OGC® DOCUMENT: 20-024

External identifier of this OGC® document: https://www.opengis.net/doc/is/ogcapi-common-2/1.0

Open Geospatial Consortium

OGC API - COMMON - PART 2: GEOSPATIAL DATA

STANDARD
Implementation

DRAFT

Version: 1.0
Submission Date: 2029-03-30
Approval Date: 2029-03-30
Publication Date: 2029-03-30
Editor: Charles Heazel, Joan Masó, Jérôme Jacovella-St-Louis

Notice for Drafts: This document is not an OGC Standard. This document is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC Standard.
Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

License Agreement

Use of this document is subject to the license agreement at https://www.ogc.org/license

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

Copyright notice

Copyright © 2025 Open Geospatial Consortium
To obtain additional rights of use, visithttps://www.ogc.org/legal

Note

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

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

CONTENTS

I. ABSTRACT viii

II. KEYWORDS viii

III. SECURITY CONSIDERATIONS ix

IV. SUBMITTING ORGANIZATIONS x

V. SUBMITTERS x

1. SCOPE 2

2. CONFORMANCE 4
   2.1. Summary of conformance URIs 5

3. NORMATIVE REFERENCES 7

4. TERMS, DEFINITIONS AND ABBREVIATED TERMS 10
   4.26. Abbreviated terms 15

5. CONVENTIONS 17
   5.1. Identifiers 17
   5.2. Link relations 17
   5.3. Geometry 18
   5.4. Coordinate Reference Systems 19
   5.5. API definition 20

6. OVERVIEW 23
   6.1. Collections 23
   6.2. Data access mechanisms 23
   6.3. Representation 24
   6.4. UML Class Model 25

7. REQUIREMENTS CLASS “COLLECTIONS” 28
   7.1. List of Collections Resource (/collections) 29
   7.2. Collection Description Resource (/collections/{collectionId}) 32
   7.3. OGC API — Records compliance 39

8. REQUIREMENTS CLASS “UNIFORM MULTI-DIMENSION COLLECTION” 41
   8.1. Uniform description of all dimensions 41

8.2. Describing gridded dimensions 42

9. REQUIREMENTS CLASSES FOR ENCODINGS 47
9.1. Overview 47
9.2. Requirements Class “HTML” 47
9.3. Requirements Class “JSON” 49

10. MEDIA TYPES 52

ANNEX A (NORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE 54
A.1. Introduction 54
A.2. Conformance Class Collections 54
A.3. Conformance Class Uniform Multi-Dimension Collection 59
A.4. Conformance Class JSON 60
A.5. Conformance Class HTML 62

ANNEX B (INFORMATIVE) CLASS MODEL AS TABLES 64
B.1. OGC API — Common package 64

ANNEX C (INFORMATIVE) GLOSSARY 87

ANNEX D (NORMATIVE) BACKUS-NAUR FORMS 90
D.1. BNF for URI 90
D.2. BNF for Date-Time 91

ANNEX E (NORMATIVE) HTTP STATUS CODES 94

ANNEX F (INFORMATIVE) REVISION HISTORY 97

BIBLIOGRAPHY 99

LIST OF TABLES

Table 1 — Conformance class URIs 5
Table 2 — Link Relations 17
Table 3 — Collection Resources 29
Table 4 39
Table A.1 — Schema and Tests for Collections content 56
Table A.2 — Schema and Tests for Collection content 58
Table B.1 — Elements of “OGC API — Common::” () 64
Table B.2 — Elements of “OGC API — Common::OGC API — Common — Part 1” () 64
Table B.3 — Elements of “OGC API — Common::OGC API — Common — Part 2” () 65
Table B.4 — Elements of “OGC API — Common::OGC API — Records” () 65

Table B.5 — Elements of “OGC API — Common::Uniform Additional Dimensions” () 66
Table B.6 — Elements of “OGC API — Common::Address” () 66
Table B.7 — Elements of “OGC API — Common::BBox” (Array) 67
Table B.8 — Elements of “OGC API — Common::CollectionDescription” () 67
Table B.9 — Elements of “OGC API — Common::Collections” () 69
Table B.10 — Elements of “OGC API — Common::Concept” () 70
Table B.11 — Elements of “OGC API — Common::ConformanceClasses” () 70
Table B.12 — Elements of “OGC API — Common::Contact” () 71
Table B.13 — Elements of “OGC API — Common::Coordinates” (Array) 72
Table B.14 — Elements of “OGC API — Common::Email” () 72
Table B.15 — Elements of “OGC API — Common::Exception” () 73
Table B.16 — Elements of “OGC API — Common::Extent” () 73
Table B.17 — Elements of “OGC API — Common::Fields” (Array) 74
Table B.18 — Elements of “OGC API — Common::Format” () 74
Table B.19 — Elements of “OGC API — Common::Grid” (abstract) 75
Table B.20 — Elements of “OGC API — Common::Interval” (type) 75
Table B.21 — Elements of “OGC API — Common::IrregularGrid” () 76
Table B.22 — Elements of “OGC API — Common::LandingPage” () 76
Table B.23 — Elements of “OGC API — Common::Link” (type) 77
Table B.24 — Elements of “OGC API — Common::OtherDimension” () 78
Table B.25 — Elements of “OGC API — Common::Phone” () 79
Table B.26 — Elements of “OGC API — Common::RegularGrid” () 79
Table B.27 — Elements of “OGC API — Common::ResourceLanguage” () 80
Table B.28 — Elements of “OGC API — Common::TemporalExtent” () 81
Table B.29 — Elements of “OGC API — Common::SpatialExtent” () 82
Table B.30 — Elements of “OGC API — Common::Theme” () 83
Table B.31 — Definition table of “OGC API — Common::AccessConstraintsCode” (enumeration) 83
Table B.32 — Definition table of “OGC API — Common::AttributionMediaTypeCode” () 84
Table B.33 — Definition table of “OGC API — Common::LanguageDirectionCode” (enumeration) 84
Table B.34 — Definition table of “OGC API — Common::VariableType” () 85
Table E.1 — Typical HTTP status codes 94
Table F.1 97

LIST OF FIGURES

Figure 1 — UML class diagram describing the common resources 26

LIST OF NORMATIVE STATEMENTS

REQUIREMENTS CLASS 1: COLLECTIONS 28
REQUIREMENTS CLASS 2: UNIFORM MULTI-DIMENSION COLLECTION 41
REQUIREMENTS CLASS 3: HTML 47
REQUIREMENTS CLASS 4: JSON 49
REQUIREMENT 1 29
REQUIREMENT 2 29
REQUIREMENT 3 30
REQUIREMENT 4 31
REQUIREMENT 5 32
REQUIREMENT 6 33
REQUIREMENT 7 34
REQUIREMENT 8 36
REQUIREMENT 9 36
REQUIREMENT 10 42
REQUIREMENT 11 44
REQUIREMENT 12 48
REQUIREMENT 13 48
REQUIREMENT 14 49
REQUIREMENT 15 49
RECOMMENDATION 1 28
RECOMMENDATION 2 30
RECOMMENDATION 3 34
RECOMMENDATION 4 35
RECOMMENDATION 5 37
RECOMMENDATION 6 37
RECOMMENDATION 7 48
PERMISSION 1 37
PERMISSION 2 37
PERMISSION 3 45
CONFORMANCE CLASS A.1 54
CONFORMANCE CLASS A.2 59

CONFORMANCE CLASS A.3 61
CONFORMANCE CLASS A.4 62

I ABSTRACT

The OGC standards baseline has been extended to include Resource Oriented Architectures and Web APIs. In the course of developing OGC Web API standards, some practices proved to be common across multiple OGC Web API standards. These common practices are documented in the OGC API — Common Multi-Part Standard. OGC API — Common standards serve as reusable building-blocks. Standards developers can use these building-blocks in the construction of OGC Web API Standards. The result is a modular suite of coherent API standards which can be adapted by a system designer for the unique requirements of their system.

Spatial data is rarely considered as a single entity. Feature Collections, Coverages, Data Sets. They are all aggregations of Spatial or Temporal Resources. It stands to reason that an OGC Web API would also expose its holdings as aggregates of spatial resources.

The purpose of the OGC API — Common — Part 2: Geospatial Data (Common API-2: GeoData) Standard is to provide a means of organizing these collections and to define operations for listing and describing available collections.

OGC Common API-2: GeoData does not specify the nature of the geospatial data that make up a collection. Rather, the standard specifies a basic capability which should be applicable to any geospatial resource type or access mechanisms. Additional OGC Web API standards extend this foundation to define specific data access mechanisms.

This document defines the *OGC API — Common — Part 2: Geospatial Data* Standard. Suggested additions, changes and comments on this standard are welcome and encouraged. Such suggestions may be submitted as an issue on the *OGC API — Common* GitHub repository.

II KEYWORDS

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

ogcdoc, OGC document, geographic information, spatial data, spatial things, dataset, distribution, API, json, html, OpenAPI, REST, Common

III SECURITY CONSIDERATIONS

Clients should avoid blindly processing HTML markup code in the attribution property of collection descriptions. In particular, care should be taken to avoid running any `<script>` code, including inline event handlers (e.g., `onclick`) and dangerous href values like `javascript:`. All untrusted input should be sanitized to prevent script execution and mitigate XSS vulnerabilities.

See also the security considerations described in OGC API — Common — Part 1: Core.

IV SUBMITTING ORGANIZATIONS

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

- Heazeltech
- Ecere Corporation
- Universitat Autònoma de Barcelona (CREAF)
- Open Geospatial Consortium

V SUBMITTERS

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

Name	Affiliation
Chuck Heazel *(editor)*	Heazeltech
Jerome St-Louis *(editor)*	Ecere Corporation
Joan Masó (editor)	Universitat Autònoma de Barcelona (CREAF)
Chris Little	UK Met Office
Carl Reed	Self
Panagiotis (Peter) A. Vretanos	CubeWerx Inc.

1 SCOPE

1 SCOPE

The OGC API — Common Standard is a multi-part standard which defines a set of modules which can be used to build resource and mission-specific Web API specifications. The OGC API — Common — Part 2: Geospatial Data Standard (Common API-2: GeoData) is one of those modules.

Geospatial resources are typically packaged into sets or collections of related resources. A single API may provide access to a large number of collections. This Common API-2: GeoData Standard provides a means of organizing these collections and defines operations for the listing and description of collections.

Common API-2: GeoData does not specify the nature of the geospatial data that make up a collection. Rather, it provides a basic capability which should be applicable to any geospatial resource type or access mechanisms. Additional OGC Web API standards extend this foundation to define specific data access mechanisms.

2

CONFORMANCE

2 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 — Part 2: Geospatial Data defines API modules intended for re-use by other OGC Web API standards. For the purpose of conformance, the applicable API modules are identified by Conformance Classes.

Typically this standard will only be implemented through reference to these Conformance Classes by other standards.

This standard identifies four conformance classes. Each conformance class is defined by one requirements class.

The tests in Annex A are organized by Requirements Class. Therefore, an implementation of the *Collections* conformance class must pass all tests specified in Annex A for the *Collections* requirements class.

The requirements classes defined in Part 2: Geospatial Data are:

- Collections
- Uniform Multi-Dimension Collection
- HTML
- JSON

The Collections Requirements Class defines a common means to describe and access collections of spatial resources.

The Collections Requirements Class does not mandate a specific encoding or format for representing resources.

The *HTML* and *JSON* requirements classes specify representations for these resources in commonly used encodings for spatial data on the web.

Common API-2: GeoData builds on API modules defined in the `OGC API - Common - Part 1: Core` (API-Core) Standard.

Each requirements class in the Common API-2: GeoData Standard identifies any API-Core Conformance Classes upon which it depends.

Proof of conformance with a Conformance Class includes demonstration of conformance with all dependencies of that Conformance Class.

The abstract tests in Annex A have been organized to facilitate validation of these dependencies.

These tests have been organized by requirements class.

A referencing standard only has to require conformance with a Conformance Class and all of the requirements and relevant tests are identified.

In addition, each requirements class is organized as one or more trees.

Starting at a root test, a test script can traverse the tree to address all of the required tests, each in the appropriate context.

NOTE: The structure and organization of a collection of spatial resources is very much dependent on the nature of that resource and the expected access patterns.

This is information which cannot be specified in a common manner. This Standard specifies the requirements necessary to discover and understand a generic collection and its contents.

Requirements governing a specific type of resource are specified in resource-specific OGC API standards.

2.1. Summary of conformance URIs

Table 1 — Conformance class URIs

CORRESPONDING REQUIREMENTS CLASS	CONFORMANCE CLASS URI
Collections	https://www.opengis.net/spec/ogcapi-common-2/1.0/conf/collections
Uniform Multi-Dimension Collection	https://www.opengis.net/spec/ogcapi-common-2/1.0/conf/umd-collection
HTML	https://www.opengis.net/spec/ogcapi-common-2/1.0/conf/html
JSON	https://www.opengis.net/spec/ogcapi-common-2/1.0/conf/json

3

NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

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

R. Fielding, J. Reschke (eds.): IETF RFC 7231, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. RFC Publisher (2014). https://www.rfc-editor.org/info/rfc7231.

T. Bray (ed.): IETF RFC 8259, *The JavaScript Object Notation (JSON) Data Interchange Format*. RFC Publisher (2017). https://www.rfc-editor.org/info/rfc8259.

M. Nottingham: IETF RFC 8288, *Web Linking*. RFC Publisher (2017). https://www.rfc-editor.org/info/rfc8288.

G. Klyne, C. Newman: IETF RFC 3339, *Date and Time on the Internet: Timestamps*. RFC Publisher (2002). https://www.rfc-editor.org/info/rfc3339.

Charles Heazel: OGC 19-072, *OGC API — Common — Part 1: Core*. Open Geospatial Consortium (2023). http://www.opengis.net/doc/is/ogcapi-common-1/1.0.0.

John Herring: OGC 06-103r4, *OpenGIS Implementation Specification for Geographic information — Simple feature access — Part 1: Common architecture*. Open Geospatial Consortium (2011). http://www.opengis.net/doc/is/sfa/1.2.1.

