OGC API - Common - Part 2

Geospatial Data

Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: 2021-07-16

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

Internal reference number of this OGC® document: 20-024

Version: 0.0.1

Category: OGC® Implementation Standard

Editor: Charles Heazel

OGC API - Common - Part 2: Geospatial Data

Document type:	OGC® Implementation Standard
Document stage:	Draft
Document language:	English

License Agreement

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

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

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

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

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

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

Table of Contents

Chapter 1. Introduction

i. Abstract

The OGC has extended their suite of standards to include Resource Oriented Architectures and Web APIs. In the course of developing these 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 will 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 an 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 Things. 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 Standard (API-GeoData) is to provide a common connection between the API landing page and resource-specific details. That connection includes metadata which describes the hosted geospatial resources, common parameters for selecting subsets of those resources, and URI templates for identifying the above.

This common connection is sufficient to start the client down the path to resource discovery. Developers of OGC Web API standards extend these first steps with details specific to the resources they intend to expose.

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. Preface

OGC Declaration

iv. Submitting organizations

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

- Heazeltech LLC
- CubeWerx Inc.

v. Submitters

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

Name	**Affiliation**
Chuck Heazel *(editor)*	Heazeltech
Panagiotis (Peter) A. Vretanos	CubeWerx Inc.
others	TBD

Chapter 2. Scope

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

Geospatial resources are typically packaged into sets or collections of related resources. A single API may host a large number of collections. This API-Geodata standard provides a means of organizing these collections and defines operations for discovering and selecting individual collections.

API-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. Additional OGC Web API standards extend this foundation to define resource-specific capabilities.

Chapter 3. Conformance

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

The one Standardization Target for this standard is Web APIs.

API-GeoData defines an API module intended for re-use by other OGC Web API standards. It is anticipated that this standard will only be implemented through inclusion in other standards. Therefore, all the relevant abstract tests in Annex A shall be included or referenced in the Abstract Test Suite for each separate standard that normatively references this standard.

This standard identifies three conformance classes. The conformance classes implemented by an OGC API are advertised through the `/conformance` path on the landing page. Each conformance class is defined by one requirements class. The tests in Annex A are organized by Requirements Class. So an implementation of the *Collections* conformance class must pass all tests specified in Annex A for the *Collections* requirements class.

The requirements classes for API-GeoData are:

- Collections
- Encodings
  - HTML
  - JSON

The *Collections Requirements Class* enables discovery and query access to collections of spatial resources.

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. The *Collections Requirements Class* 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.

The *Collections Requirements Class* enables discovery and query access to collections of spatial resources.

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. The *Collections Requirements Class* 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.

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.

Neither of these encodings is mandatory. An implementor of the *API-GeoData* standard may decide to implement another encoding instead of, or in addition to, these two.

Chapter 4. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- Rescorla, E.: **IETF RFC 2818, HTTP Over TLS**, http://tools.ietf.org/rfc/rfc2818.txt
- Klyne, G., Newman, C.: **IETF RFC 3339, Date and Time on the Internet: Timestamps**, http://tools.ietf.org/rfc/rfc3339.txt
- Berners-Lee, T., Fielding, R., Masinter, L: **IETF RFC 3986, Uniform Resource Identifier (URI): Generic Syntax**, http://tools.ietf.org/rfc/rfc3986.txt
- Greforio, J., Fielding, R., Hadley, M., Nottingham, M., Orchard, D.: **IETF RFC 6570, URI Template**, https://tools.ietf.org/html/rfc6570
- Fielding, R., Reschke, J.: **IETF RFC 7230, Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing**, https://tools.ietf.org/rfc/rfc7230.txt
- Fielding, R., Reschke, JSchaub: **IETF RFC 7231, Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content**, https://tools.ietf.org/rfc/rfc7231.txt
- Fielding, R., Reschke, J.: **IETF RFC 7232, Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests**, https://tools.ietf.org/rfc/rfc7232.txt
- Fielding, R., Reschke, J.: **IETF RFC 7235, Hypertext Transfer Protocol (HTTP/1.1): Authentication**, https://tools.ietf.org/rfc/rfc7235.txt
- Reschke, J.: **IETF RFC 7538, The Hypertext Transfer Protocol Status Code 308 (Permanent Redirect)**, https://tools.ietf.org/rfc/rfc7538.txt
- Bray, T.: **IETF RFC 8259, The JavaScript Object Notation (JSON) Data Interchange Format**, http://tools.ietf.org/rfc/rfc8259.txt* Nottingham, M.: **IETF RFC 8288, Web Linking**, http://tools.ietf.org/rfc/rfc8288.txt
- Nottingham, M.: **IETF RFC 8288, Web Linking**, http://tools.ietf.org/rfc/rfc8288.txt
- ISO 8601-1:2019, **Date and time - Representations for information interchange - Part 1: Basic rules**
- ISO 19111:2019, **Geographic information Spatial referencing by coordinates**
- ISO 15836-2:2019, **Information and documentation — The Dublin Core metadata element set — Part 2: DCMI Properties and classes**
- ISO 19108:2002/Cor 1:2006, **Geographic information — Temporal schema**
- json-schema-org: **JSON Schema,** September 2019, https://json-schema.org/specification.html
- Open API Initiative: **OpenAPI Specification 3.0.3**, https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md
- Whiteside, A., Greenwood, J.: **OGC Web Services Common Standard**, version 2.0, OGC 06-121r9
- Herring, J.: OGC 06-104r4,**OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option**,

http://portal.opengeospatial.org/files/?artifact_id=25354

- van den Brink, L., Portele, C., Vretanos, P.: OGC 10-100r3, **Geography Markup Language (GML) Simple Features Profile**, http://portal.opengeospatial.org/files/?artifact_id=42729
- Heazel, C.: OGC 19-072, **OGC API - Common - Part 1: Core**, http://docs.opengeospatial.org/DRAFTS/19-072.html
- W3C: **HTML5, W3C Recommendation**, http://www.w3.org/TR/html5/
- **Schema.org**: http://schema.org/docs/schemas.html
- W3C Recommendation: **XML Schema Part 2: Datatypes Second Edition**, 28 October 2004, https://www.w3.org/TR/xmlschema-2/

Chapter 5. Terms, Definitions and Abbreviated Terms

5.1. Terms and Definitions

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

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

- **Collection**
  A body of resources that belong or are used together. An aggregate, set, or group of related resources. (OGC 20-024)

- **Coverage**
  feature that acts as a function to return values from its range for any direct position within its spatiotemporal domain, as defined in OGC Abstract Topic 6 (OGC 09-146r6)

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

- **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.

- **Extent**
  The area covered by something. Within this document, we always imply spatial extent; e.g. size or shape that may be expresses using coordinates. (W3C/OGC Spatial Data on the Web Best Practice)

- **Feature**
  abstraction of real world phenomena (ISO 19101-1:2014)

NOTE:	More details about the term `feature` may be found in the W3C/OGC Spatial Data on the Web Best Practice in the section Spatial Things, Features and Geometry.

- **Feature Collection**
  a set of **Features** from a **dataset**

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

- **Resource**
  entity that might be identified (Dublin Core Metadata Initiative - DCMI Metadata Terms)

NOTE:	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.

- **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:	Resource types are re-usable components that are independent of where the resource resides in the API."

