OGC API - Features - Part 5 / OGC API - Common - Part 3

Schemas

Open Geospatial Consortium

Submission Date: 2025-09-23

Approval Date: 2026-04-30

Publication Date: 2026-04-30

External identifier of this OGC® document: http://www.opengis.net/doc/IS/ogcapi-common-3/1.0

Internal reference number of this OGC® document: 23-058r2

Version: 1.0.0

Latest Published Draft: n/a

Category: OGC® Implementation Standard

Editors: Clemens Portele, Panagiotis (Peter) A. Vretanos

OGC API - Features - Part 5 / OGC API - Common - Part 3: Schemas

Copyright notice

Copyright © 2026 Open Geospatial Consortium

To obtain additional rights of use, visit https://www.ogc.org/legal/

Warning

This document is an OGC Member approved international standard. This document is available on a royalty free, non-discriminatory basis. 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® Standard
Document subtype:	Implementation
Document stage:	Approved
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

1. Scope . . . 9
2. Conformance . . . 10
3. References . . . 12
4. Terms, Definitions, Symbols and Abbreviated Terms . . . 13
   4.1. Terms and Definitions . . . 13
   4.2. Abbreviated terms . . . 15
5. Conventions . . . 16
   5.1. General remarks . . . 16
   5.2. Identifiers . . . 16
   5.3. Additional link relation types . . . 16
   5.4. HTTP URIs . . . 16
6. Overview . . . 17
   6.1. Requirements and recommendations for schemas . . . 17
   6.2. Requirements and recommendations for schemas in OGC Web APIs . . . 18
   6.3. Requirements and recommendations for profiles . . . 18
7. Requirements Class "Schemas" . . . 20
   7.1. Schema representation . . . 20
   7.2. Additional keywords . . . 23
      7.2.1. General rules . . . 23
      7.2.2. Properties with a specific role . . . 23
      7.2.3. Ordering properties . . . 23
      7.2.4. The unit of measure for a numeric property . . . 24
      7.2.5. Semantic definition of a property . . . 25
      7.2.6. Special null values . . . 25
   7.3. Examples . . . 25
8. Requirements Class "Advanced property roles" . . . 29
   8.1. Primary geometry . . . 29
   8.2. Primary temporal information . . . 30
   8.3. Type . . . 31
   8.4. Examples . . . 31
9. Requirements Class "References" . . . 33
10. Requirements Class "Returnables and Receivables" . . . 36
11. Requirements Class "Queryables" . . . 37
12. Requirements Class "Sortables" . . . 40
13. Requirements Class "Profile query parameter" . . . 42
14. Requirements Class "Profiles for references" . . . 44
15. Requirements Class "Profiles for codelists" . . . 48
16. Requirements Class "Profiles for value domains" . . . 51

17. Media Types . . . 56
18. Security Considerations . . . 57
Annex A: Abstract Test Suite (Normative) . . . 58
A.1. Conformance Class "Schemas" . . . 58
A.1.1. Conformance Test 1 . . . 58
A.1.2. Conformance Test 2 . . . 60
A.1.3. Conformance Test 3 . . . 60
A.1.4. Conformance Test 4 . . . 60
A.1.5. Conformance Test 5 . . . 61
A.1.6. Conformance Test 6 . . . 61
A.1.7. Conformance Test 7 . . . 62
A.2. Conformance Class "Advanced property roles" . . . 62
A.2.1. Conformance Test 8 . . . 62
A.2.2. Conformance Test 9 . . . 63
A.2.3. Conformance Test 10 . . . 64
A.3. Conformance Class "References" . . . 64
A.3.1. Conformance Test 11 . . . 64
A.4. Conformance Class "Returnables and Receivables" . . . 65
A.4.1. Conformance Test 12 . . . 65
A.4.2. Conformance Test 13 . . . 66
A.4.3. Conformance Test 14 . . . 67
A.4.4. Conformance Test 15 . . . 68
A.4.5. Conformance Test 16 . . . 69
A.5. Conformance Class "Queryables" . . . 70
A.5.1. Conformance Test 17 . . . 70
A.5.2. Conformance Test 18 . . . 71
A.5.3. Conformance Test 19 . . . 72
A.6. Conformance Class "Sortables" . . . 74
A.6.1. Conformance Test 20 . . . 74
A.6.2. Conformance Test 21 . . . 74
A.6.3. Conformance Test 22 . . . 75
A.7. Conformance Class "Profile query parameter" . . . 76
A.7.1. Conformance Test 23 . . . 76
A.7.2. Conformance Test 24 . . . 77
A.7.3. Conformance Test 25 . . . 78
A.8. Conformance Class "Profiles for references" . . . 79
A.8.1. Conformance Test 26 . . . 80
A.8.2. Conformance Test 27 . . . 80
A.8.3. Conformance Test 28 . . . 81
A.8.4. Conformance Test 29 . . . 82
A.8.5. Conformance Test 30 . . . 83

A.9. Conformance Class "Profiles for codelists" . . . 84
A.9.1. Conformance Test 31 . . . 84
A.9.2. Conformance Test 32 . . . 85
A.10. Conformance Class "Profiles for value domains" . . . 87
A.10.1. Conformance Test 33 . . . 87
A.10.2. Conformance Test 34 . . . 87
A.10.3. Conformance Test 35 . . . 90
A.10.4. Conformance Test 36 . . . 91
Annex B: Revision History . . . 92
Annex C: Bibliography . . . 93

i. Abstract

OGC API Standards define modular API building blocks to spatially enable Web API implementations in a consistent way. The OpenAPI specification is used to define the API building blocks.

OGC API - Features Standard defines building blocks to create, modify and query features on the Web. The OGC API - Features Standard is comprised of multiple parts. Each part is a separate standard.

Some API building blocks are applicable in multiple OGC API Standards. These common building blocks are documented in the OGC API — Common Standard, which is comprised of multiple parts, too.

This document ("Schemas") specifies how geospatial data such as features can be described by a logical schema and how such schemas are published in an OGC Web API implementation. The document is published both as Part 5 of OGC API - Features and as Part 3 of OGC API - Common. This document is hereafter also referred to as "this Standard".

This Standard specifies provisions for schemas that are applicable to any type of geospatial data as well as provisions for profiles. While the provision originate from use cases involving feature data, all provisions are not specific to features.

As logical schemas, the schemas describe the data independent of a particular format to represent the data.

To use a schema for data validation, the schema must be converted into a schema representation suitable for validating data in the specific data format. For example, an XML Schema that is a GML application schema or a JSON Schema for GeoJSON or JSON-FG (OGC Features and Geometries JSON).

While the schema is a logical schema, a machine-readable representation is needed. Instead of specifying a custom schema language for geospatial data, this Standard uses JSON Schema as the schema language with:

- Extensions for additional aspects that are common in geospatial data; and
- Recommendations for restricting the use of JSON Schema capabilities in order to simplify parsing a schema.

That is, JSON Schema is primarily used as a data description language.

The main reasons for using JSON Schema are:

- Web developers are often familiar with JSON Schema and JSON;
- JSON data types (string, number, boolean, array, object, null) are simple and easy to understand; and
- JSON is often used for encoding geospatial data, i.e., the schema of a property can be used to validate property values without conversion.

In OGC Web APIs, geospatial data is shared in collections (path `/collections/{collectionId}`). The schema of items in a collection provides information as to how to interact with the collection.

Examples of interactions, where information about the schema are required, include the following.

- An application that wants to fetch data often needs information about the properties included in the data to process, present, or store the data properly ("returnable" properties or "Returnables").
- An application that wants to fetch data restricted to a subset of the available properties needs the list of the properties in the data (again, "returnable" properties or "Returnables").
- An application that wants to fetch a subset of the data needs information about the properties that can be used to filter the data ("queryable" properties or "Queryables").
- An application that wants to fetch data in a specific order needs information about the properties that can be used to sort the data ("sortable" properties or "Sortables").
- An application that wants to create or update data needs information about the required or acceptable properties of the data ("receivable" properties or "Receivables").

This Standard specifies the schema resources listed in Table 1 to support these interactions.

