OGC (add title text)

Table of Contents

Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: 2019-05-03

External identifier of this OGC® document: http://www.opengis.net/doc/{doc-type}/{standard}/{m.n}

Internal reference number of this OGC® document: YY-nnnrx

Version: 0.0.2

Category: OGC® Implementation Specification

Editor: Charles Heazel

OGC API Common

Copyright notice

Warning

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

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

Document type: OGC®ImplementationSpecification
Document subtype: if applicable
Document stage: Draft
Document language: English

License Agreement

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

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

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

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

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

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

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

Chapter 1. Introduction

i. Abstract

The OGC has extended their suite of standards to include Resource Oriented Architectures and Web APIs. In the course of developing these standards, some practices proved to be common accross all OGC API standards. The purpose of this standard is to document those practices. It also serves as a common foundation upon which all OGC APIs will be built. As such, this OGC API Common standard serves as the "OWS Common" standard for OGC Resource Oriented APIs.

Consistent with the architecture of the Web, this specification uses a resource architecture and specifies a RESTful service interface consistent with the IETF HTTP/HTTPS RFCs.

This standard defines the resources listed in Table 1. These resources are common for all OGC APIs. For an overview of these resources, see section 7.1 Overview.

Table 1. Overview of Resources

Resource	Path	HTTP Method	Document Reference
Landing page	`/`	GET	7.2 API landing page
API definition	`/api`	GET	7.3 API definition
Conformance classes	`/conformance`	GET	7.4 Declaration of conformance classes
Collections metadata	`/collections`	GET	7.11 collections metadata

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

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

ii. Keywords

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

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

iii. Preface

OGC Declaration

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

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

ISO Declaration

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

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

iv. Submitting organizations

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

- Heazeltech LLC
- others TBD

v. Submitters

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

Name	**Affiliation**
Chuck Heazel *(editor)*	Heazeltech
others	TBD

Chapter 2. Scope

This specification identifies resources, captures compliance classes, and specifies requirements which are applicable to all OGC API standards. It should be included as a normative reference by all such standards.

This specification addresses two fundamental operations; discovery and query.

Discovery operations allow the API to be interrogated to determine its capabilities and retrieve information (metadata) about this distribution of a resource. This includes the API definition of the server as well as metadata about the resources provided by the server.

Query operations allow resources to be retrieved from the underlying data store based upon simple selection criteria, defined by the client.

Chapter 3. Conformance

This standard defines four requirements / conformance classes.

The standardization target of all conformance classes is "Web APIs".

The main requirements class is:

- Core.

The *Core* specifies requirements that all OGC APIs must implement.

The *Core* does not mandate a specific encoding or format for representing resources. Two requirements classes depend on the *Core* and specify representations for these resources in commonly used encodings for spatial data on the web:

- HTML,
- GeoJSON

None of these encodings are mandatory and an implementation of the *Core* may also decide to implement none of them, but to implement another encoding instead.

That said, the *Core* requirements class includes recommendations to support where practical HTML and GeoJSON as encodings. Clause 6 (Overview) includes a discussion about the recommended encodings.

The *Core* does not mandate any encoding or format for the formal definition of the API either. The prefered option is the OpenAPI 3.0 specification. A requirements class has been specified for OpenAPI 3.0, which depends on the *Core*:

- OpenAPI specification 3.0.

An implementation of the *Core* requirements class may also decide to use other API definition representations in addition or instead of an OpenAPI 3.0 definition. Examples for alternative API definitions: OpenAPI 2.0 (Swagger), future versions of the OpenAPI specification, an OWS Common 2.0 capabilities document or WSDL.

The *Core* is intended to be the minimal useful service interface for fine-grained access to a spatial resource.

Additional capabilities such as support for transactions, complex data structures, rich queries, other coordinate reference systems, subscription/notification, returning aggregated results, etc., may be specified in future parts of OGC API Common, other OGC API standards, or as vendor-specific extensions.

