "temporal": [ "2020-02-01T12:00:00Z/2020-02-12T00:00:00Z" ],
        "vertical": ["2.0","10.0","80.0"]
      },

      "links": [
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/latest",
          "rel": "describedBy", "type": "application/json",
          "title": "Metadata for the the gfs_time-height_above_ground-lat-lon
latest run instance of the collection" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/latest?outputFormat=html",
          "rel": "describedBy", "type": "text/html",
          "title": "Metadata for the the gfs_time-height_above_ground-lat-lon
latest run instance of the collection" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/latest/position",
          "rel": "data", "type": "position",
          "title": "Point query for the the gfs_time-height_above_ground-lat-lon
latest run instance of the collection" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/latest/area",
          "rel": "data", "type": "area",
          "title": "Area query for the gfs_time-height_above_ground-lat-lon latest
run instance of the collection" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/latest/trajectory",
          "rel": "data", "type": "trajectory",
          "title": "Trajectory query for the gfs_time-height_above_ground-lat-lon
latest run instance of the collection" }
      ]
    },
    {
      "id": "2020-02-01T12:00:00Z",
      "title": "2020-02-01T12:00:00Z",
      "description": "GFS parameters based on heights above ground level from the
2020-02-01T12:00:00Z run",
      "extent": {
        "spatial": [ -180.0, -89.9, 180, 89.9 ],
        "temporal": [ "2020-02-01T12:00:00Z/2020-02-12T00:00:00Z" ],
        "vertical": ["2.0","10.0","80.0"]
      },
      "links": [
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/2020-02-01T12:00:00Z",
          "rel": "describedBy", "type": "application/json",
```

```
          "title": "Metadata for the the gfs_time-height_above_ground-lat-lon
2020-02-01T12:00:00Z run instance of the collection" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/2020-02-01T12:00:00Z?outputFormat=html",
          "rel": "describedBy", "type": "text/html",
          "title": "Metadata for the the gfs_time-height_above_ground-lat-lon
2020-02-01T12:00:00Z run instance of the collection" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/2020-02-01T12:00:00Z/position",
          "rel": "data", "type": "position",
          "title": "Point query for the the gfs_time-height_above_ground-lat-lon
2020-02-01T12:00:00Z run instance of the collection" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/2020-02-01T12:00:00Z/area",
          "rel": "data", "type": "area",
          "title": "Area query for the gfs_time-height_above_ground-lat-lon 2020-
02-01T12:00:00Z run instance of the collection" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/2020-02-01T12:00:00Z/trajectory",
          "rel": "data", "type": "trajectory",
          "title": "Trajectory query for the gfs_time-height_above_ground-lat-lon
2020-02-01T12:00:00Z run instance of the collection" }

      ]
    },
    {
      "id": "2020-02-01T00:00:00Z",
      "title": "2020-02-01T00:00:00Z",
      "description": "GFS parameters based on heights above ground level from the
2020-02-01T00:00:00Z run",
      "extent": {
        "spatial": [ -180.0, -89.9, 180, 89.9 ],
        "temporal": [ "2020-02-01T00:00:00Z/2020-02-11T12:00:00Z" ],
        "vertical": ["2.0","10.0","80.0"]
      },
      "dataDetails": {
          "@context": {
              "@version": 1.1,
              "xsd": "http://www.w3.org/2001/XMLSchema#",
              "dc": "http://purl.org/dc/terms/",
              "dcam":"http://purl.org/dc/dcam/"
          },

          "dc:accessRights": {},
          "dc:source": {
              "dc:title":"datasource name",
              "dc:identifier" : "https://data.example.org/collection/gfs_time-
height_above_ground-lat-lon/link-for-more-details"
          },
          "dc:publisher" : "contact@example.org",
          "dcam:domainIncludes":
```

```
["appparent_temperature_height_above_ground","dewpoint_temperature_height_above_gr
ound","relative_humidity_height_above_ground"]

      },
      "supportedQueryTypes": {"point","polygon","cube","trajectory"}
      "links": [
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/2020-02-01T00:00:00Z",
          "rel": "describedBy", "type": "application/json",
          "title": "Metadata for the the gfs_time-height_above_ground-lat-lon
2020-02-01T12:00:00Z run instance of the collection" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances2020-02-01T00:00:00Z?outputFormat=html",
          "rel": "describedBy", "type": "text/html",
          "title": "Metadata for the the gfs_time-height_above_ground-lat-lon
2020-02-01T12:00:00Z run instance of the collection" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/2020-02-01T00:00:00Z/position",
          "rel": "data", "type": "position",
          "title": "Point query for the the gfs_time-height_above_ground-lat-lon
2020-02-01T00:00:00Z run instance of the collection" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/2020-02-01T00:00:00Z/area",
          "rel": "data", "type": "area",
          "title": "Area query for the gfs_time-height_above_ground-lat-lon 2020-
02-01T00:00:00Z run instance of the collection" },
        { "href": "http://data.example.org/collections/gfs_time-
height_above_ground-lat-lon/instances/2020-02-01T00:00:00Z/trajectory",
          "rel": "data", "type": "trajectory",
          "title": "Trajectory query for the gfs_time-height_above_ground-lat-lon
2020-02-01T00:00:00Z run instance of the collection" }
      ]
    }

  ]
}
```

B.7. Position Query Metadata Examples

Example 40. Collection instance metadata response document

This metadata example response in JSON is for a the latest instance of the fs_time-height_above_ground-lat-lon collection. It includes links to the collection resource in all formats that are supported by the API (link relation type: "items").

There is a link to the feature collections response itself (link relation type: "self").

Representations of this resource in other formats are referenced using link relation type "alternate".

An additional link is to a GML application schema for the dataset - using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "describedBy".

A bulk download of all the features in the dataset is referenced using link relation type "enclosure"

Finally there are also links to the license information for the building data (using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "license").

Reference system information is not provided as the service provides geometries only in the default system (spatial: WGS 84 longitude/latitude; temporal: Gregorian calendar).


```
{
  "links": [
    {
      "href": "http://data.example.org/collections/gfs_025_time-
height_above_ground-lat-lon/latest/position?outputFormat=application%2Fjson",
      "rel": "self",
      "type": "application/json",
      "title": "latest position metadata document as json"
    },
    {
      "href": "http://data.example.org/collections/gfs_025_time-
height_above_ground-lat-lon/latest/position?outputFormat=application%2Fx-yaml",
      "rel": "alternate",
      "type": "application/x-yaml",
      "title": "latest position metadata document as yaml"
    },
    {
      "href": "http://data.example.org/collections/gfs_025_time-
height_above_ground-lat-lon/latest/position?outputFormat=text%2Fxml",
      "rel": "alternate",
      "type": "text/xml",
      "title": "latest position metadata document as xml"
    },
    {
      "href": "http://data.example.org/collections/gfs_025_time-
height_above_ground-lat-lon/latest/position?outputFormat=text%2Fhtml",
      "rel": "alternate",
      "type": "text/html",
      "title": "latest position metadata document as html"
    }
  ],
  "id": "gfs_time-height_above_ground-lat-lon-latest",
  "title": "Latest Global Forecast System (GFS) time height_above_ground lat lon",
  "description": "Lates Global Forecast System (GFS) Model Global 0.25 degree Data
time height_above_ground lat lon",
  "parameters": {
```