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

- **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)

- **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. (ISO 19108)

- **Temporal Position**
  location relative to a temporal reference system (ISO 19108)

- **Temporal Reference System**
  reference system against which time is measured (ISO 19108)

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

- **Temporal Thing**
  Anything with temporal extent, i.e. duration. e.g. the taking of a photograph, a scheduled meeting, a GPS time-stamped track-point. (W3C Basic Geo)

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

- **Web Resource**
  a resource that is identified by an URI.

5.2. 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

Chapter 6. Conventions

All conventions described in the OGC API - Common Part 1: Core Standard are also applicable to this API-Common Part 2: Geospatial Data Standard except where modified in the following section.

6.1. Identifiers

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

All Requirements, Requirement Classes, Conformance Modules, and Conformance Classes 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.

6.2. 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 API-Common GeoData are described in Table 1. Additional relation types may be used if the implementation warrants it.

Table 1. 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).
`http://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]
`http://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]
`http://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

Link Relation	Purpose
item	The target IRI points to a resource that is a member of the collection represented by the context IRI. [IANA]
http://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.

6.3. Geometry

6.3.1. Spatial Geometry

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

The spatial 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.

6.3.2. Temporal Geometry

Standardized concepts are also needed for temporal characteristics. The temporal geometry used in the API-Common Standards is defined in ISO 19108. Only a subset of this geometry is used. Specifically:

- TM_Instant - a point representing position in time
- TM_Period - a one dimensional geometric primitive representing an extent in time and bounded by two different temporal positions (TM_Instant)

6.4. 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 OGC API - Common Geospatial Data standard does not mandate the use of a specific coordinate reference system. However, if no CRS is specified, the following default coordinate reference systems apply for spatial geometries.

- CRS84 - WGS 84 longitude and latitude without height
- CRS84h - WGS 84 longitude and latitude with ellipsoidal height

Temporal geometry is measured relative to an underlying temporal coordinate reference system (TRS). This OGC API - Common - Part 2: Geospatial Data 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 coordinate reference systems may be used where appropriate.

ISO 19111 provides the conceptual model for Coordinate Reference Systems.

6.5. Timezones

A default time zone of UTC as defined by ISO 8601:2004 and ISO 8601:2000 (or Greenwich Mean Time, also referred to as "Z" for Zulu Time), should be used for all temporal data passed or returned to/from OGC Web APIs. ISO 8601 accounts for local time by specifying an offset to UTC. When time zone offsets are used in a temporal element of a client request, the server processing the request should interpret temporal information with respect to the client's requested time zone. When there is no time zone offset expressed in a temporal element, the server should assume a UTC zone. The local time zone of client or server should not be assumed; it shall either be explicitly stated as an offset or assumed to be UTC.

6.6. API definition

6.6.1. General remarks

This OGC standard specifies requirements and recommendations for APIs that share spatial resources and want to follow a standard way of doing so. In general, APIs will go beyond the requirements and recommendations stated in this standard. They will support additional operations, parameters, 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 for every API. In the best case, documentation would be available both in HTML for human consumption and in a machine readable format that can be processed by software for run-time binding. OpenAPI is one way to provide that machine readable documentation.

6.6.2. Role of OpenAPI

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

This standard includes a conformance class for API definitions that follow the OpenAPI specification 3.0. Conformance classes for additional API definition languages will be added as the OGC API landscape continues to evolve.

6.6.3. References to OpenAPI components in normative statements

Some normative statements (requirements, recommendations and permissions) state 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.

6.6.4. 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 http://schemas.opengis.net/ogcapi/common/part1/1.0 and http://schemas.opengis.net/ogcapi/common/part2/1.0

Chapter 7. Overview

This API-Common Geospatial Data Standard provides a common connection between the API landing page and resource-specific specifications or standards.

7.1. Collections

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

The purpose of the OGC API - Common - Part 2: Geospatial Data Standard is to provide a common connection between the API landing page and resource-specific details. That connection includes metadata which describes the hosted resources, common parameters for selecting subsets of the hosted resources, and URI templates for identifing the above.

A contentous issue is the term used to describe an aggregation of resources. The term should be consistent with its' coloquial use, should indicate that the members of the aggregation are somehow associated, and it should be independent of any resource type.

Merriam Websters Dictionary provides a few relevant definitions :

- Collection: "An accumulation of objects gathered for study, comparison, or exhibition or as a hobby."
- Aggregate: (a synonym to collection) "A mass or body of units or parts somewhat loosely associated with one another."
- Set: "A number of things of the same kind that belong or are used together."

Based on these definitions, the term `collection` will be used in this standard to indicate an aggregation of resources. **For purposes of this standard**, a collection is defined as follows:

- Collection: A body of resources that belong or are used together. An aggregate, set, or group of related resources.

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

7.2. Views

A collection of geospatial data may be represented in more than one way. For example, a point cloud may be represented as a collection of Features, or as a Coverage, or as a color-coded map. Each is a representation of the same data. But the access methods and returned data structures are very different. OGC Web API standards refer to these representations as `views`.

`Views` should not be confused with encodings. HTTP content negotiation allows a client to negotiate the encoding (XML, JSON, etc.) to be used for the returned data. Regardless of the encoding, the underlying data model is the same. A `view`, on the other hand, is both a data model and a set of access mechanisms. It is an addressible resource in its own right and must be treated as such.

The API-Common Geodata Standard does not define any `views`. These are defined in separate OGC Web API standards. What is important to understand is how these view-specific standards extend API-Common Geodata.

The URI for a view of a collection folows the URI template:

```
/collections/{collectionId}/{viewId}
```

Where:

1. collectionId = an identifier for the collection
2. viewId = an identifier for the type of view.

So the URIs for the point cloud described above could be:

For Features: `/collections/mycollection/items`

For coverages: `/collections/mycollection/coverage`

For Maps: `/collections/mycollection/maps`

The view identifiers are maintained as a controlled vocabulary by the OGC.

Chapter 8. Requirements Class "Collections"

Requirements Class	
http://www.opengis.net/spec/ogcapi-common-2/1.0/req/collections	
Target type	Web API
Dependency	Requirements Class "Core"
Dependency	ISO 19107
Dependency	GML Simple Features Profile
Dependency	Simple Features for SQL
Dependency	ISO 19108
Dependency	ISO 19111
Dependency	ISO 19108
Dependency	ISO 8601

This Requirements Class describes the resources and operations used to discover and query resource collections exposed through an OGC Web API. It does not include any requirements about how resources are aggregated into collections nor about the aggregated resources themselves. That detail is reserved for resource-specific OGC Web API standards (see Views Section)..

The two resources defined in this Requirements Class are summarized in Table 2. Detailed requirements for each of these resources are provided in the Resource Requirements section.

Table 2. Collection Resources

Resource	URI	HTTP Method	Description
Collections	/collections	GET	Information which describes the set of available Collections
Collection	/collections/{collectionId}	GET	Information about a specific collection of geospatial data with links to distribution

This Requirements Class also describes the operations which can be performed on these resources. Requirements for these operations are included with their associated resource descriptions in the Resource Requirements section.

Three parameters are defined for use with the Collections resource. These parameters subset the set of Collection entries returned based on spatial, temporal, and volumetric filters. These parameters are documented in the Parameter Requirements section. A summary of the parameters is provided in Table 3.