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

Chapter 4. References

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

- Open API Initiative: **OpenAPI Specification 3.0.1**, https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.1.md
- Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: **IETF RFC 2616, HTTP/1.1**, http://tools.ietf.org/rfc/rfc2616.txt
- 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
- Nottingham, M.: **IETF RFC 8288, Web Linking**, http://tools.ietf.org/rfc/rfc8288.txt
- Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S., Schaub, T.: **IETF RFC 7946, The GeoJSON Format**, https://tools.ietf.org/rfc/rfc7946.txt
- W3C: **HTML5, W3C Recommendation**, http://www.w3.org/TR/html5/
- **Schema.org**: http://schema.org/docs/schemas.html

Chapter 5. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r8], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

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

5.1. dataset

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

5.2. distribution

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

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

Chapter 6. Conventions

6.1. Identifiers

The normative provisions in this draft standard are denoted by the URI `http://www.opengis.net/spec/OAPI_Common/1.0`.

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

6.2. UML model

UML diagrams are included in this standard to illustrate the conceptual model that underpins OGC API implementations. The UML model is not normative. The UML profile used is specified in ISO 19103:2015.

Resources are modelled as UML interfaces.

6.3. Link relations

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

6.4. Use of HTTPS

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

6.5. API definition

6.5.1. General remarks

Good documentation is essential for every API so that developers can more easily learn how to use the API. In the best case, documentation will be available in HTML and in a format that can be processed by software to connect to the API.

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

6.5.2. Role of OpenAPI

This document uses OpenAPI 3.0 fragments as examples and to formally state requirements. Using OpenAPI 3.0 is not required for implementing a 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 requirements class advertised through the /conformance path.

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

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

6.5.3. References to OpenAPI components in normative statements

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

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

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

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

6.5.4. Paths in OpenAPI definitions

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

- An API may be hosted (replicated) on more than one server.
- Differrent parts of an API may be hosted on different servers.

Example 1. URL of the OpenAPI definition

If the OpenAPI Server Object looks like this:

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

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

6.5.5. Reusable OpenAPI components

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

CAUTION	During the development phase, these components use a base URL of "https://raw.githubusercontent.com/opengeospatial/OAPI_Common/master/", but eventually they are expected to be available under the base URL "http://schemas.opengis.net/wfs/3.0/openapi/".

Unresolved directive in OAPI_Common.adoc - include::clause_6_informative_text.adoc[]

Chapter 7. Requirement Class "Core"

7.1. Overview

Requirements Class	
http://www.opengis.net/spec/OAPI_Common/1.0/req/core	
Target type	Web API
Dependency	RFC 2616 (HTTP/1.1)
Dependency	RFC 2818 (HTTP over TLS)
Dependency	RFC 3339 (Date and Time on the Internet: Timestamps)
Dependency	RFC 5988 (Web Linking)

Figure 1 illustrates the resources supported by the *Core* requirements class using UML. Each resource type available through the API is an «interface».

Servers that implement an OGC API provide access to a collection of resources. In other words, the API is a distribution of that collection. A file download, for example, would be another distribution.

More specifically, each OGC API has a single `LandingPage` (path `/`).

```
NOTE: All paths (e.g., `/`) are relative to the base URL of the distribution of the
dataset. If the API covers other resources beyond those specified in this document,
the landing page may be a sub-resource of the base URL of the API.
```

The landing page provides links to:

- the `APIDefinition` (path `/api`),
- the `Conformance` statements (path `/conformance`),
- metadata about the resource `Collections` (path `/collections`).

The `APIDefinition` describes the capabilities of the API. Clients use that information to connect to the API. Development tools can use this information to support the implementation of the API by servers and clients. Accessing the `APIDefinition` using HTTP GET returns a description of the API.

Accessing `Conformance` using HTTP GET returns a list of URIs for the requirements classes implemented by the API.

The distribution consists of a set of resource `Collections`. This standard does not include any requirements about how the resources have to be aggregated into collections.