```
    "Apparent_temperature_height_above_ground": {
      "description": {
        "en": "Apparent_temperature_height_above_ground"
      },
      "unit": {
        "label": {
          "en": "kelvin"
        },
        "symbol": {
          "value": "K",
          "type": "http://data.example.org/metadata/uom/UCUM/K"
        }
      },
      "observedProperty": {
        "id": "http://data.example.org/metadata/grib2/codeflag/4.2-0-0-21",
        "label": {
          "en": "Apparent_temperature_height_above_ground"
        }
      },
      "measurementType": {
        "method":"instantaneous"
      },
      "extent": {
        "horizontal": {
          "name": [
            "longitude",
            "latutude"
          ],
          "coordinates": [
            "x",
            "y"
          ],
          "geographic": "BBOX[359.875,90.125,-0.125,-90.125]"
        },
        "vertical": {
          "name": [
            "height_above_ground"
          ],
          "coordinates": [
            "z"
          ],
          "range": [
            "2.0"
          ]
        },
        "temporal": {
          "name": [
            "time"
          ],
          "coordinates": [
            "time"
```

```
          ],
          "range": [
            "2020-02-02T00:00:00Z",
            "2020-02-02T03:00:00Z",
            "2020-02-02T06:00:00Z",
            "2020-02-02T09:00:00Z",
            "2020-02-02T12:00:00Z",
            "2020-02-02T15:00:00Z",
            "2020-02-02T18:00:00Z",
            "2020-02-02T21:00:00Z",
            "2020-02-03T00:00:00Z",
            "2020-02-03T03:00:00Z",
            "2020-02-03T06:00:00Z",
            "2020-02-03T09:00:00Z",
            "2020-02-03T12:00:00Z",
            "2020-02-03T15:00:00Z",
            "2020-02-03T18:00:00Z",
            "2020-02-03T21:00:00Z",
            "2020-02-04T00:00:00Z",
            "2020-02-04T03:00:00Z",
            "2020-02-04T06:00:00Z",
            "2020-02-04T09:00:00Z",
            "2020-02-04T12:00:00Z",
            "2020-02-04T15:00:00Z",
            "2020-02-04T18:00:00Z",
            "2020-02-04T21:00:00Z",
            "2020-02-05T00:00:00Z",
            "2020-02-05T03:00:00Z",
            "2020-02-05T06:00:00Z",
            "2020-02-05T09:00:00Z",
            "2020-02-05T12:00:00Z",
            "2020-02-05T15:00:00Z",
            "2020-02-05T18:00:00Z",
            "2020-02-05T21:00:00Z",
            "2020-02-06T00:00:00Z",
            "2020-02-06T03:00:00Z",
            "2020-02-06T06:00:00Z",
            "2020-02-06T09:00:00Z",
            "2020-02-06T12:00:00Z",
            "2020-02-06T15:00:00Z",
            "2020-02-06T18:00:00Z",
            "2020-02-06T21:00:00Z",
            "2020-02-07T00:00:00Z",
            "2020-02-07T03:00:00Z",
            "2020-02-07T06:00:00Z",
            "2020-02-07T09:00:00Z",
            "2020-02-07T12:00:00Z",
            "2020-02-07T15:00:00Z",
            "2020-02-07T18:00:00Z",
            "2020-02-07T21:00:00Z",
            "2020-02-08T00:00:00Z",
```

```
          "2020-02-08T03:00:00Z",
          "2020-02-08T06:00:00Z",
          "2020-02-08T09:00:00Z",
          "2020-02-08T12:00:00Z",
          "2020-02-08T15:00:00Z",
          "2020-02-08T18:00:00Z",
          "2020-02-08T21:00:00Z",
          "2020-02-09T00:00:00Z",
          "2020-02-09T03:00:00Z",
          "2020-02-09T06:00:00Z",
          "2020-02-09T09:00:00Z",
          "2020-02-09T12:00:00Z",
          "2020-02-09T15:00:00Z",
          "2020-02-09T18:00:00Z",
          "2020-02-09T21:00:00Z",
          "2020-02-10T00:00:00Z",
          "2020-02-10T03:00:00Z",
          "2020-02-10T06:00:00Z",
          "2020-02-10T09:00:00Z",
          "2020-02-10T12:00:00Z",
          "2020-02-10T15:00:00Z",
          "2020-02-10T18:00:00Z",
          "2020-02-10T21:00:00Z",
          "2020-02-11T00:00:00Z",
          "2020-02-11T03:00:00Z",
          "2020-02-11T06:00:00Z",
          "2020-02-11T09:00:00Z",
          "2020-02-11T12:00:00Z",
          "2020-02-11T15:00:00Z",
          "2020-02-11T18:00:00Z",
          "2020-02-11T21:00:00Z",
          "2020-02-12T00:00:00Z",
          "2020-02-12T12:00:00Z",
          "2020-02-13T00:00:00Z",
          "2020-02-13T12:00:00Z",
          "2020-02-14T00:00:00Z",
          "2020-02-14T12:00:00Z",
          "2020-02-15T00:00:00Z",
          "2020-02-15T12:00:00Z",
          "2020-02-16T00:00:00Z",
          "2020-02-16T12:00:00Z",
          "2020-02-17T00:00:00Z",
          "2020-02-17T12:00:00Z",
          "2020-02-18T00:00:00Z"
        ]
      }
    }
  },
  "Dewpoint_temperature_height_above_ground": {
    "description": {
      "en": "Dewpoint_temperature_height_above_ground"
```

```
      },
      "unit": {
        "label": {
          "en": "kelvin"
        },
        "symbol": {
          "value": "K",
          "type": "http://data.example.org/metadata/uom/UCUM/K"
        }
      },
      "observedProperty": {
        "id": "http://data.example.org/metadata/grib2/codeflag/4.2-0-0-6",
        "label": {
          "en": "Dewpoint_temperature_height_above_ground"
        }
      },
      "measurementType": {
        "method":"instantaneous"
      },
      "extent": {
        "horizontal": {
          "name": [
            "longitude",
            "latutude"
          ],
          "coordinates": [
            "x",
            "y"
          ],
          "geographic": "BBOX[359.875,90.125,-0.125,-90.125]"
        },
        "vertical": {
          "name": [
            "height_above_ground"
          ],
          "coordinates": [
            "z"
          ],
          "range": [
            "2.0"
          ]
        },
        "temporal": {
          "name": [
            "time"
          ],
          "coordinates": [
            "time"
          ],
          "range": [
            "2020-02-02T00:00:00Z",
```