Linda van den Brink, Clemens Portele, Panagiotis (Peter) A. Vretanos: OGC 10-100r3, *Geography Markup Language (GML) simple features profile (with Corrigendum)*. Open Geospatial Consortium (2011). https://portal.ogc.org/files/?artifact_id=42729.

J. Gregorio, R. Fielding, M. Hadley, M. Nottingham, D. Orchard: IETF RFC 6570, *URI Template*. RFC Publisher (2012). https://www.rfc-editor.org/info/rfc6570.

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

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

QUDT.org. **QUDT Ontologies, derived models and vocabularies 2.1**. Edited by R. Hodgson. 2024. Available at https://www.qudt.org/.

Regenstrief Institute, Inc. **The Unified Code for Units of Measure**. Edited by G. Schadow, .C J. McDonald. 2017. Available at https://ucum.org/ucum.

Open API Initiative: OpenAPI Specification, Version 3.0. The latest patch version at the time of publication of this standard was 3.0.3, available from https://spec.openapis.org/oas/v3.0.3.

JSON Schema: A Media Type for Describing JSON Documents. Edited by A. Wright, H. Andrews, B. Hutton, G. Dennis. 2022 [viewed 2023-10-14]. Available at https://json-schema.org/draft/2020-12/json-schema-core

JSON Schema Validation: A Vocabulary for Structural Validation of JSON. Edited by A. Wright, H. Andrews, B. Hutton. 2022 [viewed 2023-10-14]. Available at https://json-schema.org/draft/2020-12/json-schema-validation

4

TERMS, DEFINITIONS AND ABBREVIATED TERMS

4 TERMS, DEFINITIONS AND ABBREVIATED TERMS

This document uses the terms defined in OGC Policy Directive 49, which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word "shall" (not "must") is the verb form used to indicate a requirement to be strictly followed to conform to this document and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications (OGC 08-131r3), also known as the 'ModSpec'. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

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

4.1. Collection

(in the context of OGC APIs) A set of spatiotemporal data that may be available through one or more access mechanisms defined by OGC API standard(s)

4.2. Coverage

feature that acts as a function to return values from its range for any direct position within its domain

(Source: OGC 09-146r6)

4.3. Data access mechanism

(in the context of OGC APIs) A mechanism defined by an OGC API Standard to access spatiotemporal data from a collection
Such mechanism is based on Web resources which are sub-resources of that collection. The access mechanism could be as simple as the definition of a single sub-resource URL template. The access mechanism may allow Web clients to access parts and/or the totality of the data available from the collection.

4.4. Dataset

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

4.5. Distribution

A specific representation of a dataset. A dataset might be available in multiple serializations that may differ in various ways, including natural language, media-type or format, schematic organization, temporal and spatial resolution, level of detail or profiles (which might specify any or all of the above). (DCAT)

EXAMPLE: a downloadable file, an RSS feed or an API.

4.6. Extent

The area covered by something. Within this document, "extent" refers to spatial extent. The size or shape that may be expresses using coordinates. (W3C/OGC Spatial Data on the Web Best Practice)

4.7. Feature

abstraction of real world phenomena

(Source: ISO 19101-1:2014)

Note 1 to entry: A feature may occur as a type or an instance. Feature type or feature instance shall be used when only one is meant.

4.8. Feature Collection

set of features from a dataset

(Source: ISO 19168-1:2020)

4.9. Geometry

An ordered set of *n*-dimensional points in a given coordinate reference system. (W3C/OGC Spatial Data on the Web Best Practice)

4.10. OGC Web API

A Web API that implements one or more Conformance Classes from an OGC API Standard.

4.11. Resource

entity that might be identified (Dublin Core Metadata Initiative — DCMI Metadata Terms)

Note 1 to entry: The term “resource”, when used in the context of an OGC Web API standard, should be understood to mean a Web Resource unless otherwise indicated.

4.12. queryable

the name of a property of a resource that can be used in a filter expression

4.13. receivable

the name of a property of a web resource that can be included in representations of the resource when creating or updating a resource

4.14. Resource Type

the definition of a type of resource. Resource types are re-usable components which are independent of where the resource resides in the API.

Note 1 to entry: Resource types are re-usable components that are independent of where the resource resides in the API.”

4.15. returnable

the name of a property of a web resource that is available in representations of the resource when fetching the resource

4.16. sortable

the name of a property of a resource that can be used to sort resources of the same type

4.17. Spatial Resource

the resources which we usually think of as Geospatial Data. A Spatial Thing. (OGC 19-072)

4.18. Spatial Thing

anything with spatial extent, (i.e. size, shape, or position) and is a combination of the real-world phenomenon and its abstraction. (W3C/OGC Spatial Data on the Web Best Practice)

4.19. Temporal Coordinate System

temporal reference system based on an interval scale on which distance is measured as a multiple of a single unit of time

(Source: ISO 19108:2002)

4.20. Temporal Position

location relative to a temporal reference system

(Source: ISO 19108:2002)

4.21. Temporal Reference System

reference system against which time is measured

(Source: ISO 19108:2002)

4.22. Temporal Resource

the resources which we usually think of as time and date focused data. A Temporal Thing. (OGC 19-072)

4.23. Temporal Thing

Anything with temporal extent, i.e. duration. Examples are: the taking of a photograph, a scheduled meeting, a GPS time-stamped track-point. (W3C Basic Geo)

4.24. **Web API**

API using an architectural style that is founded on the technologies of the Web. (W3C Data on the Web Best Practices)

4.25. **Web Resource**

a resource that is identified by an URI.

4.26. Abbreviated terms

API	Application Programming Interface
CORS	Cross-Origin Resource Sharing
CRS	Coordinate Reference System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
OGC	Open Geospatial Consortium
TRS	Temporal Coordinate Reference System
URI	Uniform Resource Identifier
YAML	YAML Ain't Markup Language

5 CONVENTIONS

5 CONVENTIONS

This section provides details of conventions used in this document.

5.1. Identifiers

The normative provisions in this standard are denoted by the URI `https://www.opengis.net/spec/ogcapi-common-2/1.0`.

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

Additional information about the use of Identifiers in API-Common is provided in the OGC API — Common Users Guide.

5.2. Link relations

RFC 8288 (Web Linking) is used by this standard to express relationships between resources. Link relation types from the IANA Link Relations Registry are used wherever possible. Additional link relation types are registered with the OGC Naming Authority.

The link relationships used in Common API-2: GeoData are described in Table 2. Additional relation types may be used if the implementation warrants it.

Table 2 — Link Relations

LINK RELATION	PURPOSE
`alternate`	Refers to a substitute for this context [IANA]. Refers to a representation of the current resource which is encoded using another media type (the media type is specified in the `type` link attribute).
`https://www.opengis.net/def/rel/ogc/1.0/data-meta`	Identifies general metadata for the context (dataset or collection) that is primarily intended for consumption by machines.
`collection`	The target IRI points to a resource which represents the collection resource for the context IRI. [IANA]
`https://www.opengis.net/def/rel/ogc/1.0/conformance`	Refers to a resource that identifies the specifications that the link's context conforms to. [OGC]

LINK RELATION	PURPOSE
`https://www.opengis.net/def/rel/ogc/1.0/data`	Indicates that the link's context is a distribution of a dataset that is an API and refers to the root resource of the dataset in an API. [OGC]
`describedby`	Refers to a resource providing information about the link's context.[IANA] Links to external resources which further describe the subject resource
`item`	The target IRI points to a resource that is a member of the collection represented by the context IRI. [IANA]
`https://www.opengis.net/def/rel/ogc/1.0/items`	Refers to a resource that is comprised of members of the collection represented by the link's context. [OGC]
`license`	Refers to a license associated with this context. [IANA]
`self`	Conveys an identifier for the link's context. [IANA] A link to another representation of this resource.
`service-desc`	Identifies service description for the context that is primarily intended for consumption by machines. [IANA] API definitions are considered service descriptions.
`service-doc`	Identifies service documentation for the context that is primarily intended for human consumption. [IANA]
`service-meta`	Identifies general metadata for the context that is primarily intended for consumption by machines. [IANA]

Additional information on the use of link relationships is provided in the OGC API — Common Users Guide.

5.3. Geometry

5.3.1. Geospatial Geometry

Standardized concepts for geospatial characteristics are needed in order to share geographic information between applications. Concepts for geometry are key. These concepts are standardized in *ISO 19107*.

The geospatial geometry used in the OGC API — Common Standards is documented in the GML Simple Features Profile Standard. This Profile defines a subset of the ISO 19107 geometry which is aligned with the OGC Simple Features for SQL Standard. That geometry includes: Point, Curve (LineString), Surface (Polygon), MultiPoint, MultiCurve, and MultiSurface.

5.3.2. Temporal Geometry

Standardized concepts for temporal characteristics are also needed in order to share date and time information between applications. OGC API Common adopts the Gregorian calendar and a 24 hour time keeping system for its temporal geometry. All representations of that geometry which are discussed in this document conform to RFC 3339.

An ABNF representation of the RFC 3339 format is provided in Annex F.

5.4. Coordinate Reference Systems

As discussed in Chapter 9 of the W3C/OGC Spatial Data on the Web Best Practices document, the ability to express and share location in a consistent way is one of the most fundamental aspects of publishing geographic data. To do so, it is important to be clear about the coordinate reference system (CRS) within which the coordinates are expressed.

This Common API-2: GeoData Standard does not mandate the use of a specific coordinate reference system. However, if no CRS is specified, the default coordinate reference systems for spatial geometries are:

- [OGC:CRS84] — WGS 84 longitude and latitude without height
- [OGC:CRS84h] — WGS 84 longitude and latitude with ellipsoidal height

NOTE:[OGC:CRS84] is the CURIE form for the resolvable URI https://www.opengis.net/def/crs/OGC/0/CRS84, [OGC:CRS84h] is the CURIE form for the resolvable URI https://www.opengis.net/def/crs/OGC/0/CRS84h.

Temporal geometry is measured relative to an underlying temporal reference system (TRS). This Common API-2: GeoData Standard does not mandate a specific temporal coordinate reference system. However, all dates or timestamps discussed in this document are in the Gregorian calendar and conform to RFC 3339. In data, other temporal reference systems may be used where appropriate.

OGC Topic Volume 2 – Referencing by Coordinates (ISO 19111) provides the conceptual model for Coordinate Reference Systems.

5.5. API definition

5.5.1. General remarks

This OGC Standard specifies requirements and recommendations for the development of APIs that share spatial resources using a standard way of doing so. In general, deployed APIs will go beyond the requirements and recommendations stated in this Standard. They will support additional operations, parameters, and so on that are specific to the API or the software tool used to implement the API.

So that developers can more easily learn how to use the API, good documentation is essential. 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. OpenAPI is one way to provide that machine readable documentation.

5.5.2. Role of OpenAPI

This OGC API Standard uses OpenAPI 3.0 fragments in 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.

The OGC API — Common — Part 1 standard includes a <<http://www.opengis.net/spec/ogcapi-common-1/1.0/req/oas30,conformance class>> for API definitions that follow the OpenAPI specification 3.0. Alternative API definition languages are also allowed. Conformance classes for additional API definition languages will be added as the OGC API landscape continues to evolve.

5.5.3. 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 (only a subset of all possible values is allowed). 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, such as comments or description properties, may be changed or added.

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

5.5.4. Reusable OpenAPI components

Reusable components for OpenAPI definitions for an OGC API are referenced from this document. They are available from the OGC Schemas Registry at https://schemas.opengis.net/ogcapi/common/part1/1.0 and https://schemas.opengis.net/ogcapi/common/part2/1.0.

Additional information on the use of OpenAPI as an API definition is provided in the OGC API — Common Users Guide.

6

OVERVIEW

6 OVERVIEW

This OGC API — Common — Part 2: Geospatial Data Standard provides a way to discover and describe collections of spatiotemporal data. Web APIs specifying mechanisms to access this data are defined in separate OGC API Standards. This Standard depends on the Core requirements class of OGC API — Common — Part 1: Core which specifies fundamental requirements for OGC Web APIs based on HTTP. Implementations may also conform to the Landing Page requirements class of Part 1 which specifies additional requirements for the landing page (root of the API), conformance declaration (`/conformance`) and an API definition.

6.1. Collections

Spatial data is rarely considered as a single entity, but as aggregations of Spatio-Temporal things.

The purpose of the OGC API — Common — Part 2: Geospatial Data Standard is to list and describe the available geospatial data collections.

While *collection* is a common term, its specific meaning is often based on the context in which it is used. Given the focus on addressing geospatial data, a definition that reflects the unique characteristics of geospatial data collections is needed. Therefore, **for purposes of this standard**, a collection is defined as follows:

- Collection: (in the context of OGC APIs) A set of spatiotemporal data that may be available through one or more access mechanisms defined by OGC API standard(s).

OGC Web API standards could extend this definition to address the specific properties of the resources they describe.

6.2. Data access mechanisms

A collection of geospatial data may be accessed in more than one way. For example, a feature-centric approach may allow accessing individual features by their identifiers. Alternatively, some clients may wish to access a subset of geospatial information for a particular area, time and resolution of interest. Other clients may achieve better performance based on pre-determined partitioning of space using tiles or discrete global grid zones. These are all examples of data access mechanisms.

For example, a collection of data for land cover classification may be available either as individual features where one multi-polygon feature corresponds to low vegetation, another feature corresponds to trees and yet another feature corresponds to the built-up area. With OGC API — Features, each of these features may be individually accessible at:

`/collections/landcover/items/{itemId}`.

The same collection may also support an OGC API — Coverages access mechanism allowing to retrieve a subset of this same classified land cover for a given spatial area using:

`/collections/landcover/coverage?subset=Lat(10:20),Lon(30:40)`.

The OGC API — Common — Part 2: Geospatial Data Standard describes data collections, but does not define any *access mechanisms* to the data. Access mechanisms are defined in separate OGC Web API Standards extending this OGC API — Common — Part 2: Geospatial Data Standard.

The *access mechanisms* are defined by the resource paths which are specified in the different OGC API standards. The resources associated with a particular *access mechanism* for a specific collection will be under the collection resource:

`/collections/{collectionId}/{accessResources}`

where:

- `{collectionId}` is an identifier for the collection
- `{accessResources}` are resources defined for accessing data using a particular mechanism.