Table 1. Overview of resources, applicable HTTP methods and links to the document sections

Resource	Path	HTTP method	Document reference
Returnables and Receivables	`/collections/{collectionId}/schema`	GET	Requirements Class "Returnables and Receivables"
Queryables	`/collections/{collectionId}/queryables`	GET	Requirements Class "Queryables"
Sortables	`/collections/{collectionId}/sortables`	GET	Requirements Class "Sortables"

Finally,

- the query parameter `profile` is specified to support requesting variations in the representation of data in the same feature format, depending on the intended use of the data;
- three profiles are specified how to encode a reference to another resource: by the `resourceId` of the referenced resource, by its URI, or as a web link;
- two profiles are specified for representing values from a codelist: as a choice of simple schemas with a constant value or as enums with a reference to the codelist resource; and
- two profiles are specified for representing value domains: either the values that are valid in general or the values that are currently used in the data.

ii. Keywords

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

OGC, schema, JSON Schema, OGC API, geospatial data, features, queryables, sortables, returnables, receivables, profiles

iii. Preface

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

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

iv. Submitting organizations

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

- Compass, Inc.
- CubeWerx Inc.
- Ecere Corporation
- Geonovum
- interactive instruments
- Met Office
- Meteorological Service of Canada
- Natural Resources Canada

v. Submitters

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

Name	**Affiliation**
Panagiotis (Peter) A. Vretanos *(editor)*	CubeWerx Inc.
Clemens Portele *(editor)*	interactive instruments
Charles Reed	Compass, Inc.
Chris Little	Met Office
Jérôme Jacovella-St-Louis	Ecere Corporation
Joost van Ulden	Natural Resources Canada
Linda van den Brink	Geonovum
Tom Kralidis	Meteorological Service of Canada

Chapter 1. Scope

This Standard specifies:

- Logical schemas describing geospatial data, for example, the properties of the features in a feature collection;
- API building blocks related to such schemas for use in OGC Web APIs;
- Profiles for representing properties that are references to other resources; and
- Profiles for representing codelists and value domains in schemas.

Chapter 2. Conformance

This Standard defines the following requirements classes, grouped by their standardization target type.

- Schema for a collection of data:
  - Requirements Class "Schemas" specifies basic provisions for schemas of items in a collection of geospatial data, for example, features, and the representation of a schema in JSON Schema;
  - Requirements Class "Advanced property roles" specifies additional property roles for geospatial data, in particular, feature data;
  - Requirements Class "References" specifies additional provisions for properties that reference another resource;
  - Requirements Class "Profiles for codelists" specifies two profiles for representing values from a codelist in a schema; and
  - Requirements Class "Profiles for value domains" specifies two profiles for representing value domains of a property in a schema.
- Collection of data:
  - Requirements Class "Profiles for references" specifies three profiles for representing properties that are references to other resources.
- Web API:
  - Requirements Class "Returnables and Receivables" specifies the web resources representing the schema with the returnable and receivable properties of a collection in OGC Web APIs;
  - Requirements Class "Queryables" specifies the web resources representing the schema with the queryable properties of a collection in OGC Web APIs;
  - Requirements Class "Sortables" specifies the web resources representing the schema with the sortable properties of a collection in OGC Web APIs; and
  - Requirements Class "Profile query parameter" specifies the "profile" query parameter to select profiles that should be used in the representation of a resource in the negotiated media type.

Conformance with this Standard shall be checked using all the relevant tests specified in Annex A of this document for each conformance class. 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.

Table 2. Conformance class URIs

Conformance class	**URI**
Schemas	`http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/schemas`
Core roles for features	`http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/advanced-property-roles`
References	`http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/references`

Conformance class	URI
Returnables and Receivables	http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/returnables-and-receivables
Queryables	http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/queryables
Sortables	http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/sortables
Profile query parameter	http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/profile-parameter
Profiles for references	http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/profile-references
Profiles for codelists	http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/profile-codelists
Profiles for value domains	http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/profile-domains

Chapter 3. 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.

- **JSON Schema: A Media Type for Describing JSON Documents**. Edited by A. Wright, H. Andrews, B. Hutton, G. Dennis. 2022 [viewed 2023-10-14]. Available at https://json-schema.org/draft/2020-12/json-schema-core
- **JSON Schema Validation: A Vocabulary for Structural Validation of JSON**. Edited by A. Wright, H. Andrews, B. Hutton. 2022 [viewed 2023-10-14]. Available at https://json-schema.org/draft/2020-12/json-schema-validation
- Open Geospatial Consortium (OGC). OGC 17-069r4: **OGC API - Features - Part 1: Core** [online]. Edited by C. Portele, P. Vretanos, C. Heazel. 2022 [viewed 2023-10-14]. Available at https://docs.ogc.org/is/17-069r4/17-069r4.html

NOTE	OGC API - Features - Part 1: Core is a normative reference for the Abstract Test Suite only. It is not a dependency for the requirements classes defined in this Standard.

- Open Geospatial Consortium (OGC). OGC 06-103r4: **OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture**. Edited by J. Herring. 2011. Available at http://portal.opengeospatial.org/files/?artifact_id=25355
- QUDT.org. **QUDT Units Vocabulary 2.1**. Edited by R. Hodgson. 2024. Available at https://www.qudt.org/doc/DOC_VOCAB-UNITS.html.
- Regenstrief Institute, Inc. **The Unified Code for Units of Measure**. Edited by G. Schadow, .C J. McDonald. 2017. Available at https://ucum.org/ucum.

Chapter 4. Terms, Definitions, Symbols and Abbreviated Terms

4.1. Terms and Definitions

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

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

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

collection

a body of resources that belong or are used together; an aggregate, set, or group of related resources (OGC API - Common - Part 2: Collections).

endpoint

a web address (URI) at which access can be gained to a service or resource

<JSON> key

the name of a **member**

<JSON> member

a name/value pair in a JSON object

logical model

an abstracted representation of domain knowledge that could feasibly be used in Model Driven Architecture to generate physical artifacts (OGC Conceptual Modeling Discussion Paper)

OGC Web API

A Web API that implements one or more Conformance Classes from an OGC API Standard (OGC API - Features - Part 1: Core)

primary geometry

the geometry that the publisher considers as the most important spatial characteristic of a **feature**

NOTE	A feature can be described by multiple spatial properties. For example, a radio tower can have a property with a point value that describes the location of the tower and another property with a multi-polygon value that describes the area of coverage. Some feature formats can represent only a single geometry per feature. In

those cases, the primary geometry will be used when the feature is encoded in such a format.

NOTE The primary geometry of a feature can also vary depending on the zoom level. At a smaller scale, the primary geometry could be a point while a polygon could be used at a larger scale.

primary temporal information

the time instant or time interval that the publisher considers as the most important temporal characteristic of a **feature**

NOTE A feature can be described by multiple temporal properties. For example, an event can have a property with an instant or interval when the event occurred or will occur and another property when the event was recorded in the dataset. The primary temporal information can also be built from two properties, e.g., when the feature has two properties describing the start and end instants of an interval.

profile

additional semantics (constraints, conventions, extensions) that are associated with a resource representation, in addition to those defined by the media type (RFC 6906: The 'profile' Link Relation Type)

publisher

entity responsible for making a resource available (Dublin Core Metadata Initiative - DCMI Metadata Terms)

NOTE As content of OGC API standards, a resource is typically published at an endpoint.

queryable

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

receivable

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

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 API standard, should be understood to mean a web resource unless otherwise indicated.

returnable

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

sortable

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

web resource

a **resource** that is identified by a HTTP URI (OGC API - Features - Part 1: Core)

4.2. Abbreviated terms

API

Application Programming Interface

HTTP

Hypertext Transfer Protocol

HTTPS

Hypertext Transfer Protocol Secure

IANA

Internet Assigned Numbers Authority

JSON

JavaScript Object Notation

OGC

Open Geospatial Consortium

URI

Uniform Resource Identifier

YAML

YAML Ain't Markup Language

Chapter 5. Conventions

5.1. General remarks

See OGC API - Features - Part 1: Core, Clauses 5 and 6.