```
            "2020-02-02T03:00:00Z",
            "2020-02-02T06:00:00Z",
            "2020-02-02T09:00:00Z",
            "2020-02-02T12:00:00Z",
            "2020-02-02T15:00:00Z",
            "2020-02-02T18:00:00Z",
            "2020-02-02T21:00:00Z",
            "2020-02-03T00:00:00Z",
            "2020-02-03T03:00:00Z",
            "2020-02-03T06:00:00Z",
            "2020-02-03T09:00:00Z",
            "2020-02-03T12:00:00Z",
            "2020-02-03T15:00:00Z",
            "2020-02-03T18:00:00Z",
            "2020-02-03T21:00:00Z",
            "2020-02-04T00:00:00Z",
            "2020-02-04T03:00:00Z",
            "2020-02-04T06:00:00Z",
            "2020-02-04T09:00:00Z",
            "2020-02-04T12:00:00Z",
            "2020-02-04T15:00:00Z",
            "2020-02-04T18:00:00Z",
            "2020-02-04T21:00:00Z",
            "2020-02-05T00:00:00Z",
            "2020-02-05T03:00:00Z",
            "2020-02-05T06:00:00Z",
            "2020-02-05T09:00:00Z",
            "2020-02-05T12:00:00Z",
            "2020-02-05T15:00:00Z",
            "2020-02-05T18:00:00Z",
            "2020-02-05T21:00:00Z",
            "2020-02-06T00:00:00Z",
            "2020-02-06T03:00:00Z",
            "2020-02-06T06:00:00Z",
            "2020-02-06T09:00:00Z",
            "2020-02-06T12:00:00Z",
            "2020-02-06T15:00:00Z",
            "2020-02-06T18:00:00Z",
            "2020-02-06T21:00:00Z",
            "2020-02-07T00:00:00Z",
            "2020-02-07T03:00:00Z",
            "2020-02-07T06:00:00Z",
            "2020-02-07T09:00:00Z",
            "2020-02-07T12:00:00Z",
            "2020-02-07T15:00:00Z",
            "2020-02-07T18:00:00Z",
            "2020-02-07T21:00:00Z",
            "2020-02-08T00:00:00Z",
            "2020-02-08T03:00:00Z",
            "2020-02-08T06:00:00Z",
            "2020-02-08T09:00:00Z",
```

```
            "2020-02-08T12:00:00Z",
            "2020-02-08T15:00:00Z",
            "2020-02-08T18:00:00Z",
            "2020-02-08T21:00:00Z",
            "2020-02-09T00:00:00Z",
            "2020-02-09T03:00:00Z",
            "2020-02-09T06:00:00Z",
            "2020-02-09T09:00:00Z",
            "2020-02-09T12:00:00Z",
            "2020-02-09T15:00:00Z",
            "2020-02-09T18:00:00Z",
            "2020-02-09T21:00:00Z",
            "2020-02-10T00:00:00Z",
            "2020-02-10T03:00:00Z",
            "2020-02-10T06:00:00Z",
            "2020-02-10T09:00:00Z",
            "2020-02-10T12:00:00Z",
            "2020-02-10T15:00:00Z",
            "2020-02-10T18:00:00Z",
            "2020-02-10T21:00:00Z",
            "2020-02-11T00:00:00Z",
            "2020-02-11T03:00:00Z",
            "2020-02-11T06:00:00Z",
            "2020-02-11T09:00:00Z",
            "2020-02-11T12:00:00Z",
            "2020-02-11T15:00:00Z",
            "2020-02-11T18:00:00Z",
            "2020-02-11T21:00:00Z",
            "2020-02-12T00:00:00Z",
            "2020-02-12T12:00:00Z",
            "2020-02-13T00:00:00Z",
            "2020-02-13T12:00:00Z",
            "2020-02-14T00:00:00Z",
            "2020-02-14T12:00:00Z",
            "2020-02-15T00:00:00Z",
            "2020-02-15T12:00:00Z",
            "2020-02-16T00:00:00Z",
            "2020-02-16T12:00:00Z",
            "2020-02-17T00:00:00Z",
            "2020-02-17T12:00:00Z",
            "2020-02-18T00:00:00Z"
          ]
        }
      }
    },
    "Relative_humidity_height_above_ground": {
      "description": {
        "en": "Relative_humidity_height_above_ground"
      },
      "unit": {
        "label": {
```

```
        "en": "percent"
      },
      "symbol": {
        "value": "%",
        "type": "http://data.example.org/metadata/uom/UCUM/%"
      }
    },
    "observedProperty": {
      "id": "http://data.example.org/metadata/grib2/codeflag/4.2-0-1-1",
      "label": {
        "en": "Relative_humidity_height_above_ground"
      }
    },
    "measurementType": {
      "method":"instantaneous"
    },
    "extent": {
      "horizontal": {
        "name": [
          "longitude",
          "latutude"
        ],
        "coordinates": [
          "x",
          "y"
        ],
        "geographic": "BBOX[359.875,90.125,-0.125,-90.125]"
      },
      "vertical": {
        "name": [
          "height_above_ground"
        ],
        "coordinates": [
          "z"
        ],
        "range": [
          "2.0"
        ]
      },
      "temporal": {
        "name": [
          "time"
        ],
        "coordinates": [
          "time"
        ],
        "range": [
          "2020-02-02T00:00:00Z",
          "2020-02-02T03:00:00Z",
          "2020-02-02T06:00:00Z",
          "2020-02-02T09:00:00Z",
```

```
            "2020-02-02T12:00:00Z",
            "2020-02-02T15:00:00Z",
            "2020-02-02T18:00:00Z",
            "2020-02-02T21:00:00Z",
            "2020-02-03T00:00:00Z",
            "2020-02-03T03:00:00Z",
            "2020-02-03T06:00:00Z",
            "2020-02-03T09:00:00Z",
            "2020-02-03T12:00:00Z",
            "2020-02-03T15:00:00Z",
            "2020-02-03T18:00:00Z",
            "2020-02-03T21:00:00Z",
            "2020-02-04T00:00:00Z",
            "2020-02-04T03:00:00Z",
            "2020-02-04T06:00:00Z",
            "2020-02-04T09:00:00Z",
            "2020-02-04T12:00:00Z",
            "2020-02-04T15:00:00Z",
            "2020-02-04T18:00:00Z",
            "2020-02-04T21:00:00Z",
            "2020-02-05T00:00:00Z",
            "2020-02-05T03:00:00Z",
            "2020-02-05T06:00:00Z",
            "2020-02-05T09:00:00Z",
            "2020-02-05T12:00:00Z",
            "2020-02-05T15:00:00Z",
            "2020-02-05T18:00:00Z",
            "2020-02-05T21:00:00Z",
            "2020-02-06T00:00:00Z",
            "2020-02-06T03:00:00Z",
            "2020-02-06T06:00:00Z",
            "2020-02-06T09:00:00Z",
            "2020-02-06T12:00:00Z",
            "2020-02-06T15:00:00Z",
            "2020-02-06T18:00:00Z",
            "2020-02-06T21:00:00Z",
            "2020-02-07T00:00:00Z",
            "2020-02-07T03:00:00Z",
            "2020-02-07T06:00:00Z",
            "2020-02-07T09:00:00Z",
            "2020-02-07T12:00:00Z",
            "2020-02-07T15:00:00Z",
            "2020-02-07T18:00:00Z",
            "2020-02-07T21:00:00Z",
            "2020-02-08T00:00:00Z",
            "2020-02-08T03:00:00Z",
            "2020-02-08T06:00:00Z",
            "2020-02-08T09:00:00Z",
            "2020-02-08T12:00:00Z",
            "2020-02-08T15:00:00Z",
            "2020-02-08T18:00:00Z",
```