IMPORTANT

Editors of OGC API standards should carefully avoid redefining different resources with the same path in different OGC API standards.

The available data access mechanisms supported for a specific collection are typically advertised by including links using specific link relation types defined by the various OGC APIs in the `links` array of the collection description.

Access mechanisms should not be confused with *representations*.

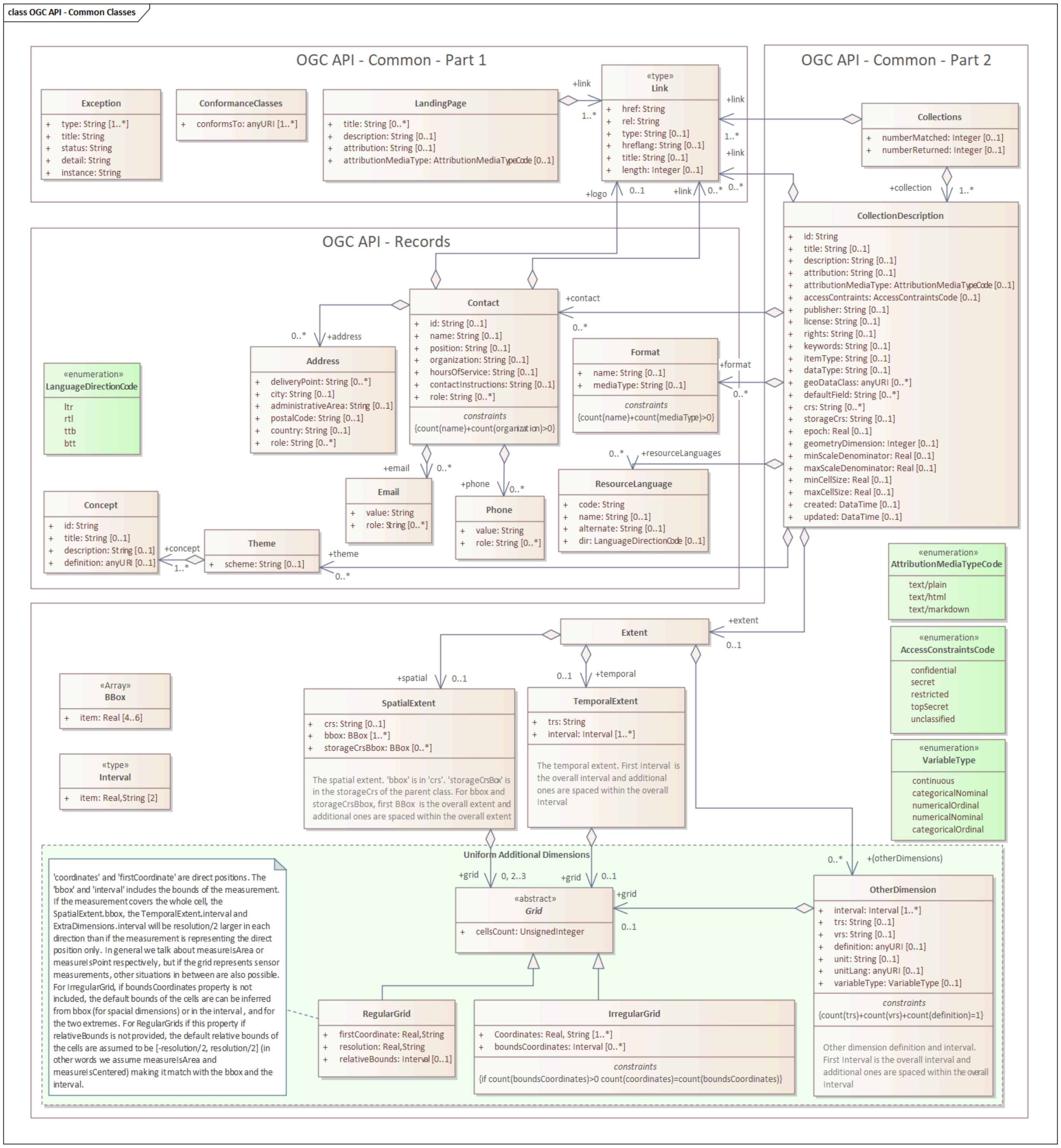
6.3. Representation

Data access mechanisms may support one or more representation (data formats / encodings) for each resource they define. A representation is typically negotiated between the server and client using the HTTP content negotiation mechanism (the `Accept:` request header). In the land cover example above, the client could request individual features as GeoJSON (`Accept:`

`application/geo+json`), and the coverage request could be requested as GeoTIFF (`Accept: image/tiff; application=geotiff`). Alternative formats supported by both the server and client could also be negotiated successfully.

6.4. UML Class Model

The following UML class diagram represents the conceptual model of the resources offered by the common part of an OGC Web API. In particular it describes the resources presented in OGC API Common — part 1: the Landing page (LandingPage class), the conformance page (ConformanceClasses) and the Exception report (Exception). It also describes the two resources specified in this Part 2: the collections page (Collections) and the collection description (CollectionDescription) and its associated classes. While they are precisely described in the requirements to follow, they are presented here as a UML diagram for clarity. The transformation of this UML diagram into a list of tables, one for each class, can be found Annex B.


class OGC API - Common Classes
OGC API - Common - Part 1
OGC API - Common - Part 2
OGC API - Records
Exception
+ type: String [1..*]
+ title: String
+ status: String
+ detail: String
+ instance: String
ConformanceClasses
+ conformsTo: anyURI [1..*]
LandingPage
+ title: String [0..*]
+ description: String [0..1]
+ attribution: String [0..1]
+ attributionMediaType: AttributionMediaTypeCode [0..1]
«type»
Link
+ href: String
+ rel: String
+ type: String [0..1]
+ hreflang: String [0..1]
+ title: String [0..1]
+ length: Integer [0..1]
+link
1..*
+link
1..*
+link
0..*
+logo
0..1
+link
0..*
Collections
+ numberMatched: Integer [0..1]
+ numberReturned: Integer [0..1]
+collection
1..*
CollectionDescription
+ id: String
+ title: String [0..1]
+ description: String [0..1]
+ attribution: String [0..1]
+ attributionMediaType: AttributionMediaTypeCode [0..1]
+ accessContraints: AccessContraintsCode [0..1]
+ publisher: String [0..1]
+ license: String [0..1]
+ rights: String [0..1]
+ keywords: String [0..*]
+ itemType: String [0..1]
+ dataType: String [0..1]
+ geoDataClass: anyURI [0..*]
+ defaultField: String [0..*]
+ crs: String [0..*]
+ storageCrs: String [0..1]
+ epoch: Real [0..1]
+ geometryDimension: Integer [0..1]
+ minScaleDenominator: Real [0..1]
+ maxScaleDenominator: Real [0..1]
+ minCellSize: Real [0..1]
+ maxCellSize: Real [0..1]
+ created: DataTime [0..1]
+ updated: DataTime [0..1]
Contact
+ id: String [0..1]
+ name: String [0..1]
+ position: String [0..1]
+ organization: String [0..1]
+ hoursOfService: String [0..1]
+ contactInstructions: String [0..1]
+ role: String [0..*]
constraints
{count(name)+count(organization)>0}
+contact
0..*
+address
0..*
Address
+ deliveryPoint: String [0..*]
+ city: String [0..1]
+ administrativeArea: String [0..1]
+ postalCode: String [0..1]
+ country: String [0..1]
+ role: String [0..*]
Format
+ name: String [0..1]
+ mediaType: String [0..1]
constraints
{count(name)+count(mediaType)>0}
+format
0..*
«enumeration»
LanguageDirectionCode
ltr
rtl
ttb
btt
+email
0..*
Email
+ value: String
+ role: String [0..*]
+phone
0..*
Phone
+ value: String
+ role: String [0..*]
+resourceLanguages
0..*
ResourceLanguage
+ code: String
+ name: String [0..1]
+ alternate: String [0..1]
+ dir: LanguageDirectionCode [0..1]
Concept
+ id: String
+ title: String [0..1]
+ description: String [0..1]
+ definition: anyURI [0..1]
+concept
1..*
Theme
+ scheme: String [0..1]
+theme
0..*
«enumeration»
AttributionMediaTypeCode
text/plain
text/html
text/markdown
«enumeration»
AccessConstraintsCode
confidential
secret
restricted
topSecret
unclassified
«enumeration»
VariableType
continuous
categoricalNominal
numericalOrdinal
numericalNominal
categoricalOrdinal
Extent
+extent
0..1
+spatial
0..1
+temporal
0..1
«Array»
BBox
+ item: Real [4..6]
«type»
Interval
+ item: Real,String [2]
SpatialExtent
+ crs: String [0..1]
+ bbox: BBox [1..*]
+ storageCrsBbox: BBox [0..*]
TemporalExtent
+ trs: String
+ interval: Interval [1..*]
Uniform Additional Dimensions
+grid
0, 2..3
+grid
0..1
+grid
0..1
«abstract»
Grid
+ cellsCount: UnsignedInteger
0..*
+(otherDimensions)
OtherDimension
+ interval: Interval [1..*]
+ trs: String [0..1]
+ vrs: String [0..1]
+ definition: anyURI [0..1]
+ unit: String [0..1]
+ unitLang: anyURI [0..1]
+ variableType: VariableType [0..1]
constraints
{count(trs)+count(vrs)+count(definition)=1}
RegularGrid
+ firstCoordinate: Real,String
+ resolution: Real,String
+ relativeBounds: Interval [0..1]
IrregularGrid
+ Coordinates: Real, String [1..*]
+ boundsCoordinates: Interval [0..*]
constraints
{if count(boundsCoordinates)>0 count(coordinates)=count(boundsCoordinates)}

The spatial extent. 'bbox' is in 'crs'. 'storageCrsBox' is in the storageCrs of the parent class. For bbox and storageCrsBbox, first BBox is the overall extent and additional ones are spaced within the overall extent

The temporal extent. First Interval is the overall interval and additional ones are spaced within the overall Interval

Other dimension definition and interval. First Interval is the overall interval and additional ones are spaced within the overall Interval

'coordinates' and 'firstCoordinate' are direct positions. The 'bbox' and 'interval' includes the bounds of the measurement. If the measurement covers the whole cell, the SpatialExtent.bbox, the TemporalExtent.interval and ExtraDimensions.interval will be resolution/2 larger in each direction than if the measurement is representing the direct position only. In general we talk about measureIsArea or measureIsPoint respectively, but if the grid represents sensor measurements, other situations in between are also possible. For IrregularGrid, if boundsCoordinates property is not included, the default bounds of the cells are can be inferred from bbox (for spacial dimensions) or in the interval, and for the two extremes. For RegularGrids if this property if relativeBounds is not provided, the default relative bounds of the cells are assumed to be [-resolution/2, resolution/2] (in other words we assume measureIsArea and measureIsCentered) making it match with the bbox and the interval.

Figure 1 — UML class diagram describing the common resources

7

REQUIREMENTS CLASS “COLLECTIONS”

7 REQUIREMENTS CLASS “COLLECTIONS”

REQUIREMENTS CLASS 1: COLLECTIONS	
IDENTIFIER	`https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/conf/collections`
PREREQUISITE	https://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
NORMATIVE STATEMENTS	Requirement 1: `/req/collections/collections-list-op` Requirement 2: `/req/collections/collections-list-success` Requirement 3: `/req/collections/collections-list-links` Requirement 4: `/req/collections/collections-list-collections` Requirement 5: `/req/collections/description-op` Requirement 6: `/req/collections/description-success` Requirement 7: `/req/collections/description-links` Requirement 8: `/req/collections/description-extent` Requirement 9: `/req/collections/description-extent-multi`

This Requirements Class describes the resources and operations used to list and describe geospatial collection resources exposed through a Web API consistently across OGC API Standards. This class does neither specify how to organize your data into collections nor how to provide access to the data described in the collections. That level of detail is reserved for OGC Web API standards defining access mechanisms through specific resources (see Access mechanisms Section).

RECOMMENDATION 1	
IDENTIFIER	`/rec/collections/rec-part1-landing-page`
A	An implementation of the `/Collections` Requirements Class SHOULD also implement the Landing Page conformance class which covers a landing page, a conformance page and an API definition as defined in OGC API — Common Part 1: Core.

The two resources defined by this Requirements Class are summarized in Table 3 and their operations are defined in the following requirements.

Table 3 — Collection Resources

RESOURCE	URI	HTTP METHOD	DESCRIPTION
List of Collections	/collections	GET	List of available Collections including some descriptions of each one
Collection description	/collections/{collectionId}	GET	Description about a specific collection of geospatial data with links to distribution

7.1. List of Collections Resource (/collections)

OGC API implementations typically organize their geospatial resources into collections. Information about those collections is accessed through the `/collections` path and the https://www.opengis.net/def/rel/ogc/1.0/data link relation.

7.1.1. Operation

REQUIREMENT 1	
IDENTIFIER	`/req/collections/collections-list-op`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`
A	The API SHALL support the HTTP GET operation at the path `/collections`.
B	If the API has a landing page (or an equivalent mechanism to expose root resources), this landing page or equivalent SHALL link to this resource listing collections (`/collections`) using the link relation type https://www.opengis.net/def/rel/ogc/1.0/data.

7.1.2. Response

REQUIREMENT 2	
IDENTIFIER	`/req/collections/collections-list-success`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`
A	A successful execution of the operation SHALL be responded with an HTTP status code `200`.

REQUIREMENT 2	
B	The content of the response SHALL validate against the JSON schema collections.yaml.

The collections response returned by this operation is based on the collections.yaml JSON schema. An example of a response listing collections are provided in Clause 7.1.2.1.

TODO: Update to schemas.opengis.net for publication

This Collections schema is further constrained by the following requirements and recommendations.

To support hypermedia navigation, the `links` property must be populated with sufficient hyperlinks to navigate through the whole dataset.

REQUIREMENT 3	
IDENTIFIER	`/req/collections/collections-list-links`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`
A	A `200`-response SHALL include the following links in the `links` property of the response: • A link to this response document (relation: `self`), • A link to the response document in every other media type supported by the API (relation: `alternate`).
B	All links SHALL include the `rel` property.
C	All links where `rel` is `self` or `alternate` SHALL include the `type` link parameter.