NOTE These parameters may also be used with other resources, but their behavior will be specific for that resource and operation.

Table 3. Parameter Summary

Parameter Name	Target	Description
Bounding Box	Extent	Selects Collection entries which have an Extent element that intersects the bounding box
Date-Time	Extent	Selects Collection entries which have an Extent element that intersects the specified time period
Limit	The result set	Limits the number of Collection entries which can be returned in one response

Finally, requirements which have general applicablity are provided in the General Requirements section.

8.1. Resource Requirements

This section expresses the requirements for resources and operations used to discover and query resource collections.

8.1.1. Collections

OGC APIs typically organize their Spatial Resources into collections. Information about those collections is accessed through the `/collections` path.

Operation

Requirement 1	**/req/collections/rc-md-op**
A	The API SHALL support the HTTP GET operation at the path `/collections`.

Response

Requirement 2	**/req/collections/rc-md-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The content of that response SHALL be based upon the JSON schema collections.json.

The collections response returned by this operation is based on the collections.json JSON schema. Examples of collections responses are provided in Collections Response Example.

collections.json

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "Collections Schema",
    "description": "This schema defines the resource returned from /collections
path.",
    "type": "object",
    "required": [
        "links",
        "collections"
        ],
    "properties": {
        "links": {
            "type": "array",
            "items": {"$href": "link.json"}
            },
        "timeStamp": {
            "type": "string",
            "format": "date-time"
            },
        "numberMatched": {
            "type": "integer",
            "min": "0"
            },
        "numberReturned": {
            "type": "integer",
            "min": "0"
            },
        "collections": {
            "type": "array",
            "items": {"$href": "collectionDesc.json"}
            }
        }
    }
```

This 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	**/req/collections/rc-md-links**

A	A `200`-response SHALL include the following links in the `links` property of the response: • A link to this response document (relation: `self`), • A link to the response document in every other media type supported by the API (relation: `alternate`).
B	All links SHALL include the `rel` and `type` link parameters.

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

Recommendation 1	**/rec/collections/rc-md-descriptions**
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 `timeStamp` property of the Collections response indicates when the response was generated.

Requirement 4	**/req/collections/rc-timeStamp**
A	If a property `timeStamp` is included in the response, the value SHALL be set to the time stamp when the response was generated.

The `collections` property of the Collections response provides a description of each individual collection hosted by the API. These descriptions are based on the Resource Collection Schema. This schema is described in detail in the Resource Collection section of this Standard.

Requirement 5	**/req/collections/rc-md-items**
A	For each spatial resource collection accessible through this API, metadata describing that collection SHALL be provided in the `collections` property of the Collections response.
B	This metadata shall be based on the same schema as the Collection resource.

A client may wish to select a subset of the hosted collections using HTTP query parameters. The

Simple Query Conformance Class defines the `bbox` and the `datetime` parameter for that purpose. It also defines how a server should process those parameters. These parameters are evaluated against the `extent` element of each Collection item in the Collections response.

The requirements governing the processing of these parameters are:

Requirement 6	**/req/collections/rc-bbox-collections-unsupported**
A	If the `bbox` parameter is provided by the client but it is not supported by the server, then the server SHALL process the request as if the parameter had not been provided.

Requirement 7	**/req/collections/rc-time-collections-unsupported**
A	If the `datetime` parameter is provided by the client but it is not supported by the server, then the server SHALL process the request as if the parameter had not been provided.

The Simple Query Conformance Class also defines a parameter to limit the number of collections returned in a response. This `limit` parameter indicates the maximum number of collections which should be included in a single response.

Requirement 8	**/req/collections/rc-limit-collections-unsupported**
A	If the `limit` parameter is provided by the client but it is not supported by the server, then the server SHALL process the request as if the parameter had not been provided.

NOTE	This requirement does not preclude server imposed limits on the size of a response.

Error situations

See HTTP Status Codes for general guidance.

8.1.2. Resource Collection

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

Operation

Requirement 9	**/req/collections/src-md-op**

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

Response

Requirement 10	**/req/collections/src-md-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The content of that response SHALL be based upon the JSON schema collectionInfo.json.
C	The content of that response SHALL be consistent with the content for this collection in the `/collections` response. That is, the values for `id`, `title`, `description` and `extent` SHALL be identical.

Collection responses are based on the Resource Collection Schema. Examples of Collection responses are provided in Collection Object Examples.

Collection Resource Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "Collection Resource Schema",
    "description": "This schema defines the resource returned from
/collections/{collectionId}.",
    "type": "object",
    "required": [
        "id",
        "links"
        ],
    "properties": {
        "id": {
            "description": "identifier of the collection used, for example, in URIs",
            "type": "string"
            },
        "title": {
            "description": "human readable title of the collection",
            "type": "string"
            },
        "description": {
            "description": "a description of the members of the collection",
```

```
            "type": "string"
            },
        "attribution" : {
            "type" : "string",
            "title" : "attribution for the collection",
            "description" : "The `attribution` 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."
            },
        "links": {
            "type": "array",
            "items": {"$href": "link.json"}
            },
        "extent": {"$href": "extent.json"},
        "itemType": {
            "description": "An indicator about the type of the items in the
collection.",
            "type": "string"
            },
        "crs": {
            "description": "the list of coordinate reference systems supported by the
API; the first item is the default coordinate reference system",
            "type": "array",
            "items": {
                "type": "string"
                },
            "default": [
                "http://www.opengis.net/def/crs/OGC/1.3/CRS84"
                ],
            "example": [
                "http://www.opengis.net/def/crs/OGC/1.3/CRS84",
                "http://www.opengis.net/def/crs/EPSG/0/4326"
                ]
            }
        }
    }
```

The purpose of the Collection resource is to provide clients with a basic understanding of a single collection. It also serves as a starting point for further navigation through the collection. Many of the properties of the Collection resource are self-explanitory. However, a few call for additional explanation.

Attribution

The attribution element is a special type of string property. Specifically, it can contain markup text. Markup allows a text string to import images and format text. The capabilities are only limited by the markup language used. See the example collection response for an example of the use of markup in the attribution element.

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 2	**/rec/collections/rc-md-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`).

Links

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

Requirement 11	**/req/collections/rc-md-items-links**
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` and `type` properties.

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

Recommendation 3	**/rec/collections/rc-md-items-descriptions**
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.

Extent

The extent property defines a spatial-temporal surface which 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.

Requirement 12	**/req/collections/rc-md-extent**
A	For each spatial collection resource, the `extent` property, if provided, SHALL define boundaries that encompass the spatial and temporal properties of all of the resources in this collection. The temporal extent may use `null` values to indicate an open time interval.
B	If a spatial resource has multiple properties with spatial or temporal information, it is the decision of the API implementation whether only a single spatial or temporal geometry property is used to determine the extent or all relevant geometries.

Recommendation 4	**/rec/collections/rc-md-extent**
A	If an extent contains multiple spatial boundaries (multiple bbox, etc.), then the extent SHOULD include in the first bbox a boundary which represents the union of all of the other boundaries.
B	If an extent contains multiple temporal intervals, then the extent SHOULD include as the first interval an interval which represents the union of all of the other intervals.