```
            "2020-02-08T21:00:00Z",
            "2020-02-09T00:00:00Z",
            "2020-02-09T03:00:00Z",
            "2020-02-09T06:00:00Z",
            "2020-02-09T09:00:00Z",
            "2020-02-09T12:00:00Z",
            "2020-02-09T15:00:00Z",
            "2020-02-09T18:00:00Z",
            "2020-02-09T21:00:00Z",
            "2020-02-10T00:00:00Z",
            "2020-02-10T03:00:00Z",
            "2020-02-10T06:00:00Z",
            "2020-02-10T09:00:00Z",
            "2020-02-10T12:00:00Z",
            "2020-02-10T15:00:00Z",
            "2020-02-10T18:00:00Z",
            "2020-02-10T21:00:00Z",
            "2020-02-11T00:00:00Z",
            "2020-02-11T03:00:00Z",
            "2020-02-11T06:00:00Z",
            "2020-02-11T09:00:00Z",
            "2020-02-11T12:00:00Z",
            "2020-02-11T15:00:00Z",
            "2020-02-11T18:00:00Z",
            "2020-02-11T21:00:00Z",
            "2020-02-12T00:00:00Z",
            "2020-02-12T12:00:00Z",
            "2020-02-13T00:00:00Z",
            "2020-02-13T12:00:00Z",
            "2020-02-14T00:00:00Z",
            "2020-02-14T12:00:00Z",
            "2020-02-15T00:00:00Z",
            "2020-02-15T12:00:00Z",
            "2020-02-16T00:00:00Z",
            "2020-02-16T12:00:00Z",
            "2020-02-17T00:00:00Z",
            "2020-02-17T12:00:00Z",
            "2020-02-18T00:00:00Z"
          ]
        }
      }
    }
  },
  "outputCRS": [
    {
      "id": "EPSG:4326",
      "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\",0.017453
2925199433,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",\"4326\"]]"
    }
```

```
  ],
  "interpolation": [
    "nearest_neighbour",
    "linear"
  ],
  "interpolationZ": [
    "cubic_spline"
  ],
  "outputFormat": [
    "CoverageJSON"
  ],
  "instanceAxes": {
    "x": {
      "label": "Longitude",
      "lowerBound": -180,
      "upperBound": 180,
      "uomLabel": "degrees"
    },
    "y": {
      "label": "Latitude",
      "lowerBound": -89.9,
      "upperBound": 89.9,
      "uomLabel": "degrees"
    },
    "z": {
      "label": null,
      "lowerBound": 2,
      "upperBound": 2,
      "uomLabel": null
    },
    "t": {
      "label": "Time",
      "lowerBound": "2020-02-02T00:00:00Z",
      "upperBound": "2020-02-18T00:00:00Z",
      "uomLabel": "ISO8601"
    },
    "attributes": {
      "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\",0.017453
29251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",\"4326\"]]",
      "proj4": "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"
    }
  },
  "name": "gfs_025_time-height_above_ground-lat-lon",
  "instance": "latest"
}
```

B.8. Area Query Metadata Examples

Example 41. Collection instance metadata response document

This metadata example response in JSON is for a the data instance of the "DEM" Collection. It includes links to the collection resource in all formats that are supported by the API (link relation type: "items").

There is a link to the feature collections response itself (link relation type: "self").

Representations of this resource in other formats are referenced using link relation type "alternate".

An additional link is to a GML application schema for the dataset - using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "describedBy".

A bulk download of all the features in the dataset is referenced using link relation type "enclosure"

Finally there are also links to the license information for the building data (using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "license").

Reference system information is not provided as the service provides geometries only in the default system (spatial: WGS 84 longitude/latitude; temporal: Gregorian calendar).


```
{
  "links": [
    {
      "href":
"http://data.example.org/collections/dem/area?outputFormat=application%2Fjson",
      "rel": "self",
      "type": "application/json",
      "title": "Area metadata document as json"
    },
    {
      "href":
"http://data.example.org/collections/dem/area?outputFormat=application%2Fx-yaml",
      "rel": "alternate",
      "type": "application/x-yaml",
      "title": "Area metadata document as yaml"
    },
    {
      "href":
"http://data.example.org/collections/dem/area?outputFormat=text%2Fxml",
      "rel": "alternate",
      "type": "text/xml",
      "title": "Area metadata document as xml"
    },
```

```
    {
      "href":
"http://data.example.org/collections/dem/area?outputFormat=text%2Fhtml",
      "rel": "alternate",
      "type": "text/html",
      "title": "Area metadata document as html"
    }
  ],
  "id": "dem_data",
  "title": "DEM height values",
  "description": "DEM height values",
  "parameters": {
    "height": {
      "description": {
        "en": "Height"
      },
      "unit": {
        "label": {
          "en": "m"
        },
        "symbol": {
          "value": "m,
          "type": "http://data.example.org/metadata/uom/UCUM/m"
        }
      },
      "observedProperty": {
        "id": "",
        "label": {
          "en": ""
        }
      },
      "extent": {
        "horizontal": {
          "name": [
            "longitude",
            "latutude"
          ],
          "coordinates": [
            "x",
            "y"
          ],
          "geographic": "BBOX[-180.0,-89.9,180.0,89.9]"
        }
      }
    }
  },
  "outputCRS": [
    {
      "id": "EPSG:4326",
      "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\",0.017453
2925199433,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",\"4326\"]]"
    }
  ],
  "interpolationX": [
    "Bilinear",
    "Bicubic"
  ],
  "interpolationY": [
    "Bilinear",
    "Bicubic"
  ],
  "outputFormat": [
    "CoverageJSON"
  ],
  "instanceAxes": {
    "x": {
      "label": "Longitude",
      "lowerBound": -180,
      "upperBound": 180,
      "uomLabel": "degrees"
    },
    "y": {
      "label": "Latitude",
      "lowerBound": -89.9,
      "upperBound": 89.9,
      "uomLabel": "degrees"
    },
    "attributes": {
      "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\",0.017453
29251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",\"4326\"]]",
      "proj4": "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"
    }
  },
  "name": "dem",
  "instance": "data"
}
```


B.9. Location Query Metadata Examples

Example 42. Collection instance metadata response document

This metadata example response in JSON is for a the Raw instance of the "Metar" Collection. It includes links to the collection resource in all formats that are supported by the API (link relation type: "items").

There is a link to the feature collections response itself (link relation type: "self").

Representations of this resource in other formats are referenced using link relation type "alternate".

An additional link is to a GML application schema for the dataset - using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "describedBy".

A bulk download of all the features in the dataset is referenced using link relation type "enclosure"

Finally there are also links to the license information for the building data (using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "license").

Reference system information is not provided as the service provides geometries only in the default system (spatial: WGS 84 longitude/latitude; temporal: Gregorian calendar).