Additional information may be available to assist in understanding and using this dataset. Links to those resources should be provided as well.

RECOMMENDATION 2	
IDENTIFIER	`/rec/collections/collections-list-describedby`
A	If external schemas or descriptions exist that provide additional information about the structure or semantics for the resource, a `200`-response SHOULD include links to each of those resources in the `links` property of the response (relation: `describedby`).
B	The `type` link parameter SHOULD be provided for each link. This applies to resources that describe the whole dataset.

The `collections` property of the Collections response provides a description of each individual collection hosted by the API.

REQUIREMENT 4	
IDENTIFIER	`/req/collections/collections-list-collections`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`
A	For each spatial resource collection accessible through this API, metadata describing that collection SHALL be provided in the `collections` property of the list of Collections response.

The array items included in the `collection` property are described in the Collection Resource section of this Requirements Class.

This Requirements Class does not define any parameters for use against a `collections` resource.

Implementers who plan supporting the offering of large numbers of collections from the same API endpoint may consider extending their API with capabilities defined in OGC API — Records (with collections acting as the local resources of a local resource catalog), as well as candidate OGC API — Common extensions introducing capabilities for hierarchical collections, searching, filtering, and sorting collections.

7.1.2.1. Collections Response Example

This example response listing available collections in JSON is for a dataset with a single “buildings” feature collection.

There is a link to the resource listing collections itself (link relation type: `self`).

Representations of this resource in other formats are referenced (link relation type: `alternate`).

An additional link is to the logical schema of the collection (link relation type: `[ogc-rel:schema]`).

The reference systems for describing the spatiotemporal extent do not need to be specified, as they default to [OGC:CRS84] and Gregorian calendar / Coordinated Universal Time (UTC), and should always be specified this way for data referenced to Earth (independently of the native / storage reference system of the data itself).

```
{
  "links": [
    { "href": "https://data.example.org/collections.json",
      "rel": "self", "type": "application/json", "title": "This document" },
    { "href": "https://data.example.org/collections.html",
      "rel": "alternate", "type": "text/html", "title": "This document as HTML" }
  ],
  "collections": [
    {
      "id": "buildings",
      "title": "Buildings",
      "description": "Buildings in the city of Bonn.",
      "attribution": "Copyright © 2025 _City of Bonn_ ![logo](https://example.
org/cityOfBonn.png)",
      "attributionMediaType": "text/markdown",
      "extent": {
        "spatial": { "bbox": [ [ 7.01, 50.63, 7.22, 50.78 ] ] },
        "temporal": { "interval": [ [ "2010-02-15T12:34:56Z", "2018-03-
18T12:11:00Z" ] ] }
      },
      "links": [
        { "href": "https://data.example.org/collections/buildings",
          "rel": "self", "type": "application/json" },
        { "href": "https://data.example.org/collections/buildings/schema",
          "rel": "[ogc-rel:schema]", "type": "application/schema+json",
          "title": "Logical schema for buildings" }
      ]
    }
  ]
}
```

Listing 1 — Example `/collections` response listing collections

7.1.3. Error situations

See Annex E for general guidance.

7.2. Collection Description Resource (/collections/{collectionId})

Each resource collection is described by a set of metadata. That metadata can be accessed directly using the `/collections/{collectionId}` path and as an entry in the `collections` property of the `/collections` resource.

7.2.1. Operation

REQUIREMENT 5	
IDENTIFIER	`/req/collections/description-op`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`

REQUIREMENT 5	
A	The API SHALL support the HTTP GET operation at the path `/collections/{collectionId}`, where the parameter `collectionId` can be any values of the `id` property in the response to the list of collections (JSONPath: `$.collections[*].id` in `/collections`).

7.2.2. Response

REQUIREMENT 6	
IDENTIFIER	`/req/collections/description-success`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The content of a *Collection* description resource SHALL be based upon the JSON schema collection Desc.yaml.
C	The content of the *Collection* description resource response SHALL be consistent with the content for this collection in the `/collections` response, with the exception of `links` which can contain additional links in the response for the individual *Collection* description resource. That is, if a property is included in the response to the list of collections at `/collections`, the value for that property of the same collection (the collection whose `id` value corresponds to the `{collectionId}`) SHALL be identical for response for the *Collection* description resource (`/collections/{collectionId}`).

TODO: Update to schemas.opengis.net for publication

The collection description response returned by this operation is based on the collectionDesc.yaml JSON schema.

An example of a response describing a collection are provided in Clause 7.2.2.5.

Most of the properties of the Collection resource are self-explanatory. However, a few properties require additional explanation.

7.2.2.1. Attribution

A collection description can include an `attribution` property allowing clients to display a short attribution for data being visualized, typically shown at the bottom of a map. This attribution can contain markup text whose format may be indicated in the `attributionMediaType` property. That format can be either plain text (`text/plain`), HTML (`text/html`) or CommonMark (`text/markdown`). If the 'attributionMediaType' indicates something other than `text/plain`, the `attribution` element string should be interpreted by a markup parser selected based on that media type to be presented to the user (e.g., `text/markdown` will be parsed by a library supporting CommonMark). By allowing markup, the attribution string can import images (e.g., organization logos) and format the text (e.g., the name of the organization in italics). See the example collection response for an example of the use of markup in the attribution element.

7.2.2.2. Item Type

In some Geospatial collections, the members (`items`) that make up that collection can be individually accessed by a client. In this case, the `itemType` property in the Collection resource identifies the type of the `items` accessible from that collection.

RECOMMENDATION 3	
IDENTIFIER	`/rec/collections/description-item-type`
A	If the members (`items`) that make up a collection can be individually accessed by a client, then the `itemType` key SHOULD be included in the Collection resource to indicate the type of the items (e.g. `feature` or `record`).

7.2.2.3. Links

To support hypermedia navigation, the `links` property must be populated with sufficient hyperlinks to navigate through the whole dataset.

REQUIREMENT 7	
IDENTIFIER	`/req/collections/description-links`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`
A	A `200`-response SHALL include the following links in the `links` property of the response: • A link to this response document (relation: `self`), • A link to the response document in every other media type supported by the API (relation: `alternate`).
B	All links SHALL include the `rel` property.
C	All links where `rel` is `self` or `alternate` SHALL include the `type` link parameter.

Additional information may be available to assist in understanding and using this dataset. Links to those resources should be provided as well.

RECOMMENDATION 4	
IDENTIFIER	`/rec/collections/collection-describedby`
A	If external schemas or descriptions exist that provide additional information about the structure or semantics of the collection, a `200`-response SHOULD include links to each of those resources in the `links` property of the response (relation: `describedby`).
B	The `type` link parameter SHOULD be provided for each link.

7.2.2.4. Extent

The extent property defines a spatial-temporal envelope that encompasses the geospatial data in the collection. Since not all collections are nicely clustered around a single place in space and time, the extent property provides flexibility in how that surface can be defined.

- Spatial Bounding Box (Bbox) provides a set of rectangular bounding boxes which use geographic coordinates to envelope portions of the collection. Typically only the first element would be populated. Additional boxes may be useful, for example, when the collection is clustered in multiple, widely-separated locations.
- Temporal Interval provides a set of temporal periods. Typically only the first temporal period would be populated. However, like Bbox, additional periods can be added if the collection does not form a single temporal cluster.

The temporal reference system (`trs`) specified in the `temporal` property of the `extent` defines not only the reference system of the `interval`, but also the reference system for the primary temporal dimension of the data.

For the spatial extent, the `crs` specified in the `spatial` property of the `extent` defines only the coordinate reference system of the `bbox`. For data referenced to Earth, this `crs` will always be [OGC:CRS84] or [OGC:CRS84h]. The native CRS of the data is specified separately in the `storageCrs` property of the collection description. For some access mechanisms, this native CRS corresponds to the default output CRS. These access mechanisms may additionally support other output CRSs which are specified in the `crs` property of the collection description.

For the `storageCrs` as well as for the `crs` list of supported output CRSs, implementors of web APIs based on OGC API standards are strongly recommended to reference existing CRS definitions in the form of URIs (or the equivalent CURIEs) such as the ones provided by the EPSG database. However, there are situations where the required CRS has not been registered with an authority.

For these cases, where a URI does not exist, a Well-Known Text definition of the coordinate reference system, or a JSON encoding of the CRS (using the PROJ JSON schema or a potential “CRS JSON” successor once defined by the OGC) can be specified.

Such alternate CRS definitions can also be used for the `crs` of the `bbox` for data in an engineering CRS or otherwise not georeferenced to Earth.

WARNING

While some OGC API Standards explicitly support the use of CURIEs for properties of the collection description such as `storageCrs` and the `crs` array of supported output CRSs, other OGC API Standards do not. For servers implementing those OGC API Standards that do not allow the use of CURIEs there, full URIs therefore need to be used.

REQUIREMENT 8	
IDENTIFIER	`/req/collections/description-extent`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`
A	For each spatial collection resource, the `extent` property, if provided, SHALL define boundaries that encompass the spatial and temporal extent of the data in this collection. The temporal extent may use `null` values to indicate a half-bounded or unbounded time interval. If the data contains spatial or temporal information defined in multiple properties, it is up to implementation how this extent is derived from those properties.
B	For a collection whose native CRS (`storageCrs`) differs from the `crs` specified in the `spatial` property (which is always `[OGC:CRS84]` for data referenced to Earth), the bounds of the spatial dimensions in that native CRS SHALL be described in the `storageCrsBbox` of the `spatial` property (unlike the other dimensions which use the `interval` property).
C	The bounds of the spatial dimension in the CRS defined in the `crs` property of the `spatial` property (which defaults to `[OGC:CRS84]` if not present) SHALL be described in the `bbox` of the `spatial` property in that CRS (always `[OGC:CRS84]` for data referenced to Earth).

REQUIREMENT 9	
IDENTIFIER	`/req/collections/description-extent-multi`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`
A	If the `extent` property includes a member `spatial`, all data in the collection SHALL be inside the extent described by the first bounding box in the `bbox` array.
B	If the `extent` property includes a member `spatial` and the `bbox` array has more than one item, individual components (e.g., a feature or scene) of the collection SHALL be inside the extent described by one of the other bounding boxes in the `bbox` array.
C	If the `extent` property includes a member `temporal`, all data in the collection SHALL be inside the extent described by the first time interval in the `interval` array.

REQUIREMENT 9	
D	If the `extent` property includes a member `temporal` and the `interval` array has more than one item, individual components of the collection SHALL be inside the extent described by one of the other time intervals in the `interval` array.

NOTE: As a consequence, the first bounding box is a union of all subsequent bounding boxes, and the first temporal interval is a union of all subsequent temporal intervals.

RECOMMENDATION 5	
IDENTIFIER	`/rec/collections/description-extent-single`
A	While the spatial and temporal extents support multiple bounding boxes (`bbox` array) and time intervals (`interval` array) for advanced use cases, implementations SHOULD provide only a single bounding box or time interval unless the use of multiple values is important for the use of the dataset and agents using the API are known to be support multiple bounding boxes or time intervals.

PERMISSION 1	
IDENTIFIER	`/per/collections/description-extent-extensions`
A	This conformance class only specifies requirements for spatial and temporal extents. However, the `extent` object MAY be extended with additional members to represent other extents, such as thermal or pressure ranges. See the `Uniform additional dimension` conformance class for a well-defined way of describing additional dimensions.

RECOMMENDATION 6	
IDENTIFIER	`/rec/collections/description-extent-storage-crs-bbox`
A	If the native CRS of the data, as specified in the `storageCrs` property, is not [OGC:CRS84] or [OGC:CRS84h], the spatial extent of the data in that native CRS SHOULD be specified in the `storageCrsBbox` of the `spatial` property.

PERMISSION 2	
IDENTIFIER	`/per/collections/description-reference-systems`
A	For data not referenced to Earth, the spatial extent's `bbox` property MAY be specified in a CRS other than [OGC:CRS84] or [OGC:CRS84h], with this CRS indicated in the `crs` property of the `spatial` property.

PERMISSION 2	
B	For data for which the Gregorian calendar is not suitable, such as geological time scale, another temporal reference system MAY be specified in the the `trs` property of the temporal property.

7.2.2.5. Collection Object Examples

This Collection Description example response in JSON is for a single "buildings" collection.

The basic descriptive information includes:

- "id": an identifier for this collection
- "title": human-readable title for this collection
- "description": longer text describing this collection
- "attribution": markup providing attribution (owner, producer, logo, etc.) of this collection

The response includes links to:

- the response itself (link relation type: `self`),
- representations of this response in other formats are referenced using (link relation type: `alternate`),
- an additional link is to a logical schema for the collection (link relation type: [`ogc-rel:schema`]).

Finally, this response includes both spatial and temporal extents.

The reference systems for describing the spatiotemporal extent do not need to be specified, as they default to [OGC:CRS84] (longitude, latitude) and Gregorian calendar / Coordinated Universal Time (UTC), and should always be specified this way for data referenced to Earth (independently of the native / storage reference system of the data itself).

```
{
  "id": "buildings",
  "title": "Buildings",
  "description": "Buildings in the city of Bonn.",
  "attribution": "Copyright © 2025 _City of Bonn_ ![logo](https://example.org/
cityOfBonn.png)",
  "attributionMediaType": "text/markdown",
  "extent": {
    "spatial": { "bbox": [ [ 7.01, 50.63, 7.22, 50.78 ] ] },
    "temporal": { "interval": [ [ "2010-02-15T12:34:56Z", "2018-03-18T12:11:00Z"
] ] }
  },
  "links": [
    { "href": "https://data.example.org/collections/buildings",
      "rel": "self", "type": "application/json" },
    { "href": "https://data.example.org/collections/buildings/schema",
      "rel": "[ogc-rel:schema]", "type": "application/schema+json",
      "title": "Logical schema for buildings" }
  ]
}
```

Listing 2 — Example `/collections/{collectionId}` response describing a collections

7.2.3. Error Situations

See Annex E for general guidance.

If the parameter `collectionId` does not exist on the server, the status code of the response will be `404` (see Table E.1).

7.3. OGC API — Records compliance

When implementing this "Collections" requirement class, the `/collections` end-point can be considered a *Local Resource Catalog*, where the local resources are the collections being cataloged.