Accessing the `Collections` using HTTP GET returns a response that consists of `CollectionMetadata` about each `Collection` and a link to the `Collection` itself. This metadata includes:

- A local identifier for the collection that is unique for the API;
- A list of coordinate reference systems (CRS) in which geometries may be returned by the API.

The first CRS is the default coordinate reference system (in the *Core*, the default is always WGS 84 with axis order longitude/latitude);

- An optional title and description for the collection;
- An optional extent that can be used to provide an indication of the spatial and temporal extent of the collection.

CollectionMetadata about an individual collection can be retrieved from path `/collections/{collectionId}` where {collectionId} is the local identifier for that collection.

Accessing a `Collection` (path `/collections/{collectionId}/items`) using HTTP GET returns a `CollectionResponse`. This response consists of resouces from the collection. The resources included in the response are determined by the server based on parameters of the request.

A `bbox` or `datetime` parameter may be used to select only a subset of the resources in the collection (the resources that are located in the bounding box or time period. The `bbox` and `datetime` parameter also match all resources in the collection that are not associated with a location, time stamp, or time interval

Each `Resource` (path `/collections/{collectionId}/items/{resourceId}`) is also a separate resource and may be requested individually using HTTP GET.

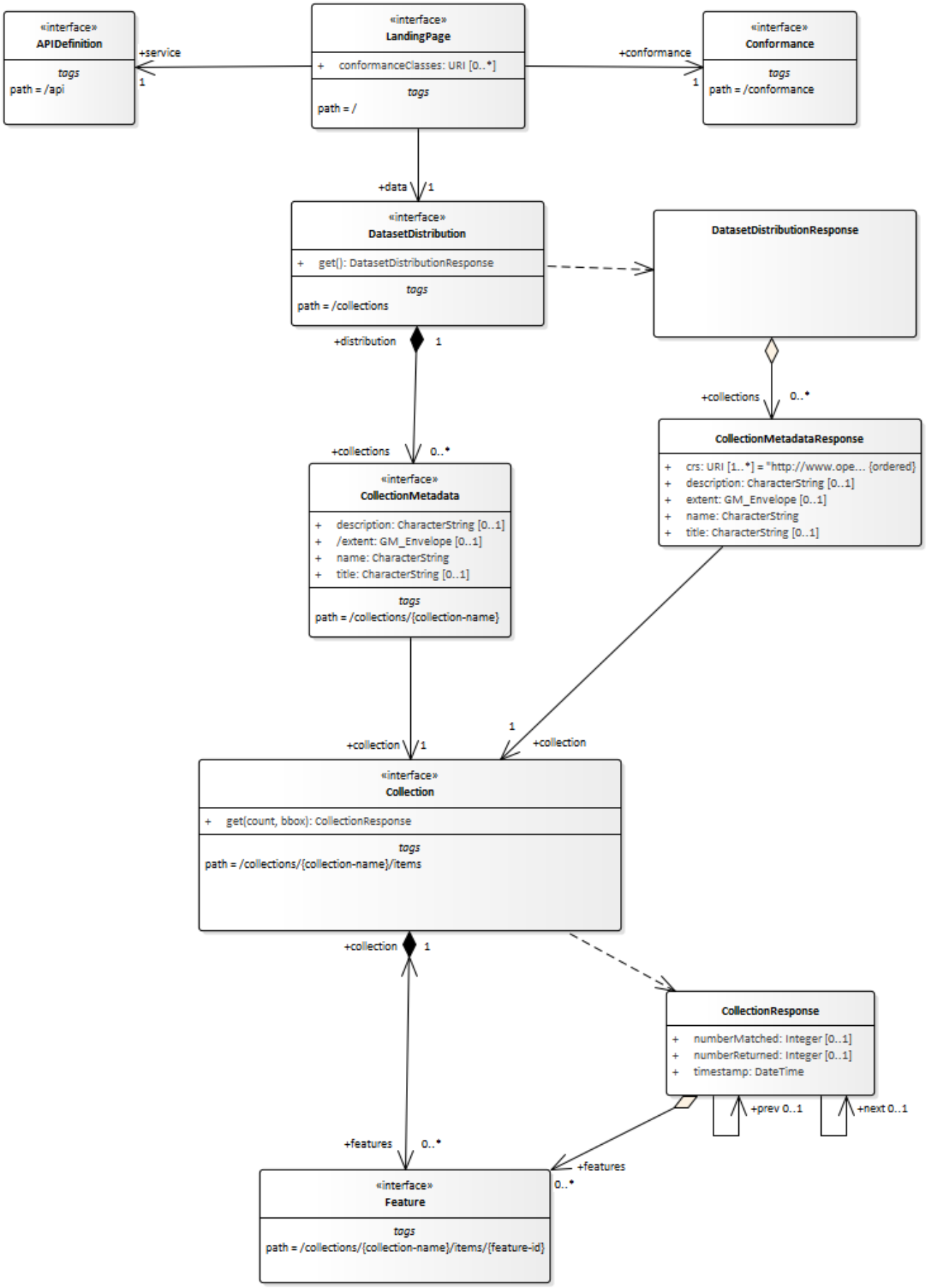

Figure 1. Resources in the Core requirements class

7.2. API landing page

7.2.1. Operation

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

7.2.2. Response

Requirement 2	/req/core/root-success
A	A successful execution of the operation SHALL be reported as a response with an HTTP status code `200`.
B	The content of that response SHALL be based upon the OpenAPI 3.0 schema root.yaml and include links to the following resources: • `/api` (relation type 'service') • `/conformance` (relation type 'conformance') • `/collections` (relation type 'data')

Schema for the landing page

```
type: object
required:
  - links
properties:
  links:
    type: array
    items:
      $ref:
https://raw.githubusercontent.com/opengeospatial/OAPI_Common/master/core/openapi/schem
as/link.yaml
```

Example 2. Landing page response document

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