5.2. Identifiers

The normative provisions in this Standard are denoted by the URI `http://www.opengis.net/spec/ogcapi-common-3/1.0`.

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

5.3. Additional link relation types

The following OGC link relation types are introduced in this document (no applicable link relation type in the IANA link relation type register could be identified):

- **http://www.opengis.net/def/rel/ogc/1.0/schema**: Refers to a resource that lists the properties of geospatial data in the collection represented by the link's context.
- **http://www.opengis.net/def/rel/ogc/1.0/queryables**: Refers to a resource that lists properties that can be used to filter sub-resources of the link's context.
- **http://www.opengis.net/def/rel/ogc/1.0/sortables**: Refers to a resource that lists properties that can be used to sort sub-resources of the link's context.

Note that links with the link relation type **http://www.opengis.net/def/rel/ogc/1.0/schema** reference a logical schema that is independent of the representation of the data in some format.

For referencing a JSON Schema that can be used to validate, for example, a GeoJSON document, use a link with the IETF link relation type **describedby** to reference the JSON Schema that can be used to validate the document. The use of the link relation type **describedby** for linking to a JSON Schema for schema validation is recommended in the JSON Schema specification.

NOTE	These link relation types need to be registered in the OGC Link Relation Type Register. This note has to be removed before publication.

5.4. HTTP URIs

If URIs include reserved characters that are delimiters in the URI subcomponent, these must be percent-encoded. See Clause 2 of RFC 3986 Uniform Resource Identifier (URI): Generic Syntax for details. Not all URIs in this document are properly percent-encoded for better readability.

Chapter 6. Overview

This Standard specifies three types of requirements classes.

6.1. Requirements and recommendations for schemas

The schemas describing geospatial data specified by this Standard are logical schemas describing the data, independent of a particular format to represent the data.

In order to use the schema for data validation, the schema first has to be converted into a schema representation suitable for validating data in the specific data format. For example, an XML Schema that is a GML application schema or a JSON Schema for GeoJSON or JSON-FG.

While the schema is a logical schema, a machine-readable representation is needed. Instead of specifying a custom schema language for geospatial data, this Standard uses JSON Schema as the schema language with:

- Extensions for additional aspects that are common in geospatial data, and
- Recommendations for restricting the use of JSON Schema capabilities to simplify parsing a schema.

That is, JSON Schema is primarily used as a data description language. Additional elements, listed in the following table, are defined in this Standard that can be used to describe additional aspects of the schema that are common in geospatial data.

Table 3. List of extension elements for describing aspects common in geospatial data

Additional element	Description
x-ogc-codelistUri	A reference to a codelist.
x-ogc-collectionId	Target collection of foreign key references.
x-ogc-definition	Reference to the semantic definition of a property.
x-ogc-propertySeq	An integer representing the relative position of a property in a schema in ascending order.
x-ogc-role	Declares a specific role of a property.
x-ogc-unit	Declares the unit of measure of a property.
x-ogc-unitLang	Declares the language/register used for the unit of measure.
x-ogc-uriTemplate	A URI template with a template `featureId` used to reference a resources in another local or remote collection.

The main reasons for using JSON Schema are:

- Web developers are often familiar with JSON Schema and JSON;
- JSON data types (string, number, boolean, array, object, null) are simple and easy to understand; and
- JSON is often used for encoding geospatial data, i.e., the schema of a property can be used to

validate property values without conversion.

NOTE	A proposal exists for a new specification JSON Structure that could eventually be more directly applicable for the scope of this Standard and, if approved, be used in a future revision of this Standard.

6.2. Requirements and recommendations for schemas in OGC Web APIs

These requirements classes specify schema resources in OGC Web APIs.

In OGC Web APIs, geospatial data is shared in collections (path `/collections/{collectionId}`). The schema of items in a collection provides information about how to interact with the collection.

Examples of interactions, where information about the schema are required, include the following.

- An application that wants to fetch data often needs information about the properties included in the data to process, present, or store the data properly ("returnable" properties or "Returnables").
- An application that wants to fetch data restricted to a subset of the available properties needs the list of the properties in the data (again, "returnable" properties or "Returnables").
- An application that wants to fetch a subset of the data needs information about the properties that can be used to filter the data ("queryable" properties or "Queryables").
- An application that wants to fetch data in a specific order needs information about the properties that can be used to sort the data ("sortable" properties or "Sortables").
- An application that wants to create or update data needs information about the required or acceptable properties of the data ("receivable" properties or "Receivables").

This Standard specifies the following schema resources to support these interactions.

- Returnables and Receivables at path: `/collections/{collectionId}/schema`
- Queryables at path `/collections/{collectionId}/queryables`
- Sortables at path `/collections/{collectionId}/sortables`

6.3. Requirements and recommendations for profiles

RFC 6906 The 'profile' Link Relation Type defines the concept of a profile to support such use cases.

Profiles are a general mechanism that allows to support variations in the resource representation without the need for a proliferation of a new media type for each variant. A profile does not to alter the semantics of the resource representation itself, but allows clients to learn about additional semantics that are associated with the resource representation, in addition to those defined by the media type.

To request one or more profiles, a query parameter "profile" can be used. The server determines the applicable profile(s) for the selected media type.

In total, four profiles are specified for schemas and three profiles are specified for geospatial data.

Chapter 7. Requirements Class "Schemas"

The Requirements Class "Schemas" specifies basic provisions for schemas for a collection of geospatial data, such as the features in a feature collection, and the representation of a schema in JSON Schema.

The schema represents a logical model, independent of the format in which the data is encoded when a representation is requested via a HTTP request.

Requirements Class	http://www.opengis.net/spec/ogcapi-common-3/1.0/req/schemas
Target type	Schema for a collection of geospatial data
Dependency	JSON Schema: A Media Type for Describing JSON Documents
Indirect Dependency	JSON Schema Validation: A Vocabulary for Structural Validation of JSON
Indirect Dependency	Simple feature access - Part 1: Common architecture
Indirect Dependency	The Unified Code for Units of Measure
Indirect Dependency	QUDT Units Vocabulary

7.1. Schema representation

Requirement 1	/req/schemas/json-schema
A	The schema SHALL be a valid JSON Schema.
B	The schema SHALL have the following characteristics: • "$schema" is "https://json-schema.org/draft/2020-12/schema"; • "$id" is a HTTP(S) URI without query parameters that returns the schema, if requested with the header "Accept: application/schema+json"; and • "type" is "object".

In addition to the JSON Schema data types, geospatial data typically also uses spatial and temporal data types.

Requirement 2	/req/schemas/properties
A	Each property SHALL include a "type" member, except for spatial properties.
B	Each spatial property SHALL not include a "type" or "$ref" member.

C	Each spatial property SHALL include a "format" member with a string value "geometry", followed by a hyphen, followed by the name of the geometry type in lower case. I.e., the values for the core Simple Feature geometry types are: "geometry-point", "geometry-multipoint", "geometry-linestring", "geometry-multilinestring", "geometry-polygon", "geometry-multipolygon", and "geometry-geometrycollection". In addition, the following special values are supported: "geometry-any" as the wildcard for any geometry type, "geometry-point-or-multipoint" for a Point or MultiPoint, "geometry-linestring-or-multilinestring" for a LineString or MultiLineString, and "geometry-polygon-or-multipolygon" for a Polygon or MultiPolygon.
D	Each temporal property that represents an instant (including the start or end of an interval) SHALL be a "string" literal with the appropriate format (e.g., "date-time" or "date", depending on the temporal granularity).
E	Each temporal property that represents an interval SHALL either be a "string" literal with format "interval-iso8601" (two instants with the same granularity separated by a "/") or an array with format "interval-array" consisting of two "string"s representing the interval start and end instants. In both variants, unbounded interval ends are represented by the string "..".
F	As a general rule, if the data type of a property has to be more specific than the JSON Schema data types ("number", "integer", "string"), the "format" keyword SHALL be used.
G	Properties that are only applicable when creating new data or updating existing data SHALL include "writeOnly: true".
H	Properties that are only applicable when data is fetched SHALL include "readOnly: true".