This requirement class is consistent with the requirements of OGC API — Records — Part 1: Core "Local Resources Catalog" (Deployment). An implementation may gain additional interoperability by conforming to that requirement class from OGC API — Records as well. In this case, the implementation should verify and declare conformance with these five conformance classes defined in Records corresponding to the following URIs:

Table 4

OGC API — RECORDS REQUIREMENTS CLASS	URI
Record Core	https://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core
Record Collection	https://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-collection
Crawlable Catalog	https://www.opengis.net/spec/ogcapi-records-1/1.0/conf/crawlable-catalog
Local Resources Catalog	https://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog
Autodiscovery	https://www.opengis.net/spec/ogcapi-records-1/1.0/conf/autodiscovery

The additional requirements include linking to the `/collections` end-point with an `https://www.opengis.net/def/rel/ogc/1.0/ogc-catalog` link relation from the landing page.

8

REQUIREMENTS CLASS “UNIFORM MULTI-DIMENSION COLLECTION”

8 REQUIREMENTS CLASS “UNIFORM MULTI-DIMENSION COLLECTION”

REQUIREMENTS CLASS 2: UNIFORM MULTI-DIMENSION COLLECTION	
IDENTIFIER	`https://www.opengis.net/spec/ogcapi-common-2/1.0/req/umd-collection`
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.2: `https://www.opengis.net/spec/ogcapi-common-2/1.0/conf/umd-collection`
PREREQUISITE	Requirements class 1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`
NORMATIVE STATEMENTS	Requirement 10: `/req/umd-collection/extent-uad-definition` Requirement 11: `/req/umd-collection/grid-description`

The Collections Requirements Class defines a Collection resource which supports both geospatial and temporal dimensions. However, the domain of some datasets cannot be fully described with only spatiotemporal dimensions. The Uniform Multi-Dimension Collection Requirements Class extends the Collection resource to support an unlimited number of dimensions defined in a uniform manner, so that clients supporting this requirements class can interpret these dimensions in a generic manner.

8.1. Uniform description of all dimensions

The Uniform Multi-Dimension Collection is an extension of the Collection resource.

This requirement class allows the definition of additional dimensions beyond spatial and temporal in a specific way which is consistent with the spatial and temporal dimensions.

Each dimension is identified by using a semantic `definition` property. An `interval` providing the lower and upper bound for the coordinates along that dimension also needs to be included, as well as a `unit` of measure where applicable.

As an alternative or in addition to the `definition`, a temporal reference system (`trs`) or a vertical reference system (`vrs`) can be specified. The `trs` or `vrs` property can be populated with either a URI or CURIE.

NOTE: If a 3D CRS exists including the vertical dimension, that CRS should be used inside the “spatial” object of the extent, instead of defining a separate vertical dimension using `vrs`. For vertical CRSs corresponding to the concept of a variable, such as a pressure level, the semantic definition for that variable should be used instead of `vrs`.

For additional dimensions beyond spatial and temporal, the reference system is defined as a combination of the `definition` and `unit`, or alternatively the `trs` or `vrs`. In addition to specifying the reference system of the `interval`, this also specifies the reference system of that dimension for the data itself. Unlike the spatial CRS, the collection description in this standard only supports defining a single reference system for the temporal dimension and each additional dimension.

To validate against this UMD Requirements Class, the `- type: object` line within the `anyOf:` in the extent.yaml YAML JSON Schema of the collection description's extent should be commented out.

REQUIREMENT 10	
IDENTIFIER	`/req/umd-collection/extent-uad-definition`
INCLUDED IN	Requirements class 2: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/umd-collection`
A	Any additional dimension property added to the `extent` of a collection description SHALL contain an `interval` property consisting of an array of one or more interval, where the first element describes the overall interval where data is available for the collection, whereas any additional elements describe intervals encompassing clusters of data availability within the overall interval.
B	Each interval element SHALL be described as an array of two values, with the first being the lower bound and the second the upper bound.
C	An unbounded or half-bounded interval SHALL be described using a null value for its lower and/or upper bound.
D	Any additional dimension SHALL specify a URI as either a `definition` to indicate the semantic concept for the variable (from any semantic definition vocabulary, such as QUDT) associated with the dimension, a `trs` indicating a temporal reference system or a `vrs` property indicating the vertical reference system.
E	For additional dimension defined using `definition`, where a particular unit is used, that dimension SHALL contain a `unit` property expressing the unit of measure, where the language for defining the unit is UCUM, unless specified otherwise in a `unitLang` property (using values such as `"UCUM"` or `"QUDT"`).

8.2. Describing gridded dimensions

For data which is gridded, whether using a regular or irregular grid, this conformance class defines a `grid` property which is used to describe such grids.

While the `interval` specified for the dimension (or `bbox` for spatial dimensions) describes the region occupied by all cells, including the region of validity or bounds of all measurements, this grid property describes in more details each individual measurement.

Common types of grids are often called *Measure-is-Area* or *Measure-is-Point*, but especially in the case of sensor measurements, other situations in-between are also possible.

8.2.1. Regular grids

A regular grid has data measurements separated by a constant distance (the resolution of the grid), while for an irregular grid the distance between data measurements varies. For both cases, the number of cells (`cellsCount`) is specified. A cell is a region within which a particular measurement is valid.

The bounds are specified as `relativeBounds` which are relative to points starting from a `firstCoordinate` separated by an equal distance of `resolution` units. The lower value for the `interval` for a particular dimension (or `bbox` in the case of spatial dimensions) corresponds to the `firstCoordinate` plus the first value of the `relativeBounds`, while the upper value corresponds to the `firstCoordinate` plus `cellsCount` times the `resolution` plus the second value of the `relativeBounds`.

For example, a regular grid where each measure is associated with infinitely small points would have `relativeBounds` of `[0, 0]`. With a `cellsCount` of 3, a `resolution` of 0.5, and a `firstCoordinate` at 0, the corresponding `interval` would be `[0, 1]`.

```
0.0     0.5     1.0
 │       │       │
 *       *       *
```

Listing 3 — Diagram illustrating a *Measure-is-Point* regular grid

When each measure is associated with an area filling entire cells, there could be multiple ways to define the same grid based on a preference of how the coordinate points are defined. Consider the following regular grids, with a `cellsCount` of 2, a `resolution` of `0.5`, and an interval of `[0, 1]`.

```
0.0      0.5     1.0
 │        │       │
 [========│=======]
```

Listing 4 — Diagram illustrating a *Measure-is-Area* regular grid

One way to describe this grid would be with the `firstCoordinate` at 0 and the relativeBounds as `[0, 0.5]`.

Another equivalent way to describe the same grid would be a `firstCoordinate` at 0.25 and the relativeBounds as `[-0.25, 0.25]`.

If the `relativeBounds` property is not provided, the default relative bounds of the cells are assumed to be `[-resolution/2, resolution/2]`. In other words, the cell, or the region of validity of the measurement, is centered on each repeating point starting from the `firstCoordinate`.

For a *Measure-is-Area* grid, the `interval` (or `bbox` for the spatial dimensions) will be `resolution/2` larger in each direction compared to the same number of cells and same resolution representing *Measure-is-Point* values.

8.2.2. Irregular grids

Irregular grids are described by explicitly listing a sequence of `coordinates` (one for each cell).

Bounds for each coordinates can also be individually specified in a parallel `boundsCoordinates` array.

If the `boundsCoordinates` property is not included, the default bounds of the cells are assumed to lie in the middle of the two values specified in `coordinates`, and for the two extremities, the lower (for the first cell) and upper bounds (for the last cell) are assumed to be at the same distance from the specified coordinate as to their immediate neighbor.

8.2.3. Requirements for describing grids

REQUIREMENT 11	
IDENTIFIER	`/req/umd-collection/grid-description`
INCLUDED IN	Requirements class 2: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/umd-collection`
A	If data is organized along a regular or irregular grid for the temporal and any additional dimension, that dimension SHALL contain a `grid` object property describing the grid of that dimension.
B	If data is organized along a regular or irregular grid for the spatial dimensions, the `spatial` object SHALL contain a `grid` array property, where each element is an object describing the grid of that dimension.
C	If data is organized along a regular or irregular grid for a dimension, the grid object(s) for that dimension SHALL indicate the number of coordinates along the dimension in a `cellsCount` property. • For values representing the whole area of contiguous cells spanning *resolution* units along the dimension, the *cellsCount* will be (*upperBound* — *lowerBound*) / *resolution*. • For values representing infinitely small point cells spaced by *resolution* units along the dimension, the *cellsCount* will be (*upperBound* — *lowerBound*) / *resolution* + 1.
D	For regularly gridded dimensions, the grid object SHALL contain a `resolution` property indicating the resolution of the grid in the unit of the `unit` property or the unit associated with the `trs`, `vrs`, or `storageCrs` in the case of the spatial dimensions (defaulting to [OGC:CRS84] if none is provided). The regular grid can be generated by incrementally adding the resolution starting from the lower bound of the overall interval.

REQUIREMENT 11	
E	For irregularly gridded (or categorical) dimensions, the grid object SHALL contain a `coordinates` property indicating the list of valid coordinates where data is available along that dimension.

PERMISSION 3	
IDENTIFIER	`/per/umd-collection/rec-categorical-dimension-interval`
A	For a categorical dimension, the first and the last categories listed in `coordinates` MAY be selected as the lower and upper bounds of the mandatory overall interval, or `null` MAY be used for both the lower and upper bounds.

9

REQUIREMENTS CLASSES FOR ENCODINGS

9 REQUIREMENTS CLASSES FOR ENCODINGS

9.1. Overview

This clause specifies two requirements classes for encodings to be used with the Collections and Collection resources. These encodings are commonly used encodings for spatial data on the web:

- HTML
- JSON

Neither of these encodings is mandatory. An implementation of the Collections requirements class may implement either, both, or neither of them.

9.2. Requirements 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: Web search engines,
- 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, this should be done in a way that enables users and search engines to access all of the data they are authorized to access.

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

REQUIREMENTS CLASS 3: HTML	
IDENTIFIER	`https://www.opengis.net/spec/ogcapi-common-2/1.0/req/html`
TARGET TYPE	Web API

REQUIREMENTS CLASS 3: HTML	
CONFORMANCE CLASS	Conformance class A.4: `https://www.opengis.net/spec/ogcapi-common-2/1.0/conf/html`
PREREQUISITES	Requirements class 1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections` Schema.org HTML Living Standard
NORMATIVE STATEMENTS	Requirement 12: `/req/html/definition` Requirement 13: `/req/html/content`

REQUIREMENT 12	
IDENTIFIER	`/req/html/definition`
INCLUDED IN	Requirements class 3: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/html`
A	`200`-responses of the server SHALL support the `text/html` media type for the list of Collections and Collection description resources.

REQUIREMENT 13	
IDENTIFIER	`/req/html/content`
INCLUDED IN	Requirements class 3: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/html`
A	Every `200`-response of the API with the media type "text/html" SHALL be a HTML 5 document that includes the following information in the HTML body: • All information identified in the schemas of the Response Object in the HTML `<body/>`, and • All links in HTML `<a/>` elements in the HTML `<body/>`.

RECOMMENDATION 7	
IDENTIFIER	`/rec/html/schema-org`
A	A `200`-response for the `/collections`, `/collections/{collectionId}` and `/collections/{collectionId}/schema` resources with the media type `text/html`, SHOULD include Schema.org annotations.

9.3. Requirements Class “JSON”

JSON is a text syntax that facilitates structured data interchange between programming languages. JSON is commonly used for Web-based software-to-software interchanges. Most Web developers are comfortable with using a JSON-based format, so supporting JSON is recommended for machine-to-machine interactions.

REQUIREMENTS CLASS 4: JSON	
IDENTIFIER	`https://www.opengis.net/spec/ogcapi-common-2/1.0/req/json`
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.3: `https://www.opengis.net/spec/ogcapi-common-2/1.0/conf/json`
PREREQUISITES	Requirements class 1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections` IETF RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format JSON Schema
NORMATIVE STATEMENTS	Requirement 14: `/req/json/definition` Requirement 15: `/req/json/content`

REQUIREMENT 14	
IDENTIFIER	`/req/json/definition`
INCLUDED IN	Requirements class 4: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/json`
A	`200`-responses of the server SHALL support the `application/json` media type for the list of Collections and Collection description resources.

REQUIREMENT 15	
IDENTIFIER	`/req/json/content`
INCLUDED IN	Requirements class 4: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/json`
A	Every `200`-response with the media type `application/json` SHALL include, or link to, a payload encoded according to the JSON Interchange Format.

REQUIREMENT 15	
B	The schema of all responses with the media type `application/json` SHALL conform with the JSON Schema specified for that resource.

JSON Schema for the Collections and Collection responses are available at collections.yaml and collectionDesc.yaml.

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

10

MEDIA TYPES

10 MEDIA TYPES

The media type for lists of collections and collection description as JSON is `application/json`.

The media type for HTML for all API resources would be `text/html`.

See also OGC API — Common — Part 1: Core Media Types section for additional media types such as Problem Details and OpenAPI API definitions.

Other media types used in OGC Standards can be found on OGC Media Type Register.

See also the IANA media type register.