7.2.3. Error Situations

See HTTP status codes for general guidance.

7.3. API definition

7.3.1. Operation

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

Requirement 3	/req/core/api-definition-op
A	The API SHALL support the HTTP GET operation at the path `/api`.

7.3.2. Response

Requirement 4	/req/core/api-definition-success
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The API SHALL return an API Definition document.

Recommendation 1	/rec/core/api-definition-oas
A	If the API definition document uses the OpenAPI Specification 3.0, the document SHOULD conform to the OpenAPI Specification 3.0 requirements class.

If multiple API definition formats are supported, use content negotiation to select the desired representation.

The idea is that any OGC API can be used by developers that are familiar with the API definition language(s) supported by the API. For example, if an OpenAPI definition is used, it should be possible to create a working client using the OpenAPI definition. The developer may need to learn a little bit about geometry data types, etc., but they should not be required to read the standard to access the data via the API.

7.3.3. Error Situations

See HTTP status codes for general guidance.

7.4. Declaration of Conformance Classes

7.4.1. Operation

To support "generic" clients that want to accessing OGC APIs in general - and not "just" a specific API / server, the API has to declare the requirements classes it implements and conforms to.

Requirement 5	/req/core/conformance-op
A	The API SHALL support the HTTP GET operation at the path `/conformance`.

7.4.2. Response

Requirement 6	/req/core/conformance-success
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The content of that response SHALL be based upon the OpenAPI 3.0 schema req-classes.yaml and list all requirements classes that the API conforms to.

Schema for the list of requirements classes

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

Example 3. Requirements class response document

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

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

7.4.3. Error situations

See HTTP status codes for general guidance.

7.5. HTTP 1.1

Requirement 7	/req/core/http
A	The API SHALL conform to HTTP 1.1.
B	If the API supports HTTPS, then the API SHALL also conform to HTTP over TLS.

This includes the correct use of status codes, headers, etc.

7.5.1. HTTP status codes

Table 2 lists the main HTTP status codes that clients should be prepared to receive.