Recommendation 5	**/rec/collections/rc-md-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	**/per/collections/rc-md-extent-extensions**

A	API-Common 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.
B	API-Common only supports • Spatial extents in CRS84 or CRS84h and • Temporal extents in the Gregorian calendar These are the only *enum* values in extent.yaml).
C	Extensions MAY add additional reference systems to the `extent` object.

Error situations

See HTTP Status Codes for general guidance.

If the parameter `collectionId` does not exist on the server, the status code of the response will be `404` (see Table 4).

8.2. General Requirements

The following requirements and recommendations are applicable to all OGC Web APIs.

8.2.1. HTTP 1.1

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

Requirement 13	**/req/core/http**
A	OGC Web APIs SHALL conform to HTTP 1.1.
B	If the API supports HTTPS, then the API SHALL also conform to HTTP over TLS.

8.2.2. HTTP Status Codes

Table 4 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 4. 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.
500	An internal error occurred in the server.

The return status codes described in Table 4 do not cover all possible conditions. See IETF RFC 7231 for a complete list of HTTP status codes.

Permission 2	/per/core/additional-status-codes
A	Servers MAY implement additional capabilities provided by the HTTP protocol. Therefore, they MAY return status codes in addition to those listed in Table 4.

When a server encounters an error in the processing of a request, the server may wish to include information in addition to the status code in the response. Since Web API interactions are often machine-to-machine, a machine-readable report would be preferred. IETF RFC 7807 addresses this

need by providing "Problem Details" response schemas for both JSON and XML.

Recommendation 6	**/rec/core/problem-details**
An OGC Web API should include a "Problem Details" report in any error response in accordance with IETF RFC 7807.	

8.2.3. Query parameters

Parameter Names

Requirement 14	**/req/core/query-param-name-unknown**
A	The server SHALL respond with a response with the status code `400`, IF 1. the request URI includes a query parameter that is not specified in the API definition 2. /per/core/query-param-specified does not apply, and 3. /per/core/query-param-tolerance does not apply.

The criteria for a parameter to be "specified" in the API definition depends on the API definition language used, the complexity of the resources exposed, and the ability of the API server to tolerate errors.

A service implementer should endeavor to provide as much detail in the server's API definition as the API definition language allows. However, there is no requirement for the API definition to list every endpoint for which there is a non-404 behavior, for it to list every possible query parameter that might affect the behavior of an endpoint, or for it to list every possible value that each query parameter might accept.

Permission 3	**/per/core/query-param-name-specified**
A	The specification of a query parameter in the API definition MAY encompass a range of parameter names. Any query parameter that falls within the specified range can be considered "specified" in the API definition. Examples of a parameter range include: • A regular expression that defines the valid parameter names, • A URI Template segment that defines the valid parameter names, • An indication that all parameter names are accepted (no parameter validation).

B	The API definition language chosen may not be capable of expressing the desired range of values. In that case the server SHOULD provide: • A definition of the parameter range that best expresses the intended use of that parameter, • Additional human readable text documenting the actual range of validity.

Permission 4	**/per/core/query-param-name-tolerance**
A	Servers MAY display tolerance for requests with incorrect query parameter names. These acts of tolerance include: • Accept alternate capitalizations, spellings, and/or aliases of parameters, • Ignore unknown/unrecognized parameters, • Return a response with a status code of 30x redirecting the client to a more correct version of the request.
B	Servers should not be excessively tolerant. The response a client receives from the server should be a reasonable response for the request submitted.

Parameter Values

Requirement 15	**/req/core/query-param-value-invalid**
A	The server SHALL respond with a response with the status code `400`, IF 1. the request URI includes a query parameter that has an invalid value and 2. /per/core/query-param-value-tolerance does not apply.

The criteria for a parameter value to be considered "invalid" varies with the expressiveness of the API definition language used and the ability of the API server to tolerate errors.

A service implementer should endeavour to provide as much detail in the server's API definition as the API definition language allows. However, there is no requirement to list every possible value that each query parameter might accept. Rather, the API implementation should include a reasonable degree of error recovery.

Permission 5	/per/core/query-param-value-tolerance
A	Servers MAY display tolerance for requests where a parameter has an incorrect value. These acts of tolerance include: • substituting default values for invalid ones, • correcting formatting errors (e.g. convert integers to float).
B	Servers should not be excessively tolerant. The response a client receives from the server should be a reasonable response for the request submitted.

8.2.4. Web Caching

Entity tags are a mechanism for web cache validation and for supporting conditional requests to reduce network traffic. Entity tags are specified by HTTP 1.1 (RFC 7232).

Recommendation 7	/rec/core/etag
A	The service SHOULD support entity tags and the associated headers as specified by HTTP 1.1.

8.2.5. Support for Cross-Origin Requests

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

Recommendation 8	/rec/core/cross-origin
A	If the server is intended to be accessed from a browser, cross-origin requests SHOULD be supported. Note that support can also be added in a proxy layer on top of the server.

Two common mechanisms to support cross-origin requests are:

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

8.2.6. String Internationalization

If the server supports representing resources in multiple languages, the usual HTTP content negotiation mechanisms apply. The client states its language preferences in the `Accept-Language` header of a request and the server responds with responses that have linguistic text in the language

that best matches the requested languages and the capabilities of the server.

Recommendation 9	**/rec/core/string-i18n**
A	For encodings that support string internationalization, the server SHOULD include information about the language for each string value that includes linguistic text.

For example, if JSON-LD is used as an encoding, the built-in capabilities to annotate a string with its language should be used.

The link object based on RFC 8288 (Web Linking) includes a `hreflang` attribute that can be used to state the language of the referenced resource. This can be used to include links to the same data in, for example, English or French. Just like with multiple encodings, a server that wants to use language-specific links will have to support a mechanism to mint language-specific URIs for resources in order to express links to, for example, the same resource in another language. Again, this document does not mandate any particular approach how such a capability is supported by the server.

8.2.7. Resource Encodings

A Web API provides access to resources through representations of those resources. One property of a representation is the format used to encode it for transfer. Components negotiate the encoding format to use through the content negotiation process defined in IETF RFC 7231.

Additional content negotiation techniques are allowed, but support is not required of implementations conformant to this Standard.

While this Standard does not specify any mandatory encoding, the following encodings are recommended:

HTML encoding recommendation:

Recommendation 10	**/rec/core/html**
A	To support browsing an API with a web browser and to enable search engines to crawl and index the dataset, implementations SHOULD consider supporting an HTML encoding.

JSON encoding recommendation:

Recommendation 11	**/rec/core/json**
A	To support processing of an API with a web applet, implementations SHOULD consider supporting a JSON encoding.

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

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

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

As clients simply need to dereference the URI of the link, the implementation details and the mechanism how the encoding is included in the URI of the link are not important. Developers interested in the approach of a particular implementation, such as, to manipulating ("hacking") URIs in the browser address bar, can study the API Definition document for that server.

Two common approaches are to use:

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

8.2.8. Parameter Encoding

The following sections provide the requirements and guidelines for encoding parameters for use in an OGC Web API request.

OGC Web API requests are issued using a Uniform Resource Identifier (URI). The URI syntax is defined in IETF RFC 3986. Rules for building URI Templates can be found in IETF RFC 6570.

The Backus-Naur Form (BNF) definition of a URI is provided in Annex F.

Capitalization

IETF RFC 3986 sections 6.2.2.1 and 2.1 provide the requirements for capitalization in URIs.