The following recommendations are intended to simplify parsing a schema and to help understanding the meaning and representation of the properties.

Recommendation 1	/rec/schemas/properties
A	Each property SHOULD have a human readable title ("title") and, where necessary for the understanding of the property, a description ("description").
B	The "type" SHOULD be one of the following values: "string" (string or temporal properties), "number"/"integer" (numeric properties), "boolean" (boolean properties), "object" (object properties) or "array" (array properties).
C	Properties that represent a URI SHOULD be represented as a string with format "uri" or "uri-reference".

D	Properties that represent a URI template SHOULD be represented as a string with format "uri-template".
E	Properties that represent a UUID SHOULD be represented as a string with format "uuid".
F	Properties that represent a password SHOULD be represented as a string with format "password" as a hint to obscure the value (defined in OpenAPI 3.1).
G	Properties that represent a 32-bit floating point number SHOULD be represented as a number with format "float" (defined in OpenAPI 3.1).
H	Properties that represent a 64-bit floating point number SHOULD be represented as a number with format "double" (defined in OpenAPI 3.1).
I	Properties that represent a 32-bit signed integer SHOULD be represented as an integer with format "int32" (defined in OpenAPI 3.1).
J	Properties that represent a 64-bit signed integer SHOULD be represented as an integer with format "int64" (defined in OpenAPI 3.1).
K	Properties that represent a 32-bit unsigned integer SHOULD be represented as an integer with format "uint32".
L	Properties that represent a 64-bit unsigned integer SHOULD be represented as an integer with format "uint64".
M	In general, "format" values SHOULD be from the JSON Schema specifciation, the OpenAPI 3.1 specification or the OGC format register.
N	For string properties, "minLength", "maxLength", "enum" and/or "pattern" SHOULD be provided, where applicable.
O	For numeric properties, "multipleOf", "minimum", "exclusiveMinimum", "maximum", "exclusiveMaximum" SHOULD be provided, where applicable.
P	For integer properties that represent enumerated values (except for codelist values), "enum" SHOULD be provided.
Q	For string or integer properties that represent codelist values, one of the profiles "codelist-inline" or "codelist-ref" (see Requirements Class "Profiles for codelists") SHOULD be applied.
R	For array properties, the property SHOULD consist of items that are strings, numbers or objects.
S	Required properties SHOULD be included in "required".

T	The JSON Schema keywords SHOULD be constrained to those mentioned in this recommendation, requirement `/req/schemas/properties` and requirements in the *additional keywords* section below.
U	"$ref" SHOULD NOT be used, schemas that are reused SHOULD be dereferenced and represented inline.

NOTE The OGC format register needs to be established.

7.2. Additional keywords

7.2.1. General rules

In order to be able to map the logical schema to a format-specific schema, extensions to the JSON Schema vocabulary are needed.

Requirement 3	/req/schemas/additional-keywords
A	Additional keywords specified by OGC SHALL start with "x-ogc-".

CAUTION The next version of JSON Schema will likely restrict the use of additional keywords. As a result, this Standard may need to change the prefix or change how the vocabulary is extended.

7.2.2. Properties with a specific role

Requirement 4	/req/schemas/role
A	The keyword "x-ogc-role" SHALL be used to declare a specific role of the property.
B	The value of the keyword "x-ogc-role" SHALL be a string.

Requirement 5	/req/schemas/role-id
A	A property with "x-ogc-role" set to "id" SHALL be the identifier of the resource in the collection that contains the resource.
B	A property with role "id" SHALL either be a string or integer.
C	Only one property in a schema SHALL have "x-ogc-role" with a value "id".

7.2.3. Ordering properties

For cases, where the properties of the resource should be ordered in some representations of the data, the sequence of the properties can be expressed using a keyword "x-ogc-propertySeq". The keyword is used to declare a hint about the relative position of the property in the object. Typical use cases are:

- the representation of the schema in a human-readable format (e.g., HTML) to show related properties next to each other;
- the representation of the data in a human-readable format (e.g., HTML) to show related property values next to each other; and
- the representation of the data in a format that requires a specific order of properties (e.g., XML based on an XML Schema that uses a `sequence` for the property elements).

Requirement 6	/req/schemas/property-seq
A	The keyword "x-ogc-propertySeq" SHALL be used to declare a hint about the relative position of the property in the object.
B	The value of the keyword "x-ogc-propertySeq" SHALL be an integer representing the relative position in ascending order.

Recommendation 2	/req/schemas/property-seq-unique
A	For each object, the value of the keyword "x-ogc-propertySeq" SHOULD be unique for each property.

7.2.4. The unit of measure for a numeric property

In geospatial data, numeric property values often represent a measurement and have a unit of measure. For fixed units, this can be expressed in the schema using the keyword "x-ogc-unit".

Requirement 7	/req/schemas/unit
A	The keyword "x-ogc-unit" SHALL be used to declare the unit of measure of the property.
B	The value of the keyword "x-ogc-unit" SHALL be a string representing the unit of measure.
C	The value of the keyword "x-ogc-unit" SHALL be the case sensitive UCUM representation ("c/s") unless a different language / register for units is identified in keyword "x-ogc-unitLang".
D	The value for UCUM, if explicitly declared as the language for units in keyword "x-ogc-unitLang", SHALL be "UCUM".
E	For specifying a unit from QUDT Units Vocabulary, "x-ogc-unitLang", SHALL be "QUDT".
F	For specifying a unit from QUDT Units Vocabulary, the value of the keyword "x-ogc-unit" SHALL be the URI of the unit.

Communities or other OGC Standards can specify additional values for other unit languages, e.g., units registered in the OGC Rainbow. For each language it must be specified how units have to be represented in the "x-ogc-unit" value.

NOTE UCUM provides a language to define units while QUDT provides a units vocabulary.

	The term "language" is used as it is the broader term.
NOTE	For example, the value for hectopascal is `hPa` in UCUM and `https://qudt.org/vocab/unit/HectoPA` in QUDT.

7.2.5. Semantic definition of a property

Requirement 8	/req/schemas/definition
A	The keyword "x-ogc-definition" SHALL be used to identify the semantic definition for the property.
B	The value of the keyword "x-ogc-definition" SHALL be a URI.

7.2.6. Special null values

Requirement 9	/req/schemas/nullvalues
A	The keyword "x-ogc-nullValues" SHALL be used to identify the values considered `null`, but that are for some reason represented by a value in the value domain of the data type of the property.
B	The value of the keyword "x-ogc-nullValues" SHALL be an array of values (numbers or strings), depending on the data type of the property.

7.3. Examples

The following example is the schema of a feature type representing cultural entities. The schema includes additional keywords that apply to feature data (specified in the next Clause).

Example 1. Schema of a "Cultural (Points)" feature type

```
{
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$id" : "https://demo.ldproxy.net/daraa/collections/CulturePnt/schema",
  "type" : "object",
  "title" : "Cultural (Points)",
  "description" : "Information about features on the landscape that have been
constructed by man.",
  "properties" : {
    "FID" : {
      "readOnly" : true,
      "x-ogc-role" : "id",
      "type" : "integer",
      "x-ogc-propertySeq": 1
    },
    "F_CODE" : {
      "title" : "Feature Type Code",
```

```
      "x-ogc-role" : "type",
      "enum" : [ "AK121", "AL012", "AL030", "AL130", "BH075" ],
      "type" : "string",
      "x-ogc-propertySeq": 2
    },
    "geometry" : {
      "x-ogc-role" : "primary-geometry",
      "format" : "geometry-point",
      "x-ogc-propertySeq": 3
    },
    "ZI001_SDV" : {
      "title" : "Last Change",
      "x-ogc-role" : "primary-instant",
      "format" : "date-time",
      "type" : "string",
      "x-ogc-propertySeq": 4
    },
    "UFI" : {
      "title" : "Unique Entity Identifier",
      "type" : "string",
      "x-ogc-propertySeq": 5
    },
    "ZI005_FNA" : {
      "title" : "Name",
      "type" : "string",
      "x-ogc-propertySeq": 6
    },
    "FCSUBTYPE" : {
      "title" : "Feature Subtype Code",
      "type" : "integer",
      "x-ogc-propertySeq": 7
    },
    "ZI037_REL" : {
      "title" : "Religious Designation",
      "enum" : [ -999999, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 ],
      "x-ogc-nullValues": [ -999999 ],
      "type" : "integer",
      "x-ogc-propertySeq": 8
    },
    "ZI006_MEM" : {
      "title" : "Memorandum",
      "type" : "string",
      "x-ogc-propertySeq": 9
    },
    "ZI001_SDP" : {
      "title" : "Source Description",
      "type" : "string",
      "x-ogc-propertySeq": 10
    }
  }
```

```
}
```

The next example is the schema of a feature type representing observations of atmospheric pressure.