This includes, for example, support for specific security schemes or URI redirection.

In addition, other error situations may occur in the transport layer outside of the server.

Table 2. Typical HTTP status codes

Status code	Description
200	A successful request.
304	An entity tag was provided in the request and the resource has not been changed since the previous request.
400	The server cannot or will not process the request due to an apparent client error. For example, a query parameter had an incorrect value.
401	The request requires user authentication. The response includes a `WWW-Authenticate` header field containing a challenge applicable to the requested resource.
403	The server understood the request, but is refusing to fulfill it. While status code `401` indicates missing or bad authentication, status code `403` indicates that authentication is not the issue, but the client is not authorised to perform the requested operation on the resource.
404	The requested resource does not exist on the server. For example, a path parameter had an incorrect value.
405	The request method is not supported. For example, a POST request was submitted, but the resource only supports GET requests.
406	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.

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

Permission 1	/per/core/additional-status-codes
A	Servers MAY support other capabilities of the HTTP protocol and, therefore, MAY return other status codes than those listed in Table 2, too.

7.6. Web caching

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

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

7.7. Support for cross-origin requests

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

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

Two common mechanisms to support cross-origin requests are:

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

7.8. Encodings

While the OAPI Common standard does not specify any mandatory encoding, the following encodings are recommended. See Clause 6 (Overview) for a discussion.

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

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

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

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

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

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

NOTE

Two common approaches are:

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

7.9. Coordinate reference systems

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

For the reasons discussed in the Best Practices, OGC APIs use WGS84 longitude and latitude as the default coordinate reference system.

Requirement 8	/req/core/crs84
A	Unless the client explicitly requests a different coordinate reference system, all spatial geometries SHALL be in the coordinate reference system http://www.opengis.net/def/crs/OGC/1.3/CRS84 (WGS84 longitude/latitude).

The implementations compliant with the Core are not required to support publishing geometries in coordinate reference systems other than http://www.opengis.net/def/crs/OGC/1.3/CRS84. The Core also does not specify a capability to request geometries in a different reference system than the native one of the published resource. Such a capability will be specified in other OGC API standards.

7.10. Link headers

Recommendation 6	/rec/core/link-header
A	Links included in payload of responses SHOULD also be included as `Link` headers in the HTTP response according to RFC 8288, Clause 3.

B	This recommendation does not apply, if there are a large number of links included in a response or a link is not known when the HTTP headers of the response are created.

7.11. Collections metadata

7.11.1. Operation

Requirement 9	/req/core/rc-md-op
A	The API SHALL support the HTTP GET operation at the path `/collections`.

7.11.2. Response

Requirement 10	/req/core/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 OpenAPI 3.0 schema content.yaml.

Schema for the metadata about resource collections

```
type: object
required:
  - links
  - collections
properties:
  links:
    type: array
    items:
      $ref:
https://raw.githubusercontent.com/opengeospatial/OAPI_Common/master/core/openapi/schem
as/link.yaml
  collections:
    type: array
    items:
      $ref:
https://raw.githubusercontent.com/opengeospatial/OAPI_Common/master/core/openapi/schem
as/collectionInfo.yaml
```

Requirement 11	/req/core/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.

Recomendation 7	/rec/core/rc-md-desciptions
A	If external schemas or descriptions for the dataset exist that provide information about the structure or semantics of the data, 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 to the whole dataset.
C	For resources that describe the contents of a resource collection, the links SHOULD be set in the `links` property of the appropriate object in the `collections` resource.

Examples for descriptions are: XML Schema, Schematron, JSON Schema, RDF Schema, OWL, SHACL, a feature catalogue, etc.

Requirement 12	/req/core/rc-md-items
A	For each resource collection in this distribution of the dataset, an item SHALL be provided in the property `collections`.

Permission 2	/per/core/rc-md-items
A	To support servers with many collections, servers MAY limit the number of items in the in the property `collections`.