Requirement 16	**/req/core/query-param-capitalization**
A	Parameter names and values SHALL be case sensitive.
B	IF a parameter name or value includes a percent encoded (escaped) character, THEN the upper case hexadecimal digits ("A" through "F") of that percent encoded character SHALL be equivalent to the lower case digits "a" through "f" respectively.

In order to minimize capitalization issues for implementers of OGC Web API standards:

Recommendation 12	**/rec/core/query-param-capitalization**
A	Query parameter names SHOULD be in Kebab case.
B	Query parameter values are usually reflective of the internal structure of the target resource. Unless otherwise specified, these values SHOULD be in kebab case.

A Web API may allow filtering on properties of the target resource. In that case, the parameter name would be the name of the resource property. These names are defined by the standards and specifications defining the resource and cannot be constrained by this Standard.

Parameter Value Lists

Parameters may pass more than one value. These lists of parameter values may be passed in two ways.

1. Repeated name:value pairs where the parameter name is repeated for each value in the list
2. A parameter name followed by a delimited list of values.

The following requirements define how to encode a delimited list (case 2) of parameter values. They do not apply if replication (case 1) is used.

Requirement 17	**/req/core/query-param-list-delimiter**
A	Parameters values containing lists SHOULD specify the delimiter to be used in the API definition.
B	The default list item delimiter SHALL be the comma (",").

Requirement 18	**/req/core/query-param-list-escape**
A	Any list item values that include a space or comma SHALL escape the space or comma character using the URI encoding rules from IETF RFC 3986

Requirement 19	**/req/core/query-param-list-empty**
A	All empty entries SHALL be represented by the empty string.

Thus, two successive commas indicates an empty item, as does a leading comma or a trailing comma. An empty list can either be interpreted as a list containing no items or as a list containing a

single empty item, depending on the context.

Numeric and Boolean Values

The Geospatial field is a mathematical discipline. A clear and accurate exchange of mathematical values is essential. The encoding rules in this section standardize the encoding of numeric and Boolean primitives when included in a URI. These rules are based on the computer science basic data types identified by Kernighan and Ritchie.

Boolean values conform to the following requirement.

Requirement 20	**/req/core/query-param-value-boolean**
A	Boolean values shall be represented by the lowercase strings "true" and "false", representing Boolean true and false respectively.

Integer values conform to the following requirement.

Requirement 21	**/req/core/query-param-value-integer**
A	Integer values SHALL be represented by a finite-length sequence of decimal digits with an optional leading negative "-" sign. Positive values are assumed if the leading sign is omitted.

Real numbers can be represented using either the decimal or double (exponential) format. The decimal format is typically used except for very large or small values.

Decimal values conform to the following requirement.

Requirement 22	**/req/core/query-param-value-decimal**
A	Decimal values SHALL be represented by a finite-length sequence of decimal digits separated by a period as a decimal indicator. • An optional leading negative sign ("-") is allowed. • If the sign is omitted, positive ("+") is assumed. • Leading and trailing zeroes are optional. • If the fractional part is zero, the period and following zero(es) can be omitted.

Double values conform to the following requirement.

Requirement 23	**/req/core/query-param-value-double**

A	Double values SHALL be represented by a mantissa followed, optionally, by the character "e", followed by an exponent.
B	The exponent SHALL be an integer.
C	The mantissa SHALL be a decimal number.
D	The representations for exponent and mantissa SHALL follow the lexical rules for integer and decimal.
E	If the "e" and the following exponent are omitted, an exponent value of 0 SHALL be assumed.

Special values conform to the following requirement.

Requirement 24	**/req/core/query-param-value-special**
A	The special values positive and negative infinity and not-a-number SHALL be represented using the strings `inf`, `-inf` and `nan`, respectively.

Chapter 9. Requirements Class "Simple Query"

Requirements Class	
http://www.opengis.net/spec/ogcapi-common-2/1.0/req/simple-query	
Target type	Web API
Dependency	Requirements Class "Collections"
Dependency	ISO 19107
Dependency	GML Simple Features Profile
Dependency	Simple Features for SQL
Dependency	ISO 19108
Dependency	ISO 19111
Dependency	ISO 19108
Dependency	ISO 8601

This Requirements Class describes the resources and operations used to discover and query resource collections exposed through an OGC Web API. It does not include any requirements about how resources are aggregated into collections nor about the aggregated resources themselves. That detail is reserved for resource-specific OGC Web API standards (see Views Section).

9.1. Parameter Requirements

NOTE

The definitions and requirements in this section are not specific to any resource. They could be moved to a registry and then referenced or imported from there.

Query parameters are used in URIs to limit the resources which are returned on a GET request. The OGC API - Common - Part 2: Geospatial Data Standard defines three query parameters for use in OGC API standards:

- bbox: Bounding Box
- datetime: Date and Time
- limit: Response resource count limit

The behavior generated by these parameters is specific to the operation and resource upon which they are applied. Those behaviors are described for each resource type and operation in the Resource Requirements section.

Use of these query parameters with any specific operation is optional. Developers of API-GeoData servers should document their supported parameters in the API definition as describe in API-Core.

9.1.1. Parameter bbox

The `bbox` parameter is used to select resources based on the geospatial footprint or extent.

The `bbox` parameter is defined as follows:

Requirement 25	**/req/collections/rc-bbox-definition**
A	The bounding box is provided as four or six numbers, depending on whether the coordinate reference system includes a vertical axis (height or depth): • Lower left corner, coordinate axis 1 • Lower left corner, coordinate axis 2 • Minimum value, coordinate axis 3 (optional) • Upper right corner, coordinate axis 1 • Upper right corner, coordinate axis 2 • Maximum value, coordinate axis 3 (optional)
B	The values for the CRS axis 1 and 2 SHALL be interpreted as WGS84 longitude/latitude (http://www.opengis.net/def/crs/OGC/1.3/CRS84) unless a different coordinate reference system is specified in a parameter `bbox-crs`.
C	The coordinate values SHALL be within the extent specified for the coordinate reference system.

D	The `bbox` parameter SHALL possess the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: bbox`
	`in: query`
	`required: false`
	`schema:`
	`  type: array`
	`  minItems: 4`
	`  maxItems: 6`
	`  items:`
	`    type: number`
	`style: form`
	`explode: false`

While the processing of the `bbox` parameter is specific to the resource and operation for which it is applied, there is a general set of requirements which all implementations must address.

Requirement 26	/req/collections/rc-bbox-response
A	If the `bbox` parameter is provided by the client and supported by the server, then only resources that have a spatial geometry that intersects the bounding box SHALL be part of the result set.
B	If a resource has multiple spatial geometry properties, it is the decision of the server whether only a single spatial geometry property is used to determine the extent or all relevant geometries.
C	The `bbox` parameter SHALL also match all resources in the collection that are not associated with a spatial geometry.

"Intersects" means that a coordinate that is part of the spatial geometry of the resource falls within the area specified in the parameter `bbox`. This includes the boundaries of the geometries. For curves the boundary includes the start and end position. For surfaces the boundary includes the outer and inner rings.

In case of a degenerate bounding box, the resulting geometry is used. For example, if the lower left corner is the same as the upper right corner, all resources match where the geometry intersects with this point.

This standard does not specify requirements for the parameter `bbox-crs`. Those requirements will be specified in a later version of this standard.