Example 2. Schema of an "Atmospheric Pressure Observation" feature type

```
{
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$id" : "https://api.example.org/foo/bar/collections/atmospheric-pressure-
obs/schema",
  "type" : "object",
  "title" : "Atmospheric Pressure Observation",
  "properties" : {
    "id" : {
      "readOnly" : true,
      "x-ogc-role" : "id",
      "type" : "integer",
      "x-ogc-propertySeq": 1
    },
    "location" : {
      "x-ogc-role" : "primary-geometry",
      "format" : "geometry-point",
      "x-ogc-propertySeq": 2
    },
    "phenomenonTime" : {
      "title" : "Phenomenon Time",
      "x-ogc-role" : "primary-instant",
      "type" : "string",
      "format" : "date-time",
      "x-ogc-propertySeq": 3
    },
    "result" : {
      "title" : "Atmospheric Pressure [hPa]",
      "description": "The pressure exerted by the weight of the air above it at
any point on the earth's surface. At sea level the atmosphere will support a
column of mercury about 760 mm high. This decreases with increasing altitude. The
standard value for the atmospheric pressure at sea level is 1.01325 hectopascal.",
      "type" : "number",
      "x-ogc-definition":
"https://qudt.org/vocab/quantitykind/AtmosphericPressure",
      "x-ogc-unit": "https://qudt.org/vocab/unit/HectoPA",
      "x-ogc-unitLang": "QUDT",
      "x-ogc-propertySeq": 4
    },
  }
}
```

The next example is the schema of a land cover collection.

Example 3. Schema of a land cover collection

```
{
   "$schema" : "https://json-schema.org/draft/2020-12/schema",
   "$id" : "https://example.com/ogcapi/collections/landcover/schema",
   "title" : "Land Cover",
   "type" : "object",
   "properties" : {
      "LC" : {
        "title" : "Land Cover",
        "type" : "integer",
        "x-ogc-propertySeq" : 1,
        "oneOf": [
          { "const": 0, "title": "no data" },
          { "const": 1, "title": "vegetation" },
          { "const": 2, "title": "bare soils" },
          { "const": 3, "title": "water" },
          { "const": 4, "title": "clouds" }
        ],
        "x-ogc-nullValues": [ 0 ]
      }
   }
}
```

Chapter 8. Requirements Class "Advanced property roles"

The Requirements Class "Advanced property roles" specifies additional property roles for geospatial data. These roles are in particular applicable to feature data.

Requirements Class	http://www.opengis.net/spec/ogcapi-common-3/1.0/req/advanced-property-roles
Target type	Schema for a collection of geospatial data
Dependency	Requirements Class "Schemas"

To understand how geospatial data is represented in a specific data format, additional property roles need to be identified in the logical schema. The mapping of the roles to the data representation depends on the format.

For example, the role "primary-geometry" is needed in feature types with multiple spatial properties to decide, for example:

- Which geometry is encoded as the geometry of a feature in formats such as FlatGeobuf, Shapefile or Mapbox Vector Tiles, which only support a single geometry; or
- Which geometry is encoded in the "geometry" member in GeoJSON.

The other roles specified in this requirements class support formats that support representing primary temporal information or the type, such as JSON-FG.

8.1. Primary geometry

If the schema defines multiple spatial properties, the role "primary-geometry" can be used to identify the primary geometry property.

Requirement 10	/req/advanced-property-roles/role-primary-geometry
A	A property with "x-ogc-role" set to "primary-geometry" SHALL be a spatial property.
B	At most one property in a schema SHALL have "x-ogc-role" with a value "primary-geometry".
C	If the schema has only one spatial property, the property SHALL be the primary geometry even if the property is not explicitly tagged with the role "primary-geometry".

NOTE	Since only a single property can be tagged in the schema as the primary geometry, varying primary geometries at different zoom levels are currently not supported. This would require the capability to assign the role depending on the zoom level.

8.2. Primary temporal information

If the schema defines multiple temporal properties, the roles "primary-instant", "primary-interval-start" and "primary-interval-end" can be used to identify the primary temporal information.

Requirement 11	/req/advanced-property-roles/role-primary-instant
A	A property with "x-ogc-role" set to "primary-instant" SHALL be a temporal property.
B	At most one property in a schema SHALL have "x-ogc-role" with a value of "primary-instant".
C	If the schema has only one temporal property, the property SHALL be the primary temporal information even if the property is not explicitly tagged with the role "primary-instant".

Requirement 12	/req/advanced-property-roles/role-primary-interval-start
A	A property with "x-ogc-role" set to "primary-interval-start" SHALL be a temporal property.
B	At most one property in a schema SHALL have "x-ogc-role" with a value "primary-interval-start".

Requirement 13	/req/advanced-property-roles/role-primary-interval-end
A	A property with "x-ogc-role" set to "primary-interval-end" SHALL be a temporal property.
B	At most one property in a schema SHALL have "x-ogc-role" with a value "primary-interval-end".

Requirement 14	/req/advanced-property-roles/role-primary-interval
A	A property with "x-ogc-role" set to "primary-interval" SHALL be a temporal property.
B	At most one property in a schema SHALL have "x-ogc-role" with a value "primary-interval".

Requirement 15	/req/advanced-property-roles/primary-temporal-constraints
A	If a schema has a property with role "primary-instant", the schema SHALL NOT have properties with role "primary-interval-start", "primary-interval-end", or "primary-interval".
B	If a schema has a property with role "primary-interval", the schema SHALL NOT have properties with role "primary-interval-start", "primary-interval-end", or "primary-instant".
C	If a schema has properties with both roles "primary-interval-start" and "primary-interval-end", both properties SHALL have the same temporal granularity ("date" or "date-time").

D	If a schema has a property with only one the roles "primary-interval-start" or "primary-interval-end", the primary temporal information SHALL be the half-bounded interval with the specified start or end.

8.3. Type

If the data is organized as features with a property representing the feature type, the role "type" can be used for this property.

Requirement 16	/req/advanced-property-roles/role-type
A	A property with "x-ogc-role" set to "type" SHALL be a string property.
B	At most one property in a schema SHALL have "x-ogc-role" with a value "type".

8.4. Examples

The previous Clause includes an example with the roles "id", "primary-geometry", "primary-instant" and "type".

The following example is the schema of a feature type with the roles "id", "primary-geometry", "primary-interval-start" and "primary-interval-end".

Example 4. Schema example of a feature type

```
{
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$id" : "https://demo.ldproxy.net/cshapes/collections/boundary/schema",
  "title" : "Historical Country Boundaries",
  "description" : "Core territories of the given country, excluding colonies and
other dependent territories, between the start and end date.",
  "type" : "object",
  "properties" : {
    "fid" : {
      "title" : "Unique identifier",
      "readOnly" : true,
      "x-ogc-role" : "id",
      "type" : "integer"
    },
    "name" : {
      "title" : "Country name",
      "type" : "string"
    },
    "area_km2" : {
      "title" : "Area (km²)",
      "description" : "The polygon area in square kilometers, computed using an
```

```
Eckert VI projection.",
      "type" : "number",
      "x-ogc-unit": "km2"

    },
    "capname" : {
      "title" : "Country capital",
      "description" : "The name of the country capital.",
      "type" : "string"
    },
    "caplong" : {
      "title" : "Longitude of the capital",
      "description" : "Longitude of the capital, in decimal degrees",
      "type" : "number"
    },
    "caplat" : {
      "title" : "Latitude of the capital",
      "description" : "Latitude of the capital, in decimal degrees",
      "type" : "number"
    },
    "gwsdate" : {
      "title" : "Start date",
      "description" : "Start date of the entry.",
      "x-ogc-role" : "primary-interval-start",
      "format" : "date",
      "type" : "string"
    },
    "gwedate" : {
      "title" : "End date",
      "description" : "End date of the entry.",
      "x-ogc-role" : "primary-interval-end",
      "format" : "date",
      "type" : "string"
    },
    "gwcode" : {
      "title" : "Source identifier",
      "description" : "Numeric identifier code in the source data.",
      "type" : "integer"
    },
    "geometry" : {
      "x-ogc-role" : "primary-geometry",
      "format" : "geometry-multipolygon"
    }
  }
}
```

Chapter 9. Requirements Class "References"

The Requirements Class "References" specifies additional provisions for properties that reference another resource.