A

ANNEX A (NORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE

A ANNEX A (NORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE

A.1. Introduction

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

The Conformance Classes addressed by this Abstract Test Suite are the:

- Collections Conformance Class
- Uniform Multi-Dimension Collection Conformance Class
- HTML Conformance Class
- JSON Conformance Class

A.2. Conformance Class Collections

CONFORMANCE CLASS A.1	
IDENTIFIER	`https://www.opengis.net/spec/ogcapi-common-2/1.0/conf/collections`
REQUIREMENTS CLASS	Requirements class 1: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections`
TARGET TYPE	Web API

CONFORMANCE CLASS A.1	
CONFORMANCE TESTS	Abstract test A.1: `/conf/collections/collections-list-op` Abstract test A.2: `/conf/collections/collections-list-success` Abstract test A.3: `/conf/collections/collections-list-links` Abstract test A.4: `/conf/collections/collections-list-collections` Abstract test A.5: `/conf/collections/description-op` Abstract test A.6: `/conf/collections/description-success` Abstract test A.7: `/conf/collections/description-links` Abstract test A.8: `/conf/collections/description-extent` Abstract test A.9: `/conf/collections/description-extent-multi`

The Collections Conformance Class has a dependency on OGC API — Common — Part 1: Core's "Core" conformance class, implying a dependency on HTTP and optionally HTTPS protocols:

https://www.opengis.net/spec/ogcapi-common-1/1.0/conf/core

Conformance with the Collections Conformance Class is demonstrated by execution, in order, of all abstract tests it consists of:

A.2.1. List of Collections (`/collections`) Tests

ABSTRACT TEST A.1	
IDENTIFIER	`/conf/collections/collections-list-op`
REQUIREMENT	Requirement 1: `/req/collections/collections-list-op`
TEST PURPOSE	Validate that information about the list of Collections can be retrieved from the expected location.
TEST METHOD	1. Issue an HTTP GET request without query parameters to the URL `{root}/collections` 2. Validate that a document was returned with a status code 200 3. Validate the contents of the returned document using test /conf/collections/collections-list-success.

ABSTRACT TEST A.2	
IDENTIFIER	`/conf/collections/collections-list-success`
REQUIREMENT	Requirement 2: `/req/collections/collections-list-success`
TEST PURPOSE	Validate that the list of Collections content complies with the required structure and contents.

ABSTRACT TEST A.2	
TEST METHOD	1. Validate the list of Collections resource for all supported media types using the resources and tests identified in Table A.1

The list of 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 against that schema. All supported formats should be exercised.

Table A.1 — Schema and Tests for Collections content

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

ABSTRACT TEST A.3	
IDENTIFIER	`/conf/collections/collections-list-links`
REQUIREMENT	Requirement 3: `/req/collections/collections-list-links`
TEST PURPOSE	Validate that the required links are included in the Collections document.
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.

ABSTRACT TEST A.4	
IDENTIFIER	`/conf/collections/collections-list-collections`
REQUIREMENT	Requirement 4: `/req/collections/collections-list-collections`
TEST PURPOSE	Validate that each collection accessible through the API is described in the Collections document.
TEST METHOD	1. Verify that the Collections document includes a `collections` property. 2. Verify that the `collections` property is an array.

ABSTRACT TEST A.4	
	3. Verify that there is an entry in the `collections` property for each resource collection accessible through the API. 4. Verify that each entry in the `collections` array is valid according to /conf/collections/description-success.

A.2.2. Collection Description (`/collections/{collectionId}`) Tests

ABSTRACT TEST A.5	
IDENTIFIER	`/conf/collections/description-op`
REQUIREMENT	Requirement 5: `/req/collections/description-op`
TEST PURPOSE	Validate that the Collection description content can be retrieved from the expected location.
TEST METHOD	For every Collection described in the list of Collections resource at `/collections`, issue an HTTP GET request to the URL `/collections/{collectionId}` where `{collectionId}` is the `id` property for the collection. 1. Validate that a Collection description was returned with a status code 200 2. Validate the contents of the returned document using test /conf/collections/description-success.

ABSTRACT TEST A.6	
IDENTIFIER	`/conf/collections/description-success`
REQUIREMENT	Requirement 6: `/req/collections/description-success`
TEST PURPOSE	Validate that a Collection document complies with the required structure, contents and values.
TEST METHOD	FOR each Collection description (`/collections/{collectionId}`) resource, validate: 1. That the Collection description resource includes an `id` property. 2. Validate the content of the Collection resource for all supported media types using the resources and tests identified in Table A.2 3. Verify that the content of the response is consistent with the content for the corresponding collection in the response to the `/collections` resource. That is, the values for `id`, `title`, `description` and `extent` are identical.

The Collection description 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 against that schema. All supported formats should be exercised.

Table A.2 — Schema and Tests for Collection content

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

ABSTRACT TEST A.7	
IDENTIFIER	`/conf/collections/description-links`
REQUIREMENT	Requirement 7: `/req/collections/description-links`
TEST PURPOSE	Validate that a Collection document includes all required links.
TEST METHOD	1. Verify that the Collection document includes a `links` property. 2. Verify that the `links` property includes an item which referes back to the Collection document (relation: `self`). 3. Verify that the `links` property includes an item for each supported encoding of this Collection document and that each of these items includes an `href` to an appropriate resource (relation: `alternate`). 4. Verify that all links include the `rel` and `type` link parameters.

ABSTRACT TEST A.8	
IDENTIFIER	`/conf/collections/description-extent`
REQUIREMENT	Requirement 8: `/req/collections/description-extent`
TEST PURPOSE	Validate the `extent` property if it is present
TEST METHOD	IF the `extent` property is present, THEN: 1. Verify that the `extent` provides bounding boxes that include all spatial geometries in the collection. 2. Verify that the `extent` provides time intervals that include all temporal geometries in the collection. NOTE:A temporal extent of `null` indicates a half-bounded or unbounded time interval.

ABSTRACT TEST A.9	
IDENTIFIER	`/conf/collections/description-extent-multi`
REQUIREMENT	Requirement 9: `/req/collections/description-extent-multi`
TEST PURPOSE	Validate the consistency of multi-elements extents property if applicable
TEST METHOD	IF the `extent` property is present, THEN: 1. Verify that for `spatial` properties having more than a single bounding box, all bounding boxes are included within the first bounding box. 2. Verify that for `temporal` properties (as well as for additional properties if conforming to Uniform Multi-dimension Collection) having more than a single interval, all intervals are within the first interval. NOTE: A temporal extent of `null` indicates a half-bounded or unbounded time interval.

A.3. Conformance Class Uniform Multi-Dimension Collection

CONFORMANCE CLASS A.2	
IDENTIFIER	`https://www.opengis.net/spec/ogcapi-common-2/1.0/conf/umd-collection`
REQUIREMENTS CLASS	Requirements class 2: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/umd-collection`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.10: `/conf/umd-collection/extent-uad-definition` Abstract test A.11: `/conf/umd-collection/grid-description`

A.3.1. Abstract Test for Uniform Multi-Dimension Collection Definition

ABSTRACT TEST A.10	
IDENTIFIER	`/conf/umd-collection/extent-uad-definition`
REQUIREMENT	Requirement 10: `/req/umd-collection/extent-uad-definition`

ABSTRACT TEST A.10	
TEST PURPOSE	To verify that the collection descriptions follows the uniform schema for describing multi-dimensional data collections
TEST METHOD	FOR each additional dimension beyond `spatial` and `temporal`: 1. Validate that the dimension property includes an `interval` property, 2. Validate that the dimension property includes includes a `trs`, `vrs` or `definition`, 3. Validate that the dimension property includes a `unit` if the `trs`, `vrs` or `definition` does not imply a specific unit.
DESCRIPTION	**NOTE 1:** The first two aspects of this validation can be performed by swapping the extent.yaml schema included by collectionDesc.yaml by a version where the `- type: object` within the `anyOf` has been removed, as per the comment saying *To validate against the Uniform Additional Dimensions requirements class, remove or comment out the following line*.

ABSTRACT TEST A.11	
IDENTIFIER	`/conf/umd-collection/grid-description`
REQUIREMENT	Requirement 11: `/req/umd-collection/grid-description`
TEST PURPOSE	Validate that the grid of gridded data is described
TEST METHOD	FOR each dimension of the extents, including `spatial` and `temporal` as well as additional dimensions: 1. If the data is known or described as being organized according to a specific regular or irregular grid, validate that the dimension property includes a `grid` property. 2. If the data is known or described as being organized according to a regular grid, validate that the `grid` property validates against regularGrid.yaml 3. If the data is known or described as being organized according to an irregular grid, validate that the `grid` property validates against irregularGrid.yaml
DESCRIPTION	**NOTE 2:** The `grid` property can be included whether an Implementation conforms to Uniform Additional Dimension or not, but grid description is mandatory for gridded data when conforming to this Conformance Class.

A.4. Conformance Class JSON

CONFORMANCE CLASS A.3	
IDENTIFIER	`https://www.opengis.net/spec/ogcapi-common-2/1.0/conf/json`
REQUIREMENTS CLASS	Requirements class 4: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/json`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.12: `/conf/json/definition` Abstract test A.13: `/conf/json/content`

A.4.1. JSON Definition

ABSTRACT TEST A.12	
IDENTIFIER	`/conf/json/definition`
REQUIREMENT	Requirement 14: `/req/json/definition`
TEST PURPOSE	Verify support for JSON
TEST METHOD	1. A resource is requested with response media type of `application/json` 2. All `200`-responses SHALL support the media type `application/json`.

A.4.2. JSON Content

ABSTRACT TEST A.13	
IDENTIFIER	`/conf/json/content`
REQUIREMENT	Requirement 15: `/req/json/content`
TEST PURPOSE	Verify the content of a JSON document given an input document and schema.
TEST METHOD	1. Validate that the document is a JSON document. 2. Validate the document against the schema using a JSON Schema validator.

A.5. Conformance Class HTML

CONFORMANCE CLASS A.4	
IDENTIFIER	`https://www.opengis.net/spec/ogcapi-common-2/1.0/conf/html`
REQUIREMENTS CLASS	Requirements class 3: `https://www.opengis.net/spec/ogcapi-common-2/1.0/req/html`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.14: `/conf/html/definition` Abstract test A.15: `/conf/html/content`

A.5.1. HTML Definition

ABSTRACT TEST A.14	
IDENTIFIER	`/conf/html/definition`
REQUIREMENT	Requirement 12: `/req/html/definition`
TEST PURPOSE	Verify support for HTML
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 A.15	
IDENTIFIER	`/conf/html/content`
REQUIREMENT	Requirement 13: `/req/html/content`
TEST PURPOSE	Verify the content of an HTML document given an input document and schema.
TEST METHOD	1. Validate that the document is an HTML 5 document 2. Manually inspect the document against the schema.

B

ANNEX B (INFORMATIVE) CLASS MODEL AS TABLES

ANNEX B (INFORMATIVE) CLASS MODEL AS TABLES

B.1. OGC API — Common package

B.1.1. OGC API — Common overview

B.1.2. Defining tables

Table B.1 — Elements of “OGC API — Common::” ()

NAME	
DEFINITION	
STEREOTYPE	interface
ABSTRACT	False
ASSOCIATIONS	(none)
ATTRIBUTES	(none)
CONSTRAINTS	(none)

Table B.2 — Elements of “OGC API — Common::OGC API — Common — Part 1” ()

NAME	OGC API — Common — Part 1
DEFINITION	

STEREOTYPE	interface
ABSTRACT	False
ASSOCIATIONS	(none)
ATTRIBUTES	(none)
CONSTRAINTS	(none)

Table B.3 — Elements of “OGC API — Common::OGC API — Common — Part 2” ()

NAME	OGC API — Common — Part 2
DEFINITION	
STEREOTYPE	interface
ABSTRACT	False
ASSOCIATIONS	(none)
ATTRIBUTES	(none)
CONSTRAINTS	(none)

Table B.4 — Elements of “OGC API — Common::OGC API — Records” ()

NAME	OGC API — Records
DEFINITION	
STEREOTYPE	interface
ABSTRACT	False
ASSOCIATIONS	(none)
ATTRIBUTES	(none)
CONSTRAINTS	(none)

Table B.5 — Elements of “OGC API — Common::Uniform Additional Dimensions” ()

NAME	Uniform Additional Dimensions
DEFINITION	
STEREOTYPE	interface
ABSTRACT	False
ASSOCIATIONS	(none)
ATTRIBUTES	(none)
CONSTRAINTS	(none)

Table B.6 — Elements of “OGC API — Common::Address” ()

NAME	Address		
DEFINITION	Postal address of the resource		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	deliveryPoint [*]		Address lines for the location (e.g. street name and door number)
	city [0..1]		City for the location
	administrativeArea [0..1]		State or province of the location
	postalCode [0..1]		ZIP or other postal code
	country [0..1]		Country of the physical address. ISO 3166-1 is recommended
	role [*]		The type of postal address (e.g. office, home, etc.)
CONSTRAINTS	(none)		

Table B.7 — Elements of “OGC API — Common::BBox” (Array)

NAME	BBox		
DEFINITION			
STEREOTYPE	Array		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	item [4..6]	Real	An array of 4 (2D) or 6 (3D) coordinates
CONSTRAINTS	(none)		

Table B.8 — Elements of “OGC API — Common::CollectionDescription” ()

NAME	CollectionDescription		
DEFINITION	Description of a collection		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	***NAME***	***TYPE***	***DEFINITION***
	Collections	Collections	(none)
	source:(self) [none]	Collections	(none)
	target:collection [1..*]	CollectionDescription	(none)
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	id [1]		Identifier of the collection, for example, used in URI path parameters
	title [0..1]		A short, human-readable summary of the collection

	description [0..1]		A human-readable explanation about data in the collection
	attribution [0..1]		For the collection that can contain markup text whose format may be indicated in the `attributionMediaType` property. That format can be either plain text (`text/plain`), HTML (`text/html`) or Common Mark (`text/markdown`)
	attributionMediaType [0..1]	AttributionMediaType Code	Media type for the markup language of the attribution: It can be either plain text (`text/plain`), HTML (`text/html`) or Common Mark (`text/markdown`).
	accessContraints [0..1]	AccessContraintsCode	Restrictions on the availability of the collection that the user needs to be aware of before using or redistributing the data.
	publisher [0..1]		Organization or individual responsible for making the data available
	license [0..1]		The legal provisions under which the data of this collection is made available
	rights [0..1]		A statement that concerns all rights not addressed by the license such as a copyright statement
	keywords [0..1]		The topic or topics of the resource. Typically represented using free-form keywords, tags, key phrases, or classification codes
	itemType [0..1]		Indicator about the type of the items in the collection if the collection has an accessible /collections/{collectionId}/items endpoint
	dataType [0..1]		Type of data use to represent the data in the collection. Can be map, vector or coverage
	geoDataClass [*]	anyURI	URIs identifying a class of data contained in the geospatial data (useful for example to determine compatibility with styles or processes)
	defaultField [*]		List of property names. In features is the typical list of properties that complement the geometry and in coverages are toe values of the cells.
	crs [*]		The list of coordinate reference systems supported by the API; the first item is the default coordinate reference system

	storageCrs [0..1]		The native coordinate reference system (i.e. , the most efficient CRS in which to request the data, possibly how the data is stored on the server); this is the default output coordinate reference system for Maps and Coverages
	epoch [0..1]		Epoch of the native (storage) Coordinate Reference System (CRS)
	geometryDimension [0..1]		Number of spatial dimensions of the primary geometry of individual elements of the data: 0 for points, 1 for curves, 2 for surfaces, 3 for solids and unspecified when mixed or unknown. Not to be confused with the dimensions of the domain which are defined by the extent element.
	minScaleDenominator [0..1]		Minimum scale denominator for usage of the collection
	maxScaleDenominator [0..1]		Maximum scale denominator for usage of the collection
	minCellSize [0..1]		Minimum cell size for usage of the collection
	maxCellSize [0..1]		Maximum cell size for usage of the collection
	created [0..1]	DataTime	Timestamp indicating when the data in the collection was first produced
	updated [0..1]	DataTime	Timestamp of the last change/revision to the data in the collection
CONSTRAINTS	(none)		