```
{
  "links": [
    {
      "href":
"http://data.example.org/collections/metar/raw?outputFormat=application%2Fjson",
      "rel": "self",
      "type": "application/json",
      "title": "raw document as json"
    },
    {
      "href":
"http://data.example.org/collections/metar/location?outputFormat=application%2Fx-
yaml",
      "rel": "alternate",
      "type": "application/x-yaml",
      "title": "Metar location metadata document as yaml"
    },
    {
      "href":
"http://data.example.org/collections/metar/location?outputFormat=text%2Fxml",
      "rel": "alternate",
      "type": "text/xml",
      "title": "Metar location metadata document as  xml"
    },
    {
      "href":
"http://data.example.org/collections/metar/location?outputFormat=text%2Fhtml",
      "rel": "alternate",
      "type": "text/html",
      "title": "Metar location metadata document as html"
    },
    {
      "href": "http://data.example.org/collections/metar/position",
```

```
      "rel": "self",
      "type": "application/json",
      "title": "Metar location metadata document as JSON"
    }
  ],
  "id": "metar",
  "title": "Metar observations",
  "description": "Last 36 Hours of Global Metar observations",
  "parameters": {
    "air_temperature": {
      "description": {
        "en": "Temperature"
      },
      "unit": {
        "label": {
          "en": "degree Celsius"
        },
        "symbol": {
          "value": "°C",
          "type": "http://data.example.org/metadata/uom/UCUM/Cel"
        }
      },
      "observedProperty": {
        "id": "http://data.example.org/metadata/bufr4/d/307045-012023",
        "label": {
          "en": "Temperature"
        }
      },
      "measurementType": {
        "method":"instantaneous"
      },
      "extent": {
        "horizontal": {
          "name": [
            "longitude",
            "latutude"
          ],
          "coordinates": [
            "x",
            "y"
          ],
          "geographic": "BBOX[-180.0,-89.9,180.0,89.9]"
        },
        "temporal": {
          "name": [
            "time"
          ],
          "coordinates": [
            "time"
          ],
          "range": [
```

```
            "2020-01-31T22:00:00Z",
            "2020-01-31T23:00:00Z",
            "2020-02-01T00:00:00Z",
            "2020-02-01T01:00:00Z",
            "2020-02-01T02:00:00Z",
            "2020-02-01T03:00:00Z",
            "2020-02-01T04:00:00Z",
            "2020-02-01T05:00:00Z",
            "2020-02-01T06:00:00Z",
            "2020-02-01T07:00:00Z",
            "2020-02-01T08:00:00Z",
            "2020-02-01T09:00:00Z",
            "2020-02-01T10:00:00Z",
            "2020-02-01T11:00:00Z",
            "2020-02-01T12:00:00Z",
            "2020-02-01T13:00:00Z",
            "2020-02-01T14:00:00Z",
            "2020-02-01T15:00:00Z",
            "2020-02-01T16:00:00Z",
            "2020-02-01T17:00:00Z",
            "2020-02-01T18:00:00Z",
            "2020-02-01T19:00:00Z",
            "2020-02-01T20:00:00Z",
            "2020-02-01T21:00:00Z",
            "2020-02-01T22:00:00Z",
            "2020-02-01T23:00:00Z",
            "2020-02-02T00:00:00Z",
            "2020-02-02T01:00:00Z",
            "2020-02-02T02:00:00Z",
            "2020-02-02T03:00:00Z",
            "2020-02-02T04:00:00Z",
            "2020-02-02T05:00:00Z",
            "2020-02-02T06:00:00Z",
            "2020-02-02T07:00:00Z",
            "2020-02-02T08:00:00Z"
          ]
        }
      }
    },
    "dewpoint_temperature": {
      "description": {
        "en": "Dewpoint temperature"
      },
      "unit": {
        "label": {
          "en": "degree Celsius"
        },
        "symbol": {
          "value": "°C",
          "type": "http://data.example.org/metadata/uom/UCUM/Cel"
        }
```

```
      },
      "observedProperty": {
        "id": "http://data.example.org/metadata/bufr4/d/307045-012024",
        "label": {
          "en": "Dewpoint temperature"
        }
      },
      "measurementType": {
        "method":"instantaneous"
      },
      "extent": {
        "horizontal": {
          "name": [
            "longitude",
            "latutude"
          ],
          "coordinates": [
            "x",
            "y"
          ],
          "geographic": "BBOX[-180.0,-89.9,180.0,89.9]"
        },
        "temporal": {
          "name": [
            "time"
          ],
          "coordinates": [
            "time"
          ],
          "range": [
            "2020-01-31T22:00:00Z",
            "2020-01-31T23:00:00Z",
            "2020-02-01T00:00:00Z",
            "2020-02-01T01:00:00Z",
            "2020-02-01T02:00:00Z",
            "2020-02-01T03:00:00Z",
            "2020-02-01T04:00:00Z",
            "2020-02-01T05:00:00Z",
            "2020-02-01T06:00:00Z",
            "2020-02-01T07:00:00Z",
            "2020-02-01T08:00:00Z",
            "2020-02-01T09:00:00Z",
            "2020-02-01T10:00:00Z",
            "2020-02-01T11:00:00Z",
            "2020-02-01T12:00:00Z",
            "2020-02-01T13:00:00Z",
            "2020-02-01T14:00:00Z",
            "2020-02-01T15:00:00Z",
            "2020-02-01T16:00:00Z",
            "2020-02-01T17:00:00Z",
            "2020-02-01T18:00:00Z",
```

```
            "2020-02-01T19:00:00Z",
            "2020-02-01T20:00:00Z",
            "2020-02-01T21:00:00Z",
            "2020-02-01T22:00:00Z",
            "2020-02-01T23:00:00Z",
            "2020-02-02T00:00:00Z",
            "2020-02-02T01:00:00Z",
            "2020-02-02T02:00:00Z",
            "2020-02-02T03:00:00Z",
            "2020-02-02T04:00:00Z",
            "2020-02-02T05:00:00Z",
            "2020-02-02T06:00:00Z",
            "2020-02-02T07:00:00Z",
            "2020-02-02T08:00:00Z"
          ]
        }
      }
    },
    "wind_speed": {
      "description": {
        "en": "Wind speed"
      },
      "unit": {
        "label": {
          "en": "unknown"
        },
        "symbol": {
          "value": "m/s",
          "type": ""
        }
      },
      "observedProperty": {
        "id": "http://data.example.org/metadata/bufr4/d/307045-011002",
        "label": {
          "en": "Wind speed"
        }
      },
      "measurementType": {
        "method":"mean",
        "period": "PT10M"
      },
      "extent": {
        "horizontal": {
          "name": [
            "longitude",
            "latutude"
          ],
          "coordinates": [
            "x",
            "y"
          ],
```