This document does not specify mechanisms how clients may access all collections from APIs with many collections. Such mechanisms may be specified in additional parts of OGC API Common or in the resource-specific API standards.

Requirement 13	/req/core/rc-md-items-links
A	For each resource collection in this distribution of the dataset, the `links` property of the collection SHALL include an item for each supported encoding with a link to the collection resource (relation: `items`).
B	All links SHALL include the `rel` and `type` properties.

Requirement 14	/req/core/rc-md-extent
A	For each resource collection, the `extent` property, if provided, SHALL provide bounding boxes that include all spatial geometries and time intervals that include all temporal geometries in this collection. The temporal extent may use `null` values to indicate an open time interval.
B	If a 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 8	/rec/core/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 3	/per/core/rc-md-extent-extensions
A	The Core only specifies requirements for spatial and temporal extents. However, the `extent` object MAY be extended with additional members to represent other extents, for example, thermal or pressure ranges.
B	The Core only supports spatial extents in WGS84 longitude/latitude andtemporal extents in the calendar (these are the only enum values in extent.yaml). Extension to the Core MAY add additional reference systems to the `extent` object.

Schema for the metadata about a feature collection

```
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 resources in the collection
    type: string
  links:
    type: array
    items:
      $ref:
https://raw.githubusercontent.com/opengeospatial/OAPI_Common/master/core/openapi/schem
as/link.yaml
  extent:
    $ref:
https://raw.githubusercontent.com/opengeospatial/OAPI_Common/master/core/openapi/schem
as/extent.yaml
  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 4. Collection metadata response document

This feature collection metadata example response in JSON is for a dataset with a single collection "buildings". It includes links to the collection resource in all formats that are supported by the API (link relation type: "items").

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

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

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

"license").

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