For WGS 84 longitude/latitude the bounding box is in most cases the sequence of minimum longitude, minimum latitude, maximum longitude and maximum latitude. However, in cases where the box spans the anti-meridian (180th meridian) the first value (west-most box edge) is larger than the third value (east-most box edge).

Example 1. The bounding box of the New Zealand Exclusive Economic Zone

> The bounding box of the New Zealand Exclusive Economic Zone in WGS84 (from 160.6°E to 170°W and from 55.95°S to 25.89°S) would be represented in JSON as `[ 160.6, -55.95, -170, -25.89 ]` and in a query as `bbox=160.6,-55.95,-170,-25.89`.

Note that the server will return an error if a latitude value of `160.0` is used.

If the vertical axis is included, the third and the sixth number are the bottom and the top of the 3-dimensional bounding box.

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

9.1.2. Parameter datetime

The `datetime` parameter selects resources based on their temporal extent. The definition of temporal extent is specific to the resource type being filtered.

The `datetime` parameter is defined as follows:

Requirement 27	**/req/collections/rc-time-definition**

A	Temporal geometries are either a date-time value or a time interval. The parameter value SHALL conform to the following syntax (using ABNF):
	`interval-closed = date-time "/" date-time`
	`interval-open-start = "../" date-time`
	`interval-open-end = date-time "/.."`
	`interval = interval-closed / interval-open-start / interval-open-end`
	`datetime = date-time / interval`

B	The syntax of `date-time` is specified by RFC 3339, 5.6.
C	Open ranges in time intervals at the start or end SHALL be supported using a double-dot (`..`).

D	The `datetime` parameter SHALL have the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: datetime`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	`style: form`
	`explode: false`

While the processing of the `datetime` parameter is specific to the resource and operation for which it is applied, there is a general set of requirements which all implementations must address.

Requirement 28	**/req/collections/rc-time-response**

A	If the `datetime` parameter is provided by the client and supported by the server, then only resources that have a temporal geometry that intersects the temporal information in the `datetime` parameter SHALL be part of the result set.
B	If a resource has multiple temporal properties, the API implementor decides whether only a single temporal property is used to determine the extent or all relevant temporal properties.
C	The `datetime` parameter SHALL match all resources in the collection that are not associated with a temporal geometry.

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

Note	ISO 8601-2 distinguishes open start/end timestamps (double-dot) and unknown start/end timestamps (empty string). For queries, an unknown start/end has the same effect as an open start/end.

Example 2. A date-time

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

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

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

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

Example 3. Intervals

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

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

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

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

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

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

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

9.1.3. Parameter limit

The `limit` parameter limits the number of resources that can be returned in a single response.

Requirement 29	**/req/collections/rc-limit-definition**

A	The operation SHALL support a parameter `limit` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: limit`
	`in: query`
	`required: false`
	`schema:`
	`  type: integer`
	`  minimum: 1`
	`  maximum: 10000`
	`  default: 10`
	`style: form`
	`explode: false`

Note:	The values for `minimum`, `maximum` and `default` are only examples and MAY be changed.

While the processing of the `limit` parameter is specific to the resource and operation for which it is applied, there is a general set of requirements which all implementations must address.

Requirement 30	**/req/collections/rc-limit-response**
A	If the `limit` parameter is provided by the client and supported by the server, then the response SHALL not contain more resources than specified by the `limit` parameter.
B	If the API definition specifies a maximum value for the `limit` parameter, the response SHALL not contain more resources than this maximum value.
C	Only items are counted that are on the first level of the collection. Any nested objects contained within the explicitly requested items SHALL not be counted.

The number of resources returned depends on the server and the value of the `limit` parameter.

- The client can request a limit to the number of resources returned.
- The server may have a default value for the limit, and a maximum limit.

- If the server has any more results available than it returns (the number it returns is less than or equal to the requested/default/maximum limit) then the server will include a link to the next set of results.

Permission 6	**/per/collections/rc-server-limit**
A	If a server is configured with a maximum response size, then the server MAY page responses which exceed that threshold.

Recommendation 13	**/rec/collections/rc-server-limit**
A	Clients SHOULD be prepared to handle a paged response even if they have not specified a `limit` parameter in their query.

The effect of the limit parameter is to divide the response into a number of pages. Each page (except for the last) contains the specified number of entities. The response contains the first page. Additional pages can be accessed through hyperlink navigation.

Recommendation 14	**/rec/collections/rc-next-1**
A	A `200`-response SHOULD include a link to the next "page" (relation: `next`), if more resources have been selected than returned in the response.

Recommendation 15	**/rec/collections/rc-next-2**
A	Dereferencing a `next` link SHOULD return additional resources from the set of selected resources that have not yet been returned.

Recommendation 16	**/rec/collections/rc-next-3**
A	The number of resources in a response to a `next` link SHOULD follow the same rules as for the response to the original query and again include a `next` link, if there are more resources in the selection that have not yet been returned.

Providing `prev` links supports navigating back and forth between pages, but depending on the implementation approach it may be too complex to implement.

Permission 7	**/per/collections/rc-prev**

A	A response to a `next` link MAY include a `prev` link to the resource that included the `next` link.

9.2. Target Resource Requirements

The target of the parameters defined in this conformance class is the Collection resource described in the Collections Requirements Class. The purpose of these parameters is to select a subset of Collection resources to be included in the response to a /collections request.

Three parameters are defined for use with the Collections resource. These parameters subset the set of Collection entries returned based on spatial, temporal, and volumetric filters. These parameters are documented in the Parameter Requirements section.

The `collections` property of the Collections response provides a description of each individual collection hosted by the API. These descriptions are based on the Resource Collection Schema. This schema is described in detail in the Resource Collection section of this Standard.

A client may select a subset of the hosted collections using the `bbox` and the `datetime` parameter. These parameters are evaluated against the `extent` element of each Collection item in the Collections response.

The requirements governing the processing of these parameters are:

Requirement 31	**/req/collections/rc-bbox-collections-response**
A	The `bbox` parameter SHALL be evaluated against the `bbox` element of the `extent` property of all `Collection` resources hosted by the API server.
B	Only `Collection` resources whose `extent` property intersects the geometry specified by the `bbox` parameter SHALL be included in the `Collections` response.
C	The `bbox` parameter SHALL also match all collections that do not have an `extent` element with a spatial geometry.

Requirement 32	**/req/collections/rc-time-collections-response**
A	The `datetime` parameter SHALL be evaluated against the `interval` element of the `extent` property of all `Collection` resources hosted by the API server.
B	Only `Collection` resources whose `extent` property intersects the temporal interval specified by the `datetime` parameter SHALL be included in the `Collections` response.

C	The `datetime` parameter SHALL also match all collections that do not have an `extent` element with a temporal geometry.

The client may further reduce the number of collections returned in a response by using the `limit` parameter. When applied against the `/collections` resource, the `limit` parameter indicates the maximum number of collections which should be included in a single response.

Requirement 33	**/req/collections/rc-limit-collections-response**
A	If the `limit` parameter is provided by the client, then the `collections` element of the collections response SHALL not contain more items than specified by the `limit` parameter.
B	If the API definition specifies a maximum value for the `limit` parameter, the `collections` element of the collections response SHALL not contain more items than this maximum value.