Table B.9 — Elements of “OGC API — Common::Collections” ()

NAME	Collections		
DEFINITION	A set of geospatial resources that may be available as one or more sub-resource distributions that conform to one or more OGC API standards.		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	***NAME***	***TYPE***	***DEFINITION***
	CollectionDescription	CollectionDescription	(none)

ATTRIBUTES	source:(self) [none]	Collections	(none)
	target:collection [1..*]	CollectionDescription	(none)
	NAME	*TYPE*	*DEFINITION*
	numberMatched [0..1]		Number of elements in the response that match the selection parameters like `bbox`.
	numberReturned [0..1]		Number of elements in the response. Omitted unknown or difficult to compute
CONSTRAINTS	(none)		

Table B.10 — Elements of “OGC API — Common::Concept” ()

NAME	Concept		
DEFINITION	A concept defined in a vocabulary		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	*NAME*	*TYPE*	*DEFINITION*
	id [1]	String	Identifier for the concept in the knowledge system
	title [0..1]	String	A human readable title for the concept
	description [0..1]	String	A human readable description for the concept
	definition [0..1]	anyURI	A URI providing further description of the concept
CONSTRAINTS	(none)		

Table B.11 — Elements of “OGC API — Common::ConformanceClasses” ()

NAME	ConformanceClasses

DEFINITION	A declaration of the conformance classes that the API conforms to.		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	conformsTo [1..*]	anyURI	A conformance class URIs from the ones defined in the OGC API standard documents
CONSTRAINTS	(none)		

Table B.12 — Elements of “OGC API — Common::Contact” ()

NAME	Contact		
DEFINITION	Contact point of the resource. It can be a person or an organization		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	id [0..1]		A value uniquely identifying a contact
	name [0..1]		The name of the responsible person
	position [0..1]		The name of the role or position of the responsible person taken from the organization’s formal organizational hierarchy or chart
	organization [0..1]		Organization/affiliation of the contact
	hoursOfService [0..1]		Time period when the contact can be contacted
	contactInstructions [0..1]		Supplemental instructions on how or when to contact can be made

	role [*]		The set of named duties, job functions and/or permissions associated with this contact. (e.g. developer, administrator, etc.)
CONSTRAINTS	(none)		

Table B.13 — Elements of “OGC API — Common::Coordinates” (Array)

NAME	Coordinates		
DEFINITION			
STEREOTYPE	Array		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	*NAME*	*TYPE*	*DEFINITION*
	item [1..*]	Real,String	(none)
CONSTRAINTS	(none)		

Table B.14 — Elements of “OGC API — Common::Email” ()

NAME	Email		
DEFINITION	Email address of the resource		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	*NAME*	*TYPE*	*DEFINITION*
	value [1]		The value is the email itself
	role [*]		The type of email (e.g. home, work, etc.)
CONSTRAINTS	(none)		

Table B.15 — Elements of “OGC API — Common::Exception” ()

NAME	Exception		
DEFINITION	A schema for exceptions based on RFC 7807		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	*NAME*	*TYPE*	*DEFINITION*
	type [1..*]		A URI that identifies the problem type. When this member is not present, its value is assumed to be about:blank
	title [1]		A short, human-readable summary of the problem type. It should not change from occurrence to occurrence of the problem
	status [1]		The HTTP status code generated by the server for this occurrence of the problem
	detail [1]		A human-readable explanation specific to this occurrence of the problem
	instance [1]		A URI reference that identifies the specific occurrence of the problem. It may or may not yield further information if dereferenced
CONSTRAINTS	(none)		

Table B.16 — Elements of “OGC API — Common::Extent” ()

NAME	Extent		
DEFINITION	The extent of the resource with spatial, temporal optional Uniform Additional Dimensions (UAD) schema. The first item in the arrays for bbox or interval describe the overall extent of the data. All subsequent items describe more precise extents, e.g., to identify clusters of data.		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	*NAME*	*TYPE*	*DEFINITION*
	OtherDimension	OtherDimension	(none)
	source:(self) [none]	Extent	(none)

	target:(otherDimensions) [0..*]	OtherDimension	(none)
	SpatialExtent	SpatialExtent	(none)
	source:(self) [none]	Extent	(none)
	target:spatial [0..1]	SpatialExtent	(none)
	TemporalExtent	TemporalExtent	(none)
	source:(self) [none]	Extent	(none)
	target:temporal [0..1]	TemporalExtent	(none)
ATTRIBUTES	(none)		
CONSTRAINTS	(none)		

Table B.17 — Elements of “OGC API — Common::Fields” (Array)

NAME	Fields		
DEFINITION			
STEREOTYPE	Array		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	item [*]		(none)
CONSTRAINTS	(none)		

Table B.18 — Elements of “OGC API — Common::Format” ()

NAME	Format
DEFINITION	Format of the resource

STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	name [0..1]	String	Name of the format
	mediaType [0..1]	String	Media type of the format
CONSTRAINTS	(none)		

Table B.19 — Elements of "OGC API — Common::Grid" (abstract)

NAME	Grid		
DEFINITION	Provides information about the limited availability of data within the collection organized as a grid (regular or irregular) along the dimension.		
STEREOTYPE	abstract		
GENERALIZATION OF	IrregularGrid, RegularGrid		
ABSTRACT	True		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	cellsCount [1]	UnsignedInteger	Number of samples available along the dimension for data organized as a regular or irregular grid
CONSTRAINTS	(none)		

Table B.20 — Elements of "OGC API — Common::Interval" (type)

NAME	Interval
DEFINITION	Commonly a pair of numbers but it is string to allow for a pair of dates
STEREOTYPE	type

ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	item [2..2]	Real,String	I
CONSTRAINTS	(none)		

Table B.21 — Elements of “OGC API — Common::IrregularGrid” ()

NAME	IrregularGrid		
DEFINITION	Irregular grid description using syncronized lists of coordinates and its bounds.		
STEREOTYPE	interface		
INHERITS FROM	Grid		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	Coordinates [1..*]	Real, String	List of coordinates along the dimension for which data organized as an irregular grid in the collection is available (e.g., 2, 10, 80, 100).
	boundsCoordinates [*]	Interval	Coordinates of the lower and upper bounds of each cell in absolute units for irregular grids describing the geometry each cell
CONSTRAINTS	(none)		

Table B.22 — Elements of “OGC API — Common::LandingPage” ()

NAME	LandingPage
DEFINITION	A page that serves as the root node of the API Resource tree and provides the information needed to navigate all the resources exposed through the API.
STEREOTYPE	interface

ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	title [*]		The title of the API. While a title is not required, implementors are strongly advised to include one
	description [0..1]		A longer description of the API.
	attribution [0..1]		The `attribution` of the API should be short and intended for presentation to a user, for example, in a corner of a map. Parts of the text can be links to other resources if additional information is needed. The string can include HTML markup or markdown.
	attributionMediaType [0..1]	AttributionMediaTypeCode	The 'attribution' can contain markup text whose format may be indicated in the `attributionMediaType` property (e.g., `text/html` for HTML or `text/markdown` for CommonMark). If the 'attributionMedia Type' indicates something other than `text/plain`, the `attribution` element string should be interpreted by a markup parser selected based on that media type to be presented to the user (e.g., `text/markdown` will be parsed by a library supporting CommonMark). By allowing markup, the attribution string can import images (e.g., organization logos) and format the text (e.g., the name of the organization in italics).
CONSTRAINTS	(none)		

Table B.23 — Elements of "OGC API — Common::Link" (type)

NAME	Link
DEFINITION	An expression of a relationship between resources.
STEREOTYPE	type
ABSTRACT	False

ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	href [1]		URI to a remote resource (or resource fragment)
	rel [1]		The type or semantics of the relation
	type [0..1]		A hint indicating what the media type of the result of dereferencing the link should be.
	hreflang [0..1]		A hint indicating what the language of the result of dereferencing the link should be.
	title [0..1]		Used to label the destination of a link such that it can be used as a human-readable description
	length [0..1]		A hint indicating the Content-length of the result of dereferencing the link should be
CONSTRAINTS	(none)		

Table B.24 — Elements of "OGC API — Common::OtherDimension" ()

NAME	OtherDimension		
DEFINITION	Other dimension definition and interval. First Interval is the overall interval and additional ones are spaced within the overall Interval		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	***NAME***	***TYPE***	***DEFINITION***
	Extent	Extent	(none)
	source:(self) [none]	Extent	(none)
	target:(other Dimensions) [0..*]	OtherDimension	(none)
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	interval [1..*]	Interval	One or more intervals that describe the extent for this dimension of the dataset. The value `null` is supported and indicates an unbounded or half-bounded interval. The first interval describes the overall extent of the data for this dimension. All subsequent intervals describe

			more precise intervals, e.g., to identify clusters of data. Clients only interested in the overall extent will only need to access the first item (a pair of lower and upper bound values).
	trs [0..1]		Temporal Coordinate Reference System (e.g. as defined by Features for 'temporal')
	vrs [0..1]		Vertical Coordinate Reference System (e.g. as defined in EDR for 'vertical')
	definition [0..1]	anyURI	A URI to the definition of the measured or observed property corresponding to this dimension
	unit [0..1]		The unit of measure in which the interval and/or grid values are expressed
	unitLang [0..1]	anyURI	The language (or vocabulary) in which the unit is expressed (defaults to "UCUM" if not specified)
	variableType [0..1]	VariableType	The type of variable which may inform correct interpretation and interpolation methods
CONSTRAINTS	(none)		

Table B.25 — Elements of "OGC API — Common::Phone" ()

NAME	Phone		
DEFINITION	Phone or Fax of the resource		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	value [1]		The value is the phone number itself
	role [*]		The type of phone number (e.g. home, work, fax, etc.)
CONSTRAINTS	(none)		

Table B.26 — Elements of "OGC API — Common::RegularGrid" ()

NAME	RegularGrid

DEFINITION	Resolution of regularly gridded data along the dimension in the collection.		
STEREOTYPE	interface		
INHERITS FROM	Grid		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	firstCoordinate [1]	Real,String	First coordinate where a regular grid begins, with subsequent coordinates adding `resolution` unit at each step. Commonly is a number but can be a string to support date first coordinates
	resolution [1]	Real,String	Resolution of regularly gridded data along the dimension in the collection. Commonly is a number but can be a string to support date resolution
	relativeBounds [0..1]	Interval	Distance in units from coordinate to the lower and upper bounds of each cell for regular grids, describing the geometry of the cells
CONSTRAINTS	(none)		

Table B.27 — Elements of "OGC API — Common::ResourceLanguage" ()

NAME	ResourceLanguage		
DEFINITION	Language of the resource.		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	code [1]	String	The language tag as per RFC-5646
	name [0..1]	String	The untranslated name of the language
	alternate [0..1]	String	The name of the language in another well-understood language, usually English

	dir [0..1]	LanguageDirection Code	The direction for text in this language. The default, `ltr` (left-to-right), represents the most common situation. However, care should be taken to set the value of `dir` appropriately if the language direction is not `ltr`. Other values supported are `rtl` (right-to-left), `ttb` (top-to-bottom), and `btt` (bottom-to-top)
CONSTRAINTS	(none)		

Table B.28 — Elements of “OGC API — Common::TemporalExtent” ()

NAME	TemporalExtent		
DEFINITION	The temporal extent. First Interval is the overall interval and additional ones are spaced within the overall Interval		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	***NAME***	***TYPE***	***DEFINITION***
	Extent	Extent	(none)
	source: (self) [none]	Extent	(none)
	target: temporal [0..1]	TemporalExtent	(none)
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	trs [1]		Coordinate reference system of the coordinates in the temporal extent (property `interval`). The default reference system is the Gregorian calendar. In the Core this is the only supported temporal coordinate reference system. Extensions may support additional temporal coordinate reference systems and add additional enum values.
	interval [1..*]	Interval	One or more time intervals that describe the temporal extent of the dataset. In the Core only a single time interval is supported. Extensions may support multiple intervals. The first time interval describes the overall temporal extent of the data. All subsequent time intervals describe more precise time intervals, e.g., to identify clusters of data. Clients only interested in the overall extent will only need to access the first item in each array.
CONSTRAINTS	(none)		

Table B.29 — Elements of “OGC API — Common::SpatialExtent” ()

NAME	SpatialExtent		
DEFINITION	The spatial extent. ‘bbox’ is in ‘crs’. ‘storageCrsBox’ is in the storageCrs of the parent class. For bbox and storageCrsBbox, first BBox is the overall extent and additional ones are spaced within the overall extent		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	***NAME***	***TYPE***	***DEFINITION***
	Extent	Extent	(none)
	source:(self) [none]	Extent	(none)
	target:spatial [0..1]	SpatialExtent	(none)
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	crs [0..1]		Coordinate reference system of the coordinates of the `bbox` property. The default reference system is CRS84. CRS84h for coordinates with height. For non-terrestrial coordinate reference system, another CRS may be specified
	bbox [1..*]	BBox	One or more bounding boxes that describe the spatial extent of the dataset. The first bounding box describes the overall spatial extent of the data. All subsequent bounding boxes describe more precise bounding boxes, e.g., to identify clusters of data. Clients only interested in the overall spatial extent will only need to access the first item in each array.
	storageCrsBbox [*]	BBox	One or more bounding boxes that describe the spatial extent of the dataset in the storage (native) CRS (`storageCrs` property in the collectionDesc). The first bounding box describes the overall spatial extent of the data. All subsequent bounding boxes describe more precise bounding boxes, e.g., to identify clusters of data. Clients only interested in the overall spatial extent will only need to access the first item in each array.
CONSTRAINTS	(none)		