```
{
  "links": [
    { "href": "http://data.example.org/collections.json",
      "rel": "self", "type": "application/json", "title": "this document" },
    { "href": "http://data.example.org/collections.html",
      "rel": "alternate", "type": "text/html", "title": "this document as HTML" },
    { "href": "http://schemas.example.org/1.0/foobar.xsd",
      "rel": "describedBy", "type": "application/xml", "title": "XML schema for
Acme Corporation data" }
  ],
  "collections": [
    {
      "id": "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://data.example.org/collections/buildings/items",
          "rel": "items", "type": "application/geo+json",
          "title": "Buildings" },
        { "href": "http://example.org/concepts/building.html",
          "rel": "describedBy", "type": "text/html",
          "title": "Feature catalogue for buildings" }
      ]
    }
  ]
}
```


7.11.3. Error situations

See HTTP status codes for general guidance.

7.12. Resource Collection metadata

7.12.1. Operation

Requirement 15	/req/core/src-md-op

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

7.12.2. Response

Requirement 16	/req/core/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 the same as the content for this resource collection in the `/collections` response.

7.12.3. 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 2).

7.13. Resource Collections

7.13.1. Operation

Requirement 17	/req/core/rc-op
A	For every resource collection identified in the metadata about the resource collection (path `/`), the API SHALL support the HTTP GET operation at the path `/collections/{collectionId}/items`.
B	The parameter `collectionId` is each `id` property in the resource collections metadata (JSONPath: `$.collections[*].id`).

7.14. Parameters

7.14.1. Parameter bbox

Requirement 18	/req/core/rc-bbox-definition

A	Each resource collection operation SHALL support a parameter `bbox`with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: bbox`
	`in: query`
	`required: false`
	`schema:`
	`  type: array`
	`  minItems: 4`
	`  maxItems: 6`
	`  items:`
	`  type: number`
	`style: form`
	`explode: false`

Requirement 19	/req/core/rc-bbox-response
A	If the `bbox` parameter is provided, only those resources that have a spatial geometry that intersects the bounding box SHALL be part of the result set.
B	The bounding box SHALL consist of 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 • Lower left corner, coordinate axis 3 (optional) • Upper right corner, coordinate axis 1 • Upper right corner, coordinate axis 2 • Upper right corner, coordinate axis 3 (optional)
C	The coordinate reference system of the values 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`.

"Intersects" means that the rectangular area specified in the parameter `bbox` includes a coordinate that is part of the (spatial) geometry of the resource. This includes the boundaries of the geometries (e.g. for curves the start and end position and for surfaces the outer and inner rings).

This standard does not specify requirements for the parameter `bbox-crs`. Those requirements will

be specified in a latter version of this specification.

For WGS84 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 the first value (west-most box edge) is larger than the third value (east-most box edge).

Example 5. 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`.

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

7.14.2. Parameter datetime

Requirement 20	/req/core/rc-time-definition
A	Each resource collection operation SHALL support a `datetime` parameter.

B	This `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`

Requirement 21	/req/core/rc-time-response
A	If the `datetime` parameter is provided, only resources that have a temporal geometry that intersects the temporal information in the `datetime` parameter SHALL be part of the result set, if the parameter is provided.

B	The temporal information is either a date-time or an 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`

C	The syntax of `date-time` is specified by RFC 3339, 5.6.
D	Open ranges in time intervals at the start or end SHALL be supported using a double-dot (`..`).
E	If a resourcee has multiple temporal properties, it is the decision of the API whether only a single temporal property is used to determine the extent or all relevant temporal properties.

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

Example 6. A date-time

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

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

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

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

Example 7. 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.

Chapter 8. Requirements classes for encodings

8.1. Overview

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

- HTML
- GeoJSON

None of these encodings are mandatory and an implementation of the Core requirements class may also implement none of them but implement another encoding instead.

The Core requirements class includes recommendations to support HTML and GeoJSON as encodings, where practical. Clause 6 (Overview) includes a discussion about recommended encodings.

8.2. Requirement Class "HTML"

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

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

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

This is discussed in detail in Best Practice 2: Make your spatial data indexable by search engines [SDWBP]. This standard therefore recommends supporting HTML as an encoding.

Requirements Class	
http://www.opengis.net/spec/OAPI_Common/1.0/req/html	
Target type	Web API
Dependency	OAPI Core
Dependency	HTML5
Dependency	Schema.org

Requirement 22	/req/html/definition

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

Requirement 23	/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 9	/rec/html/schema-org
A	A `200`-response with the media type `text/html`, SHOULD include Schema.org annotations.

8.3. Requirement Class "GeoJSON"

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

Requirements Class	
http://www.opengis.net/spec/OAPI_Common/1.1/req/geojson	
Target type	Web API
Dependency	OAPI Core
Dependency	GeoJSON

Requirement 24	/req/geojson/definition
A	`200`-responses of the server SHALL support the following media types: • `application/geo+json` for feature collections and features, and • `application/json` for all other resources.

Requirement 25	/req/geojson/content

A	Every `200`-response with the media type `application/geo+json` SHALL be • a GeoJSON FeatureCollection Object for feature collections, and • a GeoJSON Feature Object for features.
B	The links specified in the requirements `/req/core/fc-links` and `/req/core/f-links` SHALL be added in a extension property (foreign member) with the name `links`.
C	The schema of all responses with the media type `application/json` SHALL conform with the JSON Schema specified for that resource.

Templates for the definition of the schemas for the GeoJSON responses in OpenAPI definitions are available at featureCollectionGeoJSON.yaml and featureGeoJSON.yaml. These are generic schemas that do not include any application schema information about specific resource types or their properties.

Example 8. A GeoJSON FeatureCollection Object response

In the example below, only the first and tenth feature is shown. Coordinates are not shown.