```
          "geographic": "BBOX[-180.0,-89.9,180.0,89.9]"
        },
        "temporal": {
          "name": [
            "time"
          ],
          "coordinates": [
            "time"
          ],
          "range": [
            "2020-01-31T22:00:00Z",
            "2020-01-31T23:00:00Z",
            "2020-02-01T00:00:00Z",
            "2020-02-01T01:00:00Z",
            "2020-02-01T02:00:00Z",
            "2020-02-01T03:00:00Z",
            "2020-02-01T04:00:00Z",
            "2020-02-01T05:00:00Z",
            "2020-02-01T06:00:00Z",
            "2020-02-01T07:00:00Z",
            "2020-02-01T08:00:00Z",
            "2020-02-01T09:00:00Z",
            "2020-02-01T10:00:00Z",
            "2020-02-01T11:00:00Z",
            "2020-02-01T12:00:00Z",
            "2020-02-01T13:00:00Z",
            "2020-02-01T14:00:00Z",
            "2020-02-01T15:00:00Z",
            "2020-02-01T16:00:00Z",
            "2020-02-01T17:00:00Z",
            "2020-02-01T18:00:00Z",
            "2020-02-01T19:00:00Z",
            "2020-02-01T20:00:00Z",
            "2020-02-01T21:00:00Z",
            "2020-02-01T22:00:00Z",
            "2020-02-01T23:00:00Z",
            "2020-02-02T00:00:00Z",
            "2020-02-02T01:00:00Z",
            "2020-02-02T02:00:00Z",
            "2020-02-02T03:00:00Z",
            "2020-02-02T04:00:00Z",
            "2020-02-02T05:00:00Z",
            "2020-02-02T06:00:00Z",
            "2020-02-02T07:00:00Z",
            "2020-02-02T08:00:00Z"
          ]
        }
      }
    },
    "wind-gust": {
      "description": {
```

```
        "en": "Maximum wind gust speed"
      },
      "unit": {
        "label": {
          "en": "unknown"
        },
        "symbol": {
          "value": "m/s",
          "type": ""
        }
      },
      "observedProperty": {
        "id": "http://data.example.org/metadata/bufr4/d/307045-011041",
        "label": {
          "en": "Maximum wind gust speed"
        }
      },
      "measurementType": {
        "method":"mean",
        "period": "PT10M"
      },
      "extent": {
        "horizontal": {
          "name": [
            "longitude",
            "latutude"
          ],
          "coordinates": [
            "x",
            "y"
          ],
          "geographic": "BBOX[-180.0,-89.9,180.0,89.9]"
        },
        "temporal": {
          "name": [
            "time"
          ],
          "coordinates": [
            "time"
          ],
          "range": [
            "2020-01-31T22:00:00Z",
            "2020-01-31T23:00:00Z",
            "2020-02-01T00:00:00Z",
            "2020-02-01T01:00:00Z",
            "2020-02-01T02:00:00Z",
            "2020-02-01T03:00:00Z",
            "2020-02-01T04:00:00Z",
            "2020-02-01T05:00:00Z",
            "2020-02-01T06:00:00Z",
            "2020-02-01T07:00:00Z",
```

```
            "2020-02-01T08:00:00Z",
            "2020-02-01T09:00:00Z",
            "2020-02-01T10:00:00Z",
            "2020-02-01T11:00:00Z",
            "2020-02-01T12:00:00Z",
            "2020-02-01T13:00:00Z",
            "2020-02-01T14:00:00Z",
            "2020-02-01T15:00:00Z",
            "2020-02-01T16:00:00Z",
            "2020-02-01T17:00:00Z",
            "2020-02-01T18:00:00Z",
            "2020-02-01T19:00:00Z",
            "2020-02-01T20:00:00Z",
            "2020-02-01T21:00:00Z",
            "2020-02-01T22:00:00Z",
            "2020-02-01T23:00:00Z",
            "2020-02-02T00:00:00Z",
            "2020-02-02T01:00:00Z",
            "2020-02-02T02:00:00Z",
            "2020-02-02T03:00:00Z",
            "2020-02-02T04:00:00Z",
            "2020-02-02T05:00:00Z",
            "2020-02-02T06:00:00Z",
            "2020-02-02T07:00:00Z",
            "2020-02-02T08:00:00Z"
          ]
        }
      }
    },
    "wind_from_direction": {
      "description": {
        "en": "Wind direction"
      },
      "unit": {
        "label": {
          "en": "unknown"
        },
        "symbol": {
          "value": "degree true",
          "type": ""
        }
      },
      "observedProperty": {
        "id": "http://data.example.org/metadata/bufr4/d/307045-011001",
        "label": {
          "en": "Wind direction"
        }
      },
      "measurementType": {
        "method":"mean",
        "period": "PT10M"
```

```
      },
      "extent": {
        "horizontal": {
          "name": [
            "longitude",
            "latutude"
          ],
          "coordinates": [
            "x",
            "y"
          ],
          "geographic": "BBOX[-180.0,-89.9,180.0,89.9]"
        },
        "temporal": {
          "name": [
            "time"
          ],
          "coordinates": [
            "time"
          ],
          "range": [
            "2020-01-31T22:00:00Z",
            "2020-01-31T23:00:00Z",
            "2020-02-01T00:00:00Z",
            "2020-02-01T01:00:00Z",
            "2020-02-01T02:00:00Z",
            "2020-02-01T03:00:00Z",
            "2020-02-01T04:00:00Z",
            "2020-02-01T05:00:00Z",
            "2020-02-01T06:00:00Z",
            "2020-02-01T07:00:00Z",
            "2020-02-01T08:00:00Z",
            "2020-02-01T09:00:00Z",
            "2020-02-01T10:00:00Z",
            "2020-02-01T11:00:00Z",
            "2020-02-01T12:00:00Z",
            "2020-02-01T13:00:00Z",
            "2020-02-01T14:00:00Z",
            "2020-02-01T15:00:00Z",
            "2020-02-01T16:00:00Z",
            "2020-02-01T17:00:00Z",
            "2020-02-01T18:00:00Z",
            "2020-02-01T19:00:00Z",
            "2020-02-01T20:00:00Z",
            "2020-02-01T21:00:00Z",
            "2020-02-01T22:00:00Z",
            "2020-02-01T23:00:00Z",
            "2020-02-02T00:00:00Z",
            "2020-02-02T01:00:00Z",
            "2020-02-02T02:00:00Z",
            "2020-02-02T03:00:00Z",
```

```
            "2020-02-02T04:00:00Z",
            "2020-02-02T05:00:00Z",
            "2020-02-02T06:00:00Z",
            "2020-02-02T07:00:00Z",
            "2020-02-02T08:00:00Z"
          ]
        }
      }
    },
    "visibility": {
      "description": {
        "en": "METAR/SPECI visibility"
      },
      "unit": {
        "label": {
          "en": ""
        },
        "symbol": {
          "value": "",
          "type": ""
        }
      },
      "observedProperty": {
        "id": "http://data.example.org/metadata/bufr4/d/307051-307046",
        "label": {
          "en": "METAR/SPECI visibility"
        }
      },
      "measurementType": {
        "method":"instantaneous"
      },
      "extent": {
        "horizontal": {
          "name": [
            "longitude",
            "latutude"
          ],
          "coordinates": [
            "x",
            "y"
          ],
          "geographic": "BBOX[-180.0,-89.9,180.0,89.9]"
        },
        "temporal": {
          "name": [
            "time"
          ],
          "coordinates": [
            "time"
          ],
          "range": [
```