Requirements Class	http://www.opengis.net/spec/ogcapi-common-3/1.0/req/references
Target type	Schema for a collection of data
Dependency	Requirements Class "Schemas"

If resources have a property that represents a reference to another resource, the role "reference" can be used for this property. The property is a foreign key, e.g., the local feature identifier of a referenced feature in its collection.

Requirement 17	/req/references/role-reference
A	A property with "x-ogc-role" set to "reference" SHALL be a string or integer property.
B	If the property is of type "integer", the property with role "id" of the referenced resource type SHALL be of type "integer", too.
C	If all instances of the property reference geospatial data in a fixed collection of the same server, the id of that collection SHALL be provided in "x-ogc-collectionId".
D	If Requirements C does not apply, the URI template SHALL be provided in "x-ogc-uriTemplate" with a template parameter `resourceId`.

NOTE The requirements stated in this Standard only cover simple cases. More complex cases, such as references to geospatial data in different collections, are not covered.

Example 5. Schema example

This is the schema of a road accident feature type. The "roadSegment" property is a reference to the road segment on which the accident occurred, which is available in the same API.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://example.com/apis/roads/collections/accidents/schema",
  "type": "object",
  "title": "Road accidents",
  "properties": {
    "id": {
      "title": "Feature identifier",
      "readOnly": true,
      "x-ogc-role": "id",
      "type": "integer"
```

```
    },
    "timeOfAccident": {
      "title": "Time of the accident",
      "x-ogc-role": "primary-instant",
      "format": "date-time",
      "type": "string"
    },
    "roadSegment": {
      "title": "Road segment",
      "description": "Road segment on which the accident occured, identified by
its 16-character code (8 characters for the start and end node).",
      "x-ogc-role": "reference",
      "x-ogc-collectionId": "roadsegments",
      "type": "string"
    },
    "distanceFromStart": {
      "title": "Distance from start [m]",
      "description": "Distance from the start node of the road segment. The unit
is meter.",
      "minimum": 0.0,
      "type": "number",
      "x-ogc-unit": "m"
    },
    "geometry": {
      "x-ogc-role": "primary-geometry",
      "format": "geometry-point"
    }
  }
}
```

If the road segment would be available from a different OGC Web API, the "x-ogc-uriTemplate" keyword can be used with the `resourceId` parameter. The schema of the "roadSegment" property could look like the following.

```
{
  "title": "Road segment",
  "description": "Road segment on which the accident occured, identified by its
16-character code (8 characters for the start and end node).",
  "x-ogc-role": "reference",
  "x-ogc-uriTemplate":
"https://api.example.org/foo/bar/collections/roadsegments/items/{resourceId}",
  "type": "string"
}
```

A client that wants to fetch the road segment on which an accident occurred would use the URI template to construct the URI of the road segment, e.g., https://api.example.org/foo/bar/collections/roadsegments/items/5209062A5209047O, if the value of road segment property is "5209062A5209047O". The referenced feature will include a link to its "roadsegments" collection, or a client that is OGC-API-aware can determine the collection URI

("https://api.example.org/foo/bar/collections/roadsegments") from the URI.

Chapter 10. Requirements Class "Returnables and Receivables"

The Requirements Class "Returnables and Receivables" specifies the web resources representing the schema with the returnable and receivable properties of a collection in OGC Web APIs.

Requirements Class	http://www.opengis.net/spec/ogcapi-common-3/1.0/req/returnables-and-receivables
Target type	Web API
Indirect Dependency	Requirements Class "Schemas"

This requirements class supports clients that want to discover the list of resource properties with their types and constraints that is returned when fetching a representation of the resource or when creating a new or updating an existing resource.

Requirement 18	/req/returnables-and-receivables/link
A	A Returnables and Receivables resource SHALL be referenced from all Collection resources with a link with the link relation type `http://www.opengis.net/def/rel/ogc/1.0/schema`.

Requirement 19	/req/returnables-and-receivables/op
A	The Returnables and Receivables resource SHALL support the HTTP GET operation and the media type `application/schema+json`.

Requirement 20	/req/returnables-and-receivables/response
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	For responses that use `application/schema+json` as the `Content-Type` of the response, the response SHALL conform to Requirements Class "Schemas".
C	The "additionalProperties" member with a value of "true" (the default) or "false" is used to state the expected behavior with respect to properties that are not explicitly declared in the schema. If "additionalProperties" is set to "false", properties that are not explicitly declared in the schema SHALL NOT be allowed, otherwise they SHALL be allowed.

Chapter 11. Requirements Class "Queryables"

The Requirements Class "Queryables" specifies the web resources representing the schema with the queryable properties of a collection in OGC Web APIs.

Requirements Class	http://www.opengis.net/spec/ogcapi-common-3/1.0/req/queryables
Target type	Web API
Indirect Dependency	Requirements Class "Schemas"

This requirements class supports clients that want to discover the list of resource properties with their types and constraints that may be used to construct filter expressions on a collection of resources, for example, a set of features.

OGC API Standards do not assume that the content schema of a resource being queried is available for inspection. Therefore, a means needs to exist to interrogate an endpoint to determine the names and types of the properties that may be used to construct a filter expression ("queryables").

In addition, a publisher may want to support queryables that are not directly represented as resource properties in the content schema of the resource. Or the publisher may want to restrict filtering on certain properties. For example, because the backend datastore has not been configured to allow high-performance queries on those properties.

Requirement 21	/req/queryables/link
A	A Queryables resource SHALL be referenced from all Collection resources with a link with the link relation type `http://www.opengis.net/def/rel/ogc/1.0/queryables`.

Requirement 22	/req/queryables/op
A	The Queryables resource SHALL support the HTTP GET operation and the media type `application/schema+json`.

Requirement 23	/req/queryables/response
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	For responses that use `application/schema+json` as the `Content-Type` of the response, the response SHALL conform to Requirements Class "Schemas".

C	If `additionalProperties` is not included or has the default value `true`, any property name is valid in a filter expression on the collection that is evaluated by the Web API and the property reference SHALL evaluate to `null`, if the property does not exist for a resource. If `additionalProperties` is set to false, property references that are not explicitly declared in the queryables schema SHALL result in a 400 response.

NOTE Depending on the filter language, some properties cannot be supported in filter expressions. For example, if the filter language does not support predicates on object-valued properties, using such properties in a filter expression will result in an error.

Example 6. Filtering on complex data structures

Queryables also support simplified filtering on complex data structures. This example illustrates one potential approach to define a queryable on specific SpatioTemporal Asset Catalog (STAC) assets. The following is a STAC snippet:

```
"features": [
  {
    "type": "Feature",
    "assets": {
      "B5": {
        "eo:bands": [
          {
            "common_name": "nir",
            "name": "B5"
          }
        ],
        "href": "https://landsat-pds.s3.us-west-
2.amazonaws.com/c1/L8/060/247/LC08_L1TP_060247_20180905_20180912_01_T1/LC08_L1TP_0
60247_20180905_20180912_01_T1_B5.TIF",
        "type": "image/tiff; application=geotiff; profile=cloud-optimized"
      }
    }
  }
]
```

A common type of filter is to fetch only resources that have in their `assets` object a value that has an `eo:bands` value that contains a `common_name` value of "nir".

This could be implemented using a queryable `common_band_names` defined as an array of strings:

```
{
  "$schema": "https://json-schema.org/draft/2019-09/schema",
  "$id": "https://example.net/stac/collections/landsat/queryables",
```

```
  "type": "object",
  "title": "A STAC item",
  "properties": {
    "common_band_names": {
      "title": "Common band names",
      "description": "an array of common band names included in the assets",
      "type" : "array",
      "items": {
        "type": "string"
      }
    }
  }
}
```

Internally this could be mapped to the following JSON Path for a STAC item (`item`):

`item['assets'][*]['eo:bands'][\*]['common_name']`

Note that the JSON Path is just included here to clarify the mapping. An implementation would use a mapping that fits its backend datastore, but these details are hidden from the user.

A client can now use, e.g., the CQL2 array predicates to filter with the queryable. Examples using the CQL2 text encoding:

- `A_CONTAINS(common_band_names, ["nir"])` - includes at least a 'nir' band
- `A_EQUALS(common_band_names, ["nir","blue"])` - 'nir' and 'blue' exist for the feature, but no other band

Chapter 12. Requirements Class "Sortables"

The Requirements Class "Sortables" specifies the web resources representing the schema with the sortable properties of a collection returned by Web API endpoint(s) implementing OGC API Standards.

Requirements Class	http://www.opengis.net/spec/ogcapi-common-3/1.0/req/sortables
Target type	Web API
Indirect Dependency	Requirements Class "Schemas"

This requirements class supports clients that want to discover the list of resource properties with their types and constraints that may be used to sort the resources in a collection.

OGC API Standards do not assume that the content schema of a resource being queried is available for inspection. Therefore, a means needs to exist to interrogate an endpoint to determine the names and types of the properties that may be used to sort data ("sortables").

In addition, a publisher may want to support sortable that are not directly represented as resource properties in the content schema of the resource. Or the publisher may want to restrict sorting on certain properties. For example, because the backend datastore has not been configured to allow high-performance queries on those properties.

Requirement 24	/req/sortables/link
A	A Sortables resource SHALL be referenced from all Collection resources with a link with the link relation type `http://www.opengis.net/def/rel/ogc/1.0/sortables`.

Requirement 25	/req/sortables/op
A	The Sortables resource SHALL support the HTTP GET operation and the media type `application/schema+json`.

Requirement 26	/req/sortables/response
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	For responses that use `application/schema+json` as the `Content-Type` of the response, the response SHALL conform to Requirements Class "Schemas".
C	No property SHALL be of type "object" or "array".
D	No property SHALL be a spatial property.

E	If `additionalProperties` is not included or has the default value `true`, any property name is valid in a sorting expression on the collection that is evaluated by the server and the property reference SHALL evaluate to `null`, if the property does not exist for a resource. If `additionalProperties` is set to false, property references that are not explicitly declared in the sortables schema SHALL result in a 400 response.

Chapter 13. Requirements Class "Profile query parameter"

The Requirements Class "Profile query parameter" specifies additional provisions for Web APIs that support profiles.

Requirements Class	http://www.opengis.net/spec/ogcapi-common-3/1.0/req/profile-parameter
Target type	Web API

Some resources may be represented in with variations in the same format, depending on the intended use of the representation. One example are references to another web resource (see the Requirements Class "Profiles for references").

RFC 6906 The 'profile' Link Relation Type defines the concept of a profile to support such use cases.

A profile is defined not to alter the semantics of the resource representation itself, but to allow clients to learn about additional semantics (constraints, conventions, extensions) that are associated with the resource representation, in addition to those defined by the media type and possibly other mechanisms.

To request one or more profiles, a query parameter "profile" can be used.

Requirement 27	/req/profile-parameter/profile-param

A	The GET operation on selected resources SHALL support a query parameter "profile" with the following characteristics (using an OpenAPI Specification 3.1 fragment):
	
	`name: profile`
	`in: query`
	`required: false`
	`schema:`
	`  type: array`
	`  items:`
	`  type: string`
	`style: form`
	`explode: false`

B	Each item SHALL be one of the follwoing: • The profile identifier of a profile in the OGC Profile Register (the `profileId` value in the URI template `http://www.opengis.net/def/profile/OGC/0/{profileId}`); • A HTTP(S) URI of a profile.

NOTE	Profile URIs could be URIs from the OGC Profile Register

(http://www.opengis.net/def/profile) but they do not have to be. For example, communities of interest can develop and document their our profile URIs without the involvement of OGC.

Permission 1	/per/profile-parameter/profile-param-default
A	The server MAY specify a default value for the query parameter "profile".

Determining the profile(s) of the response is part of the content negotiation process after the proactive content negotiation as specified by the HTTP RFC has been completed. The server determines the applicable profile(s) for the selected media type.

Different media types have different characteristics. A consequence is that it can be impossible to support a profile for a media type or it can be against the design goals of a media type to support a profile.

Permission 2	/per/profile-parameter/profiles-of-media-type
A	For any media type that can represent a resource, the server MAY support only a subset of the profiles offered for the resource.
B	The subset of supported profiles for a media type MAY be empty, too.

The server will select the profile(s) of the response, if any, from the list of profiles supported for the media type and resource.

Recommendation 3	/rec/profile-parameter/profile-negotiation
A	If the server supports one or more of the requested profile(s) for the media type and resource, these profiles SHOULD be used for the response.
B	The profile negotiation SHOULD NOT result in an error, e.g., because a requested profile cannot be provided.

Requirement 28	/req/profile-parameter/profile-param-response
A	The query parameter is not required, that is, omitting the parameter in a request SHALL not result in an error.
B	If the media type of the response supports web links in accordance with RFC 8288 Web Linking, the response SHALL include links to the selected profile(s) with the link relation type "profile".

Recommendation 4	/rec/profile-parameter/profile-link-header
A	The response SHOULD include links to the selected profile(s) in the HTTP response headers.

Chapter 14. Requirements Class "Profiles for references"

The Requirements Class "Profiles for references" specifies three profiles for representing references to other resources.

The requirements stated in this Standard cover only simple cases. More complex cases, such as references to geospatial data in different collections, are not covered.

Requirements Class	http://www.opengis.net/spec/ogcapi-common-3/1.0/req/profile-references
Target type	Collection of data
Indirect Dependency	Requirements Class "Returnables and Receivables"
Indirect Dependency	Requirements Class "Profile query parameter"
Indirect Dependency	Requirements Class "References"

Requirement 29	/req/profile-references/ref-profiles
A	If the Returnables and Receivables schema of a collection includes a property with role „reference“, the GET operation on resources providing representations of the data SHALL provide the query parameter profile.

For features, the query parameter "profile" will be applicable to the GET operation on the paths `/collections/{collectionId}/items` and `/collections/{collectionId}/items/{resourceId}`.

Requirement 30	/req/profile-references/rel-as-key
A	In the profile "rel-as-key" (`http://www.opengis.net/def/profile/OGC/0/rel-as-key`) a reference in the response SHALL be represented by the `resourceId` of the referenced resource (a string or integer, depending on the type of the identifier property in the referenced collection), if the property with role "reference" has the keyword "x-ogc-collectionId".

Example 7. Encoding of a road accident feature in GeoJSON and profile "rel-as-key"

The examples in this Clause use the example road accident feature schema.