```
{
  "type" : "FeatureCollection",
  "links" : [ {
    "href" : "http://data.example.com/collections/buildings/items/?f=json",
    "rel" : "self",
    "type" : "application/geo+json",
    "title" : "this document"
  }, {
    "href" : "http://data.example.com/collections/buildings/items/?f=html",
    "rel" : "alternate",
    "type" : "text/html",
    "title" : "this document as HTML"
  }, {
    "href" :
"http://data.example.com/collections/buildings/items/?f=json&startIndex=10&limit=1
0",
    "rel" : "next",
    "type" : "application/geo+json",
    "title" : "next page"
  } ],
  "timeStamp" : "2018-04-03T14:52:23Z",
  "numberMatched" : 123,
  "numberReturned" : 10,
  "features" : [ {
    "type" : "Feature",
    "id" : "123",
    "geometry" : {
      "type" : "Polygon",
      "coordinates" : [ ... ]
    },
    "properties" : {
      "function" : "residential",
      "floors" : "2",
      "lastUpdate" : "2015-08-01T12:34:56Z"
    }
  }, { ...
  }, {
    "type" : "Feature",
    "id" : "132",
    "geometry" : {
      "type" : "Polygon",
      "coordinates" : [ ... ]
    },
    "properties" : {
      "function" : "public use",
      "floors" : "10",
      "lastUpdate" : "2013-12-03T10:15:37Z"
    }
  } ]
}
```

Example 9. A GeoJSON Feature Object response

In the example below, coordinates are not shown.


```
{
  "type" : "Feature",
  "links" : [ {
    "href" : "http://data.example.com/collections/buildings/items/123/?f=json",
    "rel" : "self",
    "type" : "application/geo+json",
    "title" : "this document"
  }, {
    "href" : "http://data.example.com/collections/buildings/items/123/?f=html",
    "rel" : "alternate",
    "type" : "text/html",
    "title" : "this document as HTML"
  }, {
    "href" : "http://data.example.com/collections/buildings/items",
    "rel" : "collection",
    "type" : "application/geo+json",
    "title" : "the collection document"
  } ],
  "id" : "123",
  "geometry" : {
    "type" : "Polygon",
    "coordinates" : [ ... ]
  },
  "properties" : {
    "function" : "residential",
    "floors" : "2",
    "lastUpdate" : "2015-08-01T12:34:56Z"
  }
}
```

Chapter 9. Requirements class "OpenAPI 3.0"

9.1. Basic requirements

APIs conforming to this requirements class document themselves by an OpenAPI Document.

Requirements Class	
http://www.opengis.net/spec/OAPI_Common/1.0/req/oas30	
Target type	Web API
Dependency	OAPI Core
Dependency	OpenAPI Specification 3.0.1

Requirement 26	/req/oas30/oas-definition-1
A	The API SHALL provide an OpenAPI definition in JSON and HTML at the path `/api` using the media type `application/openapi+json;version=3.0`.

CAUTION

ISSUE 117

The OpenAPI media type has not been registered yet with IANA and will likely change. We need to update the media type after registration.

Requirement 27	/req/oas30/oas-definition-2
A	The JSON representation SHALL conform to the OpenAPI Specification, version 3.0.

Two example OpenAPI documents are included in Annex B.

Requirement 28	/req/oas30/oas-impl
A	The API SHALL implement all capabilities specified in the OpenAPI definition.

9.2. Complete definition

Requirement 29	/req/oas30/completeness

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

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

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

9.3. Exceptions

Requirement 30	/req/oas30/exceptions-codes
A	For error situations that originate from an API server, the API definition SHALL cover all applicable HTTP Status Codes.

Example 10. An exception response object definition

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

9.4. Security

Requirement 31	/req/oas30/security
A	For cases, where the operations of the API are access-controlled, the security scheme(s) and requirements SHALL be documented in the OpenAPI definition.

The OpenAPI specification currently supports the following security schemes:

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

Chapter 10. Media Types

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

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

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

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

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

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

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

Unresolved directive in OAPI_Common.adoc - include::annex-ats.adoc[]

Unresolved directive in OAPI_Common.adoc - include::annex-history.adoc[]

Unresolved directive in OAPI_Common.adoc - include::annex-bibliography.adoc[]