```
            "2020-01-31T22:00:00Z",
            "2020-01-31T23:00:00Z",
            "2020-02-01T00:00:00Z",
            "2020-02-01T01:00:00Z",
            "2020-02-01T02:00:00Z",
            "2020-02-01T03:00:00Z",
            "2020-02-01T04:00:00Z",
            "2020-02-01T05:00:00Z",
            "2020-02-01T06:00:00Z",
            "2020-02-01T07:00:00Z",
            "2020-02-01T08:00:00Z",
            "2020-02-01T09:00:00Z",
            "2020-02-01T10:00:00Z",
            "2020-02-01T11:00:00Z",
            "2020-02-01T12:00:00Z",
            "2020-02-01T13:00:00Z",
            "2020-02-01T14:00:00Z",
            "2020-02-01T15:00:00Z",
            "2020-02-01T16:00:00Z",
            "2020-02-01T17:00:00Z",
            "2020-02-01T18:00:00Z",
            "2020-02-01T19:00:00Z",
            "2020-02-01T20:00:00Z",
            "2020-02-01T21:00:00Z",
            "2020-02-01T22:00:00Z",
            "2020-02-01T23:00:00Z",
            "2020-02-02T00:00:00Z",
            "2020-02-02T01:00:00Z",
            "2020-02-02T02:00:00Z",
            "2020-02-02T03:00:00Z",
            "2020-02-02T04:00:00Z",
            "2020-02-02T05:00:00Z",
            "2020-02-02T06:00:00Z",
            "2020-02-02T07:00:00Z",
            "2020-02-02T08:00:00Z"
          ]
        }
      }
    },
    "pressure": {
      "description": {
        "en": "Pressure"
      },
      "unit": {
        "label": {
          "en": "pascal"
        },
        "symbol": {
          "value": "Pa",
          "type": "http://data.example.org/metadata/uom/UCUM/Pa"
        }
```

```
      },
      "observedProperty": {
        "id": "http://data.example.org/metadata/bufr4/d/302031-007004",
        "label": {
          "en": "Pressure"
        }
      },
      "measurementType": {
        "method":"instantaneous"
      },
      "extent": {
        "horizontal": {
          "name": [
            "longitude",
            "latutude"
          ],
          "coordinates": [
            "x",
            "y"
          ],
          "geographic": "BBOX[-180.0,-89.9,180.0,89.9]"
        },
        "temporal": {
          "name": [
            "time"
          ],
          "coordinates": [
            "time"
          ],
          "range": [
            "2020-01-31T22:00:00Z",
            "2020-01-31T23:00:00Z",
            "2020-02-01T00:00:00Z",
            "2020-02-01T01:00:00Z",
            "2020-02-01T02:00:00Z",
            "2020-02-01T03:00:00Z",
            "2020-02-01T04:00:00Z",
            "2020-02-01T05:00:00Z",
            "2020-02-01T06:00:00Z",
            "2020-02-01T07:00:00Z",
            "2020-02-01T08:00:00Z",
            "2020-02-01T09:00:00Z",
            "2020-02-01T10:00:00Z",
            "2020-02-01T11:00:00Z",
            "2020-02-01T12:00:00Z",
            "2020-02-01T13:00:00Z",
            "2020-02-01T14:00:00Z",
            "2020-02-01T15:00:00Z",
            "2020-02-01T16:00:00Z",
            "2020-02-01T17:00:00Z",
            "2020-02-01T18:00:00Z",
```

```
            "2020-02-01T19:00:00Z",
            "2020-02-01T20:00:00Z",
            "2020-02-01T21:00:00Z",
            "2020-02-01T22:00:00Z",
            "2020-02-01T23:00:00Z",
            "2020-02-02T00:00:00Z",
            "2020-02-02T01:00:00Z",
            "2020-02-02T02:00:00Z",
            "2020-02-02T03:00:00Z",
            "2020-02-02T04:00:00Z",
            "2020-02-02T05:00:00Z",
            "2020-02-02T06:00:00Z",
            "2020-02-02T07:00:00Z",
            "2020-02-02T08:00:00Z"
          ]
        }
      }
    },
    "msl_pressure": {
      "description": {
        "en": "Pressure reduced to mean sea level"
      },
      "unit": {
        "label": {
          "en": "pascal"
        },
        "symbol": {
          "value": "Pa",
          "type": "http://data.example.org/metadata/uom/UCUM/Pa"
        }
      },
      "observedProperty": {
        "id": "http://data.example.org/metadata/bufr4/d/302051-010051",
        "label": {
          "en": "Pressure reduced to mean sea level"
        }
      },
      "measurementType": {
        "method":"instantaneous"
      },
      "extent": {
        "horizontal": {
          "name": [
            "longitude",
            "latutude"
          ],
          "coordinates": [
            "x",
            "y"
          ],
          "geographic": "BBOX[-180.0,-89.9,180.0,89.9]"
```

```
        },
        "temporal": {
          "name": [
            "time"
          ],
          "coordinates": [
            "time"
          ],
          "range": [
            "2020-01-31T22:00:00Z",
            "2020-01-31T23:00:00Z",
            "2020-02-01T00:00:00Z",
            "2020-02-01T01:00:00Z",
            "2020-02-01T02:00:00Z",
            "2020-02-01T03:00:00Z",
            "2020-02-01T04:00:00Z",
            "2020-02-01T05:00:00Z",
            "2020-02-01T06:00:00Z",
            "2020-02-01T07:00:00Z",
            "2020-02-01T08:00:00Z",
            "2020-02-01T09:00:00Z",
            "2020-02-01T10:00:00Z",
            "2020-02-01T11:00:00Z",
            "2020-02-01T12:00:00Z",
            "2020-02-01T13:00:00Z",
            "2020-02-01T14:00:00Z",
            "2020-02-01T15:00:00Z",
            "2020-02-01T16:00:00Z",
            "2020-02-01T17:00:00Z",
            "2020-02-01T18:00:00Z",
            "2020-02-01T19:00:00Z",
            "2020-02-01T20:00:00Z",
            "2020-02-01T21:00:00Z",
            "2020-02-01T22:00:00Z",
            "2020-02-01T23:00:00Z",
            "2020-02-02T00:00:00Z",
            "2020-02-02T01:00:00Z",
            "2020-02-02T02:00:00Z",
            "2020-02-02T03:00:00Z",
            "2020-02-02T04:00:00Z",
            "2020-02-02T05:00:00Z",
            "2020-02-02T06:00:00Z",
            "2020-02-02T07:00:00Z",
            "2020-02-02T08:00:00Z"
          ]
        }
      }
    },
    "raw_ob": {
      "type": "Parameter",
      "description": {
```