Table B.30 — Elements of “OGC API — Common::Theme” ()

NAME	Theme		
DEFINITION	A theme defined by a concept in an scheme of concepts.		
STEREOTYPE	interface		
ABSTRACT	False		
ASSOCIATIONS	(none)		
ATTRIBUTES	***NAME***	***TYPE***	***DEFINITION***
	scheme [0..1]	String	An identifier for the knowledge organization system used to classify the resource. It is recommended that the identifier be a resolvable URI. The list of schemes used in a searchable catalog can be determined by inspecting the server’s OpenAPI document or, if the server implements CQL2, by exposing a queryable (e.g. named `scheme`) and enumerating the list of schemes in the queryable’s schema definition.
CONSTRAINTS	(none)		

Table B.31 — Definition table of “OGC API — Common::AccessConstraintsCode” (enumeration)

NAME:	AccessConstraintsCode	
DEFINITION:	Enumeration of contraints	
STEREOTYPE:	enumeration	
ABSTRACT:	False	
ASSOCIATIONS:	(none)	
VALUES:	*Name*	*Definition*
	confidential	
	secret	
	restricted	
	topSecret	

	unclassified

Table B.32 — Definition table of “OGC API — Common::AttributionMediaTypeCode” ()

NAME:	AttributionMediaTypeCode	
DEFINITION:	Enumeration of media types for attribution	
STEREOTYPE:	interface	
ABSTRACT:	False	
ASSOCIATIONS:	(none)	
VALUES:	*Name*	*Definition*
	text/plain	
	text/html	
	text/markdown	

Table B.33 — Definition table of “OGC API — Common::LanguageDirectionCode” (enumeration)

NAME:	LanguageDirectionCode	
DEFINITION:	Enumeration of language directions.	
STEREOTYPE:	enumeration	
ABSTRACT:	False	
ASSOCIATIONS:	(none)	
VALUES:	*Name*	*Definition*
	ltr	left to right
	rtl	right to left
	ttb	Top to bottom

btt	Bottom to top

Table B.34 — Definition table of “OGC API — Common::VariableType” ()

NAME:	VariableType	
DEFINITION:	Enumeration of Variable types. We reduced the 8 possible categories (combinations of: quantitative/qualitative, discrete/continuous, ordinal/nominal) to 5 because continuous shall be ordinal and numerical.	
STEREOTYPE:	interface	
ABSTRACT:	False	
ASSOCIATIONS:	(none)	
VALUES:	*Name*	*Definition*
	continuous	Continuous
	categoricalNominal	Categorical and nominal
	numericalOrdinal	Numerical and ordinal
	numericalNominal	Numerical and nominal
	categoricalOrdinal	Categorical and ordinal

C

ANNEX C (INFORMATIVE) GLOSSARY

C

ANNEX C (INFORMATIVE) GLOSSARY

Conformance Test Module
set of related tests, all within a single conformance test class (OGC 08-131r3)

NOTE 1: 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.
This term and those associated to it are included here for consistency with ISO 19105.

Conformance Test Class; Conformance Test Level
set of conformance test modules that must be applied to receive a single **certificate of conformance**. (OGC 08-131r3)

NOTE 2: When no ambiguity is possible, the word `test` may be left out, so conformance test class may be called a conformance class.

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)

Recommendation
expression in the content of a document conveying that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required, or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited (OGC 08-131r3)

NOTE 3: "Although using normative language, a recommendation is not a requirement. The usual form replaces the `shall` (imperative or command) of a requirement with a `should` (suggestive or conditional)." (ISO Directives Part 2)

Requirement
expression in the content of a document conveying criteria to be fulfilled if compliance with the document is to be claimed and from which no deviation is permitted (OGC 08-131r3)

Requirements Class
aggregate of all requirement modules that must all be satisfied to satisfy a conformance test class (OGC 08-131r3)

Requirements Module

aggregate of requirements and recommendations of a specification against a single standardization target type (OGC 08-131r3)

Standardization Target
entity to which some requirements of a standard apply (OGC 08-131r3)

NOTE 4: The standardization target is the entity which may receive a certificate of conformance for a requirements class.

D

ANNEX D (NORMATIVE) BACKUS-NAUR FORMS

D

ANNEX D (NORMATIVE) BACKUS-NAUR FORMS

D.1. BNF for URI

The following Augmented Backus-Naur Form (ABNF) is from Appendix A of IETF RFC 3986.

```
URI           = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

hier-part     = "//" authority path-abempty
             / path-absolute
             / path-rootless
             / path-empty

URI-reference = URI / relative-ref

absolute-URI  = scheme ":" hier-part [ "?" query ]

relative-ref  = relative-part [ "?" query ] [ "#" fragment ]

relative-part = "//" authority path-abempty
                 / path-absolute
                 / path-noscheme
                 / path-empty

scheme        = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )

authority     = [ userinfo "@" ] host [ ":" port ]
userinfo      = *( unreserved / pct-encoded / sub-delims / ":" )
host          = IP-literal / IPv4address / reg-name
port          = *DIGIT

IP-literal    = "[" ( IPv6address / IPvFuture  ) "]"

IPvFuture     = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )

IPv6address   =                            6( h16 ":" ) ls32
              /                       "::" 5( h16 ":" ) ls32
              / [               h16 ] "::" 4( h16 ":" ) ls32
              / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
              / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
              / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
              / [ *4( h16 ":" ) h16 ] "::"              ls32
              / [ *5( h16 ":" ) h16 ] "::"              h16
              / [ *6( h16 ":" ) h16 ] "::"

h16           = 1*4HEXDIG
```

```
ls32          = ( h16 ":" h16 ) / IPv4address
IPv4address   = dec-octet "." dec-octet "." dec-octet "."

dec-octet     = DIGIT                 ; 0-9
              / %x31-39 DIGIT         ; 10-99
              / "1" 2DIGIT            ; 100-199
              / "2" %x30-34 DIGIT     ; 200-249
              / "25" %x30-35          ; 250-255

reg-name      = *( unreserved / pct-encoded / sub-delims )

path          = path-abempty    ; begins with "/" or is empty
              / path-absolute   ; begins with "/" but not "//"
              / path-noscheme   ; begins with a non-colon segment
              / path-rootless   ; begins with a segment
              / path-empty      ; zero characters

path-abempty  = *( "/" segment )
path-absolute = "/" [ segment-nz *( "/" segment ) ]
path-noscheme = segment-nz-nc *( "/" segment )
path-rootless = segment-nz *( "/" segment )
path-empty    = 0<pchar>

segment       = *pchar
segment-nz    = 1*pchar
segment-nz-nc = 1*( unreserved / pct-encoded / sub-delims / "@" )
              ; non-zero-length segment without any colon ":"

pchar         = unreserved / pct-encoded / sub-delims / ":" / "@"

query         = *( pchar / "/" / "?" )

fragment      = *( pchar / "/" / "?" )

pct-encoded   = "%" HEXDIG HEXDIG

unreserved    = ALPHA / DIGIT / "-" / "." / "_" / "~"
reserved      = gen-delims / sub-delims
gen-delims    = ":" / "/" / "?" / "#" / "[" / "]" / "@"
sub-delims    = "!" / "$" / "&" / "'" / "(" / ")"
              / "*" / "+" / "," / ";" / "="
```

Listing D.1

D.2. BNF for Date-Time

The following Augmented Backus-Naur Form (ABNF) is from IETF RFC 3339.

```
date-fullyear   = 4DIGIT
date-month      = 2DIGIT  ; 01-12
date-mday       = 2DIGIT  ; 01-28, 01-29, 01-30, 01-31 based on month/year
time-hour       = 2DIGIT  ; 00-23
time-minute     = 2DIGIT  ; 00-59
time-second     = 2DIGIT  ; 00-58, 00-59, 00-60 based on leap second rules
time-secfrac    = "." 1*DIGIT
time-numoffset  = ("+" / "-") time-hour ":" time-minute
time-offset     = "Z" / time-numoffset
partial-time    = time-hour ":" time-minute ":" time-second [time-secfrac]
```

```
full-date      = date-fullyear "-" date-month "-" date-mday
full-time      = partial-time time-offset
date-time      = full-date "T" full-time
```

Listing D.2

Note that unlike ISO 8601, the local time zone offset is required by RFC 3339.

E

ANNEX E (NORMATIVE) HTTP STATUS CODES

E ANNEX E (NORMATIVE) HTTP STATUS CODES

Table E.1 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 E.1 — Typical HTTP status codes

STATUS CODE	DESCRIPTION
`200`	A successful request.
`302`	The target resource was found but resides temporarily under a different URI. A 302 response is not evidence that the operation has been successfully completed.
`303`	The server is redirecting the user agent to a different resource. A 303 response is not evidence that the operation has been successfully completed.
`304`	An entity tag was provided in the request and the resource has not changed since the previous request.
`307`	The target resource resides temporarily under a different URI and the user agent MUST NOT change the request method if it performs an automatic redirection to that URI.
`308`	Indicates that the target resource has been assigned a new permanent URI and any future references to this resource ought to use one of the enclosed URIs.
`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 authorized to perform the requested operation on the resource.
`404`	The requested resource does not exist on the server. For example, a path parameter had an incorrect value.
`405`	The request method is not supported. For example, a POST request was submitted, but the resource only supports GET requests.
`406`	Content negotiation failed. For example, the `Accept` header submitted in the request did not support any of the media types supported by the server for the requested resource.

STATUS CODE	DESCRIPTION
`500`	An internal error occurred in the server.

The status codes described in Table E.1 do not cover all possible conditions. See IETF RFC 7231 for a complete list of HTTP status codes.

F

ANNEX F (INFORMATIVE) REVISION HISTORY

F ANNEX F (INFORMATIVE) REVISION HISTORY

Table F.1

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2021-10-06	0.0.9	Charles Heazel	all	SWG review draft.
2024-04-18	0.0.10	Jerome St-Louis	all	SWG review draft.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] John Herring: OGC 06-104r4, *OpenGIS Implementation Specification for Geographic information — Simple feature access — Part 2: SQL option*. Open Geospatial Consortium (2010). https://portal.ogc.org/files/?artifact_id=25354.

[2] Policy SWG: OGC 08-131r3, *The Specification Model — Standard for Modular specifications*. Open Geospatial Consortium (2009). https://portal.ogc.org/files/?artifact_id=34762&version=2.

[3] Policy SWG: OGC 08-131r3, *The Specification Model — Standard for Modular specifications*. Open Geospatial Consortium (2009). https://portal.ogc.org/files/?artifact_id=34762&version=2.

[4] Peter Baumann, Eric Hirschorn, Joan Masó: OGC 09-146r6, *OGC Coverage Implementation Schema*. Open Geospatial Consortium (2017). http://www.opengis.net/doc/IS/cis/1.1.0.

[5] T. Berners-Lee, R. Fielding, L. Masinter: IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*. RFC Publisher (2005). https://www.rfc-editor.org/info/rfc3986.

[6] P. Overell: IETF RFC 5234, *Augmented BNF for Syntax Specifications: ABNF*. RFC Publisher (2008). https://www.rfc-editor.org/info/rfc5234.

[7] ISO: ISO 15836-2:2019, *Information and documentation — The Dublin Core metadata element set — Part 2: DCMI Properties and classes*. International Organization for Standardization, Geneva (2019). https://www.iso.org/standard/71341.html.

[8] ISO: ISO 19101-1:2014, *Geographic information — Reference model — Part 1: Fundamentals*. International Organization for Standardization, Geneva (2014). https://www.iso.org/standard/59164.html.

[9] ISO: ISO 19107:2019, *Geographic information — Spatial schema*. International Organization for Standardization, Geneva (2019). https://www.iso.org/standard/66175.html.

[10] ISO: ISO 19108:2002, *Geographic information — Temporal schema*. International Organization for Standardization, Geneva (2002). https://www.iso.org/standard/26013.html.

[11] ISO: ISO 19111:2019, *Geographic information — Referencing by coordinates*. International Organization for Standardization, Geneva (2019). https://www.iso.org/standard/74039.html.

[12] ISO: ISO 19168-1:2020, *Geographic information — Geospatial API for features — Part 1: Core*. International Organization for Standardization, Geneva (2020). https://www.iso.org/standard/32586.html.

[13] W3C: **Data Catalog Vocabulary (DCAT) — Version 2**, W3C Recommendation, 04 February 2020, https://www.w3.org/TR/vocab-dcat-2/

[14] W3C: **Data on the Web Best Practices**, W3C Recommendation, 31 January 2017, https://www.w3.org/TR/dwbp/

[15] Open Geospatial Consortium: **Welcome To The OGC APIs** [online, viewed 2020-03-16]. Available at https://www.ogcapi.org/.

[16] Open Geospatial Consortium: **OGC Link Relations Registry**, [online, viewed 2024-11-20]. Available at https://www.opengis.net/def/rel

[17] W3C/OGC: **Spatial Data on the Web Best Practices**, W3C Working Group Note, 28 September 2017, https://www.w3.org/TR/sdw-bp/

[18] **YAML Ain't Markup Language** [online, viewed 2020-03-16]. Edited by O. Ben-Kiki, C. Evans, Ingy döt Net. Available at https://yaml.org.

[19] Open Geospatial Consortium: **Compliance Testing Program Policies & Procedures** [online, viewed 2020-04-18]. Available at https://portal.ogc.org/files/?artifact_id=28982.

[20] IANA: **Link Relation Types**, https://www.iana.org/assignments/link-relations/link-relations.xml

[21] Fielding, Roy Thomas: **Architectural Styles and the Design of Network-based Software Architectures**. Doctoral dissertation, University of California, Irvine, 2000, https://www.ics.uci.edu/fielding/pubs/dissertation/fielding_dissertation.pdf