If the collections response does not contain all of the collection resources available from this server, then the client must be informed or that fact.

Requirement 34	**/req/collections/rc-subset-response**
A	If the number of items in the `collections` element is less than the number available through the API, then the `numberMatched` and `numberReurned` properties SHALL be included in the Collections response.

The `numberMatched` property of the Collections response indicates the number of Collection items included in the Collections response. This may be a subset of the total set of collections hosted by the API. Selection of which collections to include in a subset is controled through the `bbox`, `datetime` and other selection parameters provided by the client.

Requirement 35	**/req/collections/rc-numberMatched**
A	If a property `numberMatched` is included in the response, the value SHALL be identical to the number of hosted collections that meet the selection parameters provided by the client.
B	A server MAY omit this information in a response, if the information about the number of matching resources is not known or difficult to compute.

The Collections response should describe all of the collections that meet the client's selection criteria. However, in some cases that is impractical. An API implementation has the option of

limiting the size of the Collections response.

Permission 8	**/per/collections/rc-md-items**
A	To support servers with many collections, servers MAY limit the number of items included in the `collections` property.

The number of collection items included in a Collections response may be a subset of the number matched. In that case, the `numberReturned` property of the Collections response indicates the number of collection items returned in this "page" of the Collections response.

Requirement 36	**/req/collections/rc-numberReturned**
A	If a property `numberReturned` is included in the response, the value SHALL be identical to the number of items in the `collections` array in the `Collections` document.
B	A server MAY omit this information in a response, if the information about the number of resources in the response is not known or difficult to compute.

If the Collections response contains a subset of the selected collection items (numberReturned is less than numberMatched) then the Collections response should contain links for navigating to the rest of the collection items as described in the limit parameter section.

Chapter 10. Encoding Requirements Classes

10.1. Overview

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

- HTML
- JSON

None of these encodings are mandatory. An implementation of the Collections requirements class may implement either, both, or none of them.

10.2. Requirement Class "HTML"

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

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

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

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

Requirements Class	
http://www.opengis.net/spec/ogcapi_common-2/1.0/req/html	
Target type	Web API
Dependency	Requirements Class "Collections"
Dependency	HTML5
Dependency	Schema.org

Requirement 37	**/req/html/definition**
A	Every `200`-response of an operation of the API SHALL support the media type `text/html`.

Requirement 38	**/req/html/content**

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 17	**/rec/html/schema-org**
A	A `200`-response with the media type `text/html`, SHOULD include Schema.org annotations.

10.3. Requirement Class "JSON"

JSON is a text syntax that facilitates structured data interchange between programming languages. It 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	
http://www.opengis.net/spec/ogcapi_common-2/1.0/req/json	
Target type	Web API
Dependency	Requirements Class "Collections"
Dependency	IETF RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format
Dependency	JSON Schema

Requirement 39	**/req/json/definition**
A	`200`-responses of the server SHALL support the `application/json` media type.

Requirement 40	**/req/json/content**
A	Every `200`-response with the media type `application/json` SHALL include, or link to, a payload encoded according to the JSON Interchange Format.

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.

Chapter 11. Media Types

JSON media types that would typically be supported by a server that supports JSON are:

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

XML media types that would typically be supported by a server that supports XML are:

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

The typical HTML media type for all "web pages" in a server would be `text/html`.

Chapter 12. Security Considerations

See OGC API - Common - Part 1: Core, Clause 11.

add additional text as needed

Annex A: Abstract Test Suite (Normative)

A.1. Introduction

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

A.2. Conformance Class Collections

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

A.2.1. General Tests

CRS 84

Abstract Test 1	**/ats/collections/crs84**
Test Purpose	Validate that all spatial geometries provided through the API are in the CRS84 or CRS84h spatial reference system unless otherwise requested by the client.
Requirement	/req/collections/crs84
Test Method	1. Do not specify a coordinate reference system in any request. All spatial data should be in the default coordinate reference system. 2. If the retrieved spatial data includes elevations, validate that the data is in the CRS84h reference system. 3. If the retrieved spatial data does not include elevations, validate that the data is in the CRS84 reference system.

A.2.2. Feature Collections {root}/collections

Abstract Test 2	/ats/collections/rc-md-op
Test Purpose	Validate that information about the Collections can be retrieved from the expected location.
Requirement	/req/collections/rc-md-op
Test Method	1. Issue an HTTP GET request to the URL {root}/collections 2. Validate that a document was returned with a status code 200 3. Validate the contents of the returned document using test /ats/collections/rc-md-success.

Abstract Test 3	/ats/collections_rc-md-success
Test Purpose	Validate that the Collections content complies with the required structure and contents.
Requirement	/req/collections/rc-md-success, /req/collections/crs84
Test Method	1. Validate that all response documents comply with /ats/collections/rc-md-links 2. In case the response includes a "crs" property, validate that the first value is either: "http://www.opengis.net/def/crs/OGC/1.3/CRS84" or "http://www.opengis.net/def/crs/OGC/0/CRS84h" 3. Validate the collections content for all supported media types using the resources and tests identified in Table 5

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

Table 5. Schema and Tests for Collections content

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

A.2.3. Feature Collection {root}/collections/{collectionId}

Abstract Test 4	/ats/collections/src-md-op

Test Purpose	Validate that the Collection content can be retrieved from the expected location.
Requirement	/req/collections/src-md-op
Test Method	For every Feature Collection described in the Collections content, issue an HTTP GET request to the URL `/collections/{collectionId}` where `{collectionId}` is the `id` property for the collection. 1. Validate that a Collection was returned with a status code 200 2. Validate the contents of the returned document using test /ats/collections/src-md-success.

Abstract Test 5	**/ats/collections/src-md-success**
Test Purpose	Validate that the Collection content complies with the required structure and contents.
Requirement	/req/collections/src-md-success
Test Method	Verify that the content of the response is consistent with the content for this Resource Collection in the `/collections` response. That is, the values for `id`, `title`, `description` and `extent` are identical.

A.2.4. Second Tier Tests

These tests are invoked by other tests.

Extent

Abstract Test 6	**/ats/coollections/rc-md-extent**
Test Purpose	Validate that the `extent` property if it is present
Requirement	/req/collections/rc-md-extent
Test Method	1. Verify that the `extent` provides bounding boxes that include all spatial geometries 2. Verify that if the `extent` provides time intervals that include all temporal geometries in this collection. 3. A temporal extent of `null` indicates an open time interval.

Items

Abstract Test 7	**/ats/collections/rc-md-items**
Test Purpose	Validate that each collection provided by the server is described in the Collections Metadata.
Requirement	/req/collections/rc-md-items
Test Method	1. Verify that there is an entry in the `collections` array of the Collections Metadata for each feature collection provided by the API. 2. Verify that each collection entry includes an identifier. 3. Verify that each collection entry includes links in accordance with /collections/rc-md-items-links. 4. Verify that if the collection entry includes an extent property, that that property complies with /collections/rc-md-extent 5. Validate each collection entry for all supported media types using the resources and tests identified in Table 6

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

Table 6. Schema and Tests for Collection Entries

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

Abstract Test 8	**/ats/collections/rc-md-items-links**
Test Purpose	Validate that each Feature Collection metadata entry in the Collections Metadata document includes all required links.
Requirement	/req/collections/rc-md-items-links

Test Method	1. Verify that each Collection item in the Collections Metadata document includes a `link` property for each supported encoding. 2. Verify that the `links` properties of the collection includes an item for each supported encoding with a link to the features resource (relation: `items`). 3. Verify that all links include the `rel` and `type` link parameters.