```
        "en": "raw ob"
      },
      "unit": {
        "label": {
          "en": ""
        },
        "symbol": {
          "value": "",
          "type":
"https://www.wmo.int/pages/prog/www/WMOCodes/WMO306_vI1/Publications/2014update/306_vol_I1_2014_en_track.pdf"
        }
      },
      "observedProperty": {
        "id": "Raw Metar Observation",
        "label": {
          "en": "Raw Metar Observation"
        }
      },
      "measurementType": {
        "method":"instantaneous"
      },
      "extent": {
        "horizontal": {
          "name": [
            "longitude",
            "latutude"
          ],
          "coordinates": [
            "x",
            "y"
          ],
          "geographic": "BBOX[-180.0,-89.9,180.0,89.9]"
        },
        "temporal": {
          "name": [
            "time"
          ],
          "coordinates": [
            "time"
          ],
          "range": [
            "2020-01-31T22:00:00Z",
            "2020-01-31T23:00:00Z",
            "2020-02-01T00:00:00Z",
            "2020-02-01T01:00:00Z",
            "2020-02-01T02:00:00Z",
            "2020-02-01T03:00:00Z",
            "2020-02-01T04:00:00Z",
            "2020-02-01T05:00:00Z",
            "2020-02-01T06:00:00Z",
```

```
            "2020-02-01T07:00:00Z",
            "2020-02-01T08:00:00Z",
            "2020-02-01T09:00:00Z",
            "2020-02-01T10:00:00Z",
            "2020-02-01T11:00:00Z",
            "2020-02-01T12:00:00Z",
            "2020-02-01T13:00:00Z",
            "2020-02-01T14:00:00Z",
            "2020-02-01T15:00:00Z",
            "2020-02-01T16:00:00Z",
            "2020-02-01T17:00:00Z",
            "2020-02-01T18:00:00Z",
            "2020-02-01T19:00:00Z",
            "2020-02-01T20:00:00Z",
            "2020-02-01T21:00:00Z",
            "2020-02-01T22:00:00Z",
            "2020-02-01T23:00:00Z",
            "2020-02-02T00:00:00Z",
            "2020-02-02T01:00:00Z",
            "2020-02-02T02:00:00Z",
            "2020-02-02T03:00:00Z",
            "2020-02-02T04:00:00Z",
            "2020-02-02T05:00:00Z",
            "2020-02-02T06:00:00Z",
            "2020-02-02T07:00:00Z",
            "2020-02-02T08:00:00Z"
          ]
        }
      }
    },
    "icao_id": {
      "type": "Parameter",
      "description": {
        "en": "icao id"
      },
      "unit": {
        "label": {
          "en": ""
        },
        "symbol": {
          "value": "",
          "type": "https://en.wikipedia.org/wiki/ICAO_airport_code"
        }
      },
      "observedProperty": {
        "id": "ICAO id",
        "label": {
          "en": "ICAO id"
        }
      },
      "measurementType": {
```

```
        "method":"instantaneous"
      },
      "extent": {
        "horizontal": {
          "name": [
            "longitude",
            "latutude"
          ],
          "coordinates": [
            "x",
            "y"
          ],
          "geographic": "BBOX[-180.0,-89.9,180.0,89.9]"
        },
        "temporal": {
          "name": [
            "time"
          ],
          "coordinates": [
            "time"
          ],
          "range": [
            "2020-01-31T22:00:00Z",
            "2020-01-31T23:00:00Z",
            "2020-02-01T00:00:00Z",
            "2020-02-01T01:00:00Z",
            "2020-02-01T02:00:00Z",
            "2020-02-01T03:00:00Z",
            "2020-02-01T04:00:00Z",
            "2020-02-01T05:00:00Z",
            "2020-02-01T06:00:00Z",
            "2020-02-01T07:00:00Z",
            "2020-02-01T08:00:00Z",
            "2020-02-01T09:00:00Z",
            "2020-02-01T10:00:00Z",
            "2020-02-01T11:00:00Z",
            "2020-02-01T12:00:00Z",
            "2020-02-01T13:00:00Z",
            "2020-02-01T14:00:00Z",
            "2020-02-01T15:00:00Z",
            "2020-02-01T16:00:00Z",
            "2020-02-01T17:00:00Z",
            "2020-02-01T18:00:00Z",
            "2020-02-01T19:00:00Z",
            "2020-02-01T20:00:00Z",
            "2020-02-01T21:00:00Z",
            "2020-02-01T22:00:00Z",
            "2020-02-01T23:00:00Z",
            "2020-02-02T00:00:00Z",
            "2020-02-02T01:00:00Z",
            "2020-02-02T02:00:00Z",
```

```
            "2020-02-02T03:00:00Z",
            "2020-02-02T04:00:00Z",
            "2020-02-02T05:00:00Z",
            "2020-02-02T06:00:00Z",
            "2020-02-02T07:00:00Z",
            "2020-02-02T08:00:00Z"
          ]
        }
      }
    }
  },
  "location_ids":[
    {"locID":"KIAD", "name": "WASH DC DULLES", "WKT":"POINT(-77.45 38.93 93)"},
    {"locID":"EGLL", "name": "LONDON HEATHROW", "WKT":"POINT(-0.45 50.48 24)"},
    {"locID":"LFPO", "name": "PARIS ORLY", "WKT":"POINT(2.38 28.58 96)"},
    {"locID":"FACT", "name": "CAPETOWN DF MALA", "WKT":"POINT(18.06 -33.97 42)"},
    {"locID":"VIDD", "name": "DELHI SAFDARJUNG", "WKT":"POINT(77.22 28.58 216)"},
    {"locID":"ZBAA", "name": "BEIJING", "WKT":"POINT(116.58 40.07 30)"},
    {"locID":"YSSY", "name": "SYDNEY INTL AIRP", "WKT":"POINT(150.17 -33.93 3)"}
  ]
  "outputCRS": [
    {
      "id": "EPSG:4326",
      "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\",0.017453
2925199433,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",\"4326\"]]"
    }
  ],
  "outputFormat": [
    "CoverageJSON"
  ],
  "instanceAxes": {
    "x": {
      "label": "Longitude",
      "lowerBound": -180,
      "upperBound": 180,
      "uomLabel": "degrees"
    },
    "y": {
      "label": "Latitude",
      "lowerBound": -89.9,
      "upperBound": 89.9,
      "uomLabel": "degrees"
    },
    "attributes": {
      "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\",0.017453
29251994328,AUTHORITY[\"EPSG\",\"9122\"]],AUTHORITY[\"EPSG\",\"4326\"]]",
      "proj4": "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"
```

```
    }
  },
  "name": "metar"
}
```

Annex C: Revision History

Date	Release	Editor	Primary clauses modified	Description
2019-10-31	October 2019 snapshot	C. Heazel	all	Baseline update

Annex D: Bibliography

- Open Geospatial Consortium: The Specification Model — A Standard for Modular specifications, OGC 08-131
- W3C/OGC: Spatial Data on the Web Best Practices, W3C Working Group Note 28 September 2017, https://www.w3.org/TR/sdw-bp/
- W3C: Data on the Web Best Practices, W3C Recommendation 31 January 2017, https://www.w3.org/TR/dwbp/
- W3C: Data Catalog Vocabulary, W3C Recommendation 16 January 2014, https://www.w3.org/TR/vocab-dcat/
- IANA: Link Relation Types, https://www.iana.org/assignments/link-relations/link-relations.xml