Links

Abstract Test 9	**/ats/collections/rc-md-links**
Test Purpose	Validate that the required links are included in the Collections Metadata document.
Requirement	/req/collections/rc-md-links
Test Method	Verify that the response document includes: 1. a link to this response document (relation: `self`), 2. a link to the response document in every other media type supported by the server (relation: `alternate`). Verify that all links include the `rel` and `type` link parameters.

A.3. Conformance Class GeoJSON

Conformance Class	
http://www.opengis.net/spec/ogcapi-common-2/1.0/conf/json	
Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi_common-2/1.0/req/json
Dependency	Conformance Class "OAPI Core"

A.3.1. GeoJSON Definition

Abstract Test 10	**/ats/json/definition**
Test Purpose	Verify support for JSON
Requirement	/req/json/definition

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.3.2. GeoJSON Content

Abstract Test 11	**/ats/json/content**
Test Purpose	Verify the content of a JSON document given an input document and schema.
Requirement	/req/json/content
Test Method	1. Validate that the document is a JSON document. 2. Validate the document against the schema using a JSON Schema validator.

A.4. Conformance Class HTML

Conformance Class	
http://www.opengis.net/spec/ogcapi-common-2/1.0/conf/html	
Target type	Web API
Requirements Class	http://www.opengis.net/spec/ogcapi_common-2/1.0/req/html
Dependency	Conformance Class "OAPI Core"

A.4.1. HTML Definition

Abstract Test 12	**/ats/html/definition**
Test Purpose	Verify support for HTML
Requirement	/req/html/definition
Test Method	Verify that every `200`-response of every operation of the API where HTML was requested is of media type `text/html`

A.4.2. HTML Content

Abstract Test 13	**/ats/html/content**
Test Purpose	Verify the content of an HTML document given an input document and schema.
Requirement	/req/html/content
Test Method	1. Validate that the document is an HTML 5 document 2. Manually inspect the document against the schema.

Annex B: Examples (Informative)

B.1. Collections Response Example

This collections example response in JSON is for a dataset with a single "buildings" feature collection.

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

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

An additional link is to a GML application schema for the collection (link relation type: "describedby").


```
{
  "links": [
    { "href": "http://data.example.org/collections.json",
      "rel": "self", "type": "application/json", "title": "this document" },
    { "href": "http://data.example.org/collections.html",
      "rel": "alternate", "type": "text/html", "title": "this document as HTML" },
    { "href": "http://schemas.example.org/1.0/foobar.xsd",
      "rel": "describedby", "type": "application/xml", "title": "XML schema for Acme
Corporation data" }
  ],
  "collections": [
    {
      "id": "buildings",
      "title": "Buildings",
      "description": "Buildings in the city of Bonn.",
      "extent": {
        "spatial": [ 7.01, 50.63, 7.22, 50.78 ],
        "temporal": [ "2010-02-15T12:34:56Z", "2018-03-18T12:11:00Z" ]
      },
      "links": [
        { "href": "http://example.org/concepts/building.html",
          "rel": "describedby", "type": "text/html",
          "title": "Feature catalogue for buildings" }
      ]
    }
  ]
}
```


B.2. Collection Object Examples

B.2.1. Building Collection

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": the title of this collection
- "description": self evident
- "attribution": markup providing attribution (owner, producer, logo, etc.) of this collection

The response includes links to:

- There is a link 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 GML application schema for the collection - using:https://www.iana.org/assignments/link-relations/link-relations.xhtml[link relation type] "describedby".

Finally, this response includes both spatial and temporal extents.

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


```
{
  "id": "1234567890",
  "title": "Example Collection Description Response",
  "description": "This is an example of a Collection Description in JSON format",
  "attribution": "<a href='www.ign.es' rel=' '>IGN</a> <a href='www.govdata.de/dl-
de/by-2-0'>(c)</a>",
  "links": [
    { "href": "http://data.example.org/collections.json",
      "rel": "self", "type": "application/json", "title": "this document" },
    { "href": "http://data.example.org/collections.html",
      "rel": "alternate", "type": "text/html", "title": "this document as HTML" },
    { "href": "http://schemas.example.org/1.0/foobar.xsd",
      "rel": "describedby", "type": "application/xml", "title": "XML schema for Acme
Corporation data" }
  ],
  "extent": {
    "spatial": [ 7.01, 50.63, 7.22, 50.78 ],
    "temporal": [ "2010-02-15T12:34:56Z", "2018-03-18T12:11:00Z" ]
    }
}
```

Annex C: Revision History

Date	Release	Editor	Primary clauses modified	Description
2019-11-25	1.0.0-SNAPSHOT	Panagiotis (Peter) Vretanos, Clemens Portele	all	initial version
2020-04-21	1.0.1-SNAPSHOT	Chuck Heazel	all	Initial API-Common version

Annex D: Glossary

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

NOTE:	When no ambiguity is possible, the word `test` may be omitted. i.e. conformance test module is the same as conformance module. Conformance modules may be nested in a hierarchical way. 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:	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:	"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:	The standardization target is the entity which may receive a certificate of conformance for a requirements class.

Annex E: Bibliography

- Fielding, Roy Thomas. **Architectural Styles and the Design of Network-based Software Architectures**. Doctoral dissertation, University of California, Irvine, 2000.
- **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.
- Internet Assigned Numbers Authority (IANA). **Link Relation Types** [online, viewed 2020-03-16], Available at https://www.iana.org/assignments/link-relations/link-relations.xml.
- Open Geospatial Consortium: **The Specification Model — A Standard for Modular specifications**, OGC 08-131
- Open Geospatial Consortium (OGC) / World Wide Web Consortium (W3C): **Spatial Data on the Web Best Practices** [online]. Edited by J. Tandy, L. van den Brink, P. Barnaghi. 2017 [viewed 2020-03-16]. Available at https://www.w3.org/TR/sdw-bp/.
- World Wide Web Consortium (W3C): **Data on the Web Best Practices** [online]. Edited by B.F. Lóscio, C. Burle, N. Calegari. 2017 [viewed 2020-03-16]. Available at https://www.w3.org/TR/dwbp/.
- World Wide Web Consortium (W3C): **Data Catalog Vocabulary (DCAT) - Version 3** [online]. Edited by R. Albertoni, D. Browning, S. Cox, A.G. Beltran, A. Perego, P. Winstanley. W3C Editors Draft 15 June 2021. Available at https://w3c.github.io/dxwg/dcat/.
- Open Geospatial Consortium (OGC): **Welcome To The OGC APIs** [online, viewed 2020-03-16]. Available at http://www.ogcapi.org/.
- Open Geospatial Consortium (OGC): **OGC Link Relations Registry**, [online, viewed 2020-04-17]. Available at https://github.com/opengeospatial/NamingAuthority/blob/master/incubation/linkRelationTypes/linkrelations.csv.
- Open Geospatial Consortium (OGC): **Compliance Testing Program Policies & Procedures** [online, viewed 2020-04-18]. Available at https://portal.ogc.org/files/?artifact_id=28982.