```
{
  "type": "Feature",
  "id": 1,
  "geometry": {
```

```
    "type": "Point",
    "coordinates": [
      7.2789399,
      50.7772485
    ]
  },
  "properties": {
    "timeOfAccident": "2019-02-05T07:00:00Z",
    "roadSegment": "5209062A52090470",
    "distanceFromStart": 851.0
  },
  "links": [
    {
      "href": "http://www.opengis.net/def/profile/ogc/0/rel-as-key",
      "rel": "profile",
      "title": "Profile used in the response"
    }
  ]
}
```


Requirement 31	/req/profile-references/rel-as-uri
A	In the profile "rel-as-uri" (http://www.opengis.net/def/profile/OGC/0/rel-as-uri) a reference in the response SHALL be represented by an HTTP(S) URI.

Example 8. Encoding of the same road accident feature for the profile "rel-as-uri"


```
{
  "type": "Feature",
  "id": 1,
  "geometry": {
    "type": "Point",
    "coordinates": [
      7.2789399,
      50.7772485
    ]
  },
  "properties": {
    "timeOfAccident": "2019-02-05T07:00:00Z",
    "roadSegment":
"https://example.com/apis/roads/collections/roadsegments/items/5209062A52090470",
    "distanceFromStart": 851.0
  },
  "links": [
    {
      "href": "http://www.opengis.net/def/profile/ogc/0/rel-as-uri",
      "rel": "profile",
      "title": "Profile used in the response"
```

```
    }
  ]
}
```

Requirement 32	/req/profile-references/rel-as-link
A	In the profile "rel-as-link" (http://www.opengis.net/def/profile/OGC/0/rel-as-link) a reference in the response SHALL be represented by an object with the property "href" (the URI of the referenced resource) and, optionally, a property "title" (a human readable label, e.g. for a clickable link).

How the object is represented in a media type will depend on the media type. For example, in HTML it could be `<a href="{href}">{title}</a>`, in XML it could be an XLink, in JSON it could be an object with the properties.

Example 9. Encoding of a road accident feature in GeoJSON and profile "rel-as-link"

The feature includes a "roadSegment" property that is a reference to the road segment on which the accident occurred.

```
{
  "type": "Feature",
  "id": 1,
  "geometry": {
    "type": "Point",
    "coordinates": [
      7.2789399,
      50.7772485
    ]
  },
  "properties": {
    "timeOfAccident": "2019-02-05T07:00:00Z",
    "roadSegment": {
        "href":
"https://example.com/apis/roads/collections/roadsegments/items/5209062A52090470",
        "title": "Road L333, Segment 5209062A52090470, Hennef"
    },
    "distanceFromStart": 851.0
  },
  "links": [
    {
      "href": "http://www.opengis.net/def/profile/ogc/0/rel-as-link",
      "rel": "profile",
      "title": "Profile used in the response"
    }
  ]
}
```

Servers will typically not support the profile "rel-as-link" for media types that do not have an adequate representation for such an object. Examples include CSV or FlatGeobuf.

"rel-as-link" will typically be the most useful representation, if a human user interacts with the resource. This is because of the possibility to include a descriptive label of the referenced resource. Of course, this only applies if the server can provide a meaningful title for each link, in which case this information can help the user to understand the referenced resource.

Recommendation 5	/rec/profile-references/default
A	The default value of the query parameter "profile" SHOULD include the profile "rel-as-link", if the server can provide meaningful link titles.

Chapter 15. Requirements Class "Profiles for codelists"

A codelist is a list of codes with their definitions. The Requirements Class "Profiles for codelists" specifies two profiles for representing enumerated values (codes) with additional information about each value, typically including at least a human-readable label. The codes are either strings or integers.

When requesting a schema that contains properties with values from a codelist ("codelist-valued properties"), the response should not only enumerate the codes, but also provide access to additional information about each code.

Two different representations of a codelist-valued property are specified as profiles. It is up to the API publisher to decide which profile is applied when a schema is requested. The publisher may also choose to support both profiles or other profiles.

The two profiles are as follows.

- "codelist-inline": The schema of each codelist-valued property is represented by a "oneOf" schema with a schema for each code. The schema for each code has a "const" member with the code as the value. This representation has the advantage that all information is in the schema; no external information needs to be accessed.
- "codelist-ref": The schema of each codelist-valued property is represented by an "enum" member with the codes. In addition the codelist is referenced by a HTTP(S) URI using the keyword "x-ogc-codelistUri". This profile has the advantage that "enum" is easier to parse/handle than "oneOf" and that a separate resource for codelists may be useful for other purposes, too, and in fact may already be published.

Requirements Class	http://www.opengis.net/spec/ogcapi-common-3/1.0/req/profile-codelists
Target type	Schema for a collection of data
Dependency	Requirements Class "Schemas"

Requirement 33	/req/profile-codelists/consistency
A	All codelist-valued properties in a schema SHALL use the same profile.

Recommendation 6	/rec/profile-codelists/profile-link-header
A	The response SHOULD include a link to the applied codelist profile in the HTTP response headers.

Requirement 34	/req/profile-codelists/codelist-inline

A	If the profile "codelist-inline" (http://www.opengis.net/def/profile/OGC/0/codelist-inline) is applied, the schema of each property with values from a codelist SHALL be represented by a "oneOf" schema with a schema for each code.
B	If the profile "codelist-inline" is applied, the schema for each code SHALL have a "const" member with the code as the value.

Recommendation 7	/req/profile-codelists/codelist-inline-title
A	If the profile "codelist-inline" is applied, the schema for each code SHOULD have a "title" member.

Example 10. Profile "codelist-inline"

```
{
  "title": "Language",
  "description": "Languages identified by ISO 639-3 codes",
  "oneOf": [
    {
      "const": "eng",
      "title": "English"
    },
    {
      "const": "deu",
      "title": "German"
    },
    {
      "const": "fra",
      "title": "French"
    },
    {
      "const": "spa",
      "title": "Spanish"
    },
    {
      "const": "ell",
      "title": "Modern Greek"
    }
  ]
}
```

Requirement 35	/req/profile-codelists/codelist-ref

A	If the profile "codelist-ref" (http://www.opengis.net/def/profile/OGC/0/codelist-ref) is applied, the codelist SHALL be referenced by a HTTP(S) URI using the keyword "x-ogc-codelistUri".

This standard does not specify the representation of the referenced codelist. The publisher may choose to publish the codelist in media types that are suitable for the community using the API. The reason for this approach is that different communities have different preferences for the representation of codelists. Typical representations are HTML (for human readers using a web browser or search engines), SKOS (for semantic web applications), JSON (for web applications), Genericode, etc.

Note that the "x-ogc-codelistUri" keyword may be included, too, when the "codelist-inline" profile is applied.

Example 11. Profile "codelist-ref"

```
{
  "title": "Language",
  "description": "Languages identified by ISO 639-3 codes",
  "type": "string",
  "enum": [
    "eng",
    "deu",
    "fra",
    "spa",
    "ell"
  ],
  "x-ogc-codelistUri": "https://example.com/codelists/languages"
}
```

Permission 3	/rec/profile-codelists/options
A	A resource MAY support only one codelist profile.
B	If more than one profile is supported, any codelist profile MAY be the default profile for a schema resource.
C	The schema resource MAY support the "profile" query parameter to allow clients to request a specific profile supported by the API.

Chapter 16. Requirements Class "Profiles for value domains"

For properties, two profiles of value domains can be distinguished.

- "valid-domain": The domain of all possible values of a property, according to the underlying data model. The domain constraints are specified in the data specification. The valid domain is a static domain: it does not change over time (unless the data specification changes). No value should be accepted by the API that is not in the valid domain.
- "actual-domain": The domain of all actual values of a property in the dataset at the time of the request. All queries that select features with a filter that selects property values outside of the actual domain will return an empty result set. The actual domain is a subset of the valid domain and is dynamic: it changes over time as the dataset is updated.

Beyond declaring the data type of a property, the valid domain is expressed in the schema using the following keywords (see Requriement /req/schemas/properties and Recommendation /rec/schemas/properties).

- For textual values:
  - "enum": The property has a finite set of values;
  - "pattern": All property values match a regular expression pattern; and
  - "minLength", "maxLength": The property values have a length constraint.
- For numeric values:
  - "enum": The property has a finite set of values (integers only);
  - "minimum", "maximum", "exclusiveMinimum", "exclusiveMaximum": The property values are from a range; and
  - "multipleOf": The property values are always a multiple of the specified value.
- For geometric values:
  - "format": The property can take only certain geometry types (e.g., "geometry-point-or-multipoint").

For the actual domain, the values of the following keywords will tyipically be different from the valid domain.

- For textual values:
  - "enum": The list of actual values, if the values are from a finite set; and
  - "minLength", "maxLength": The actual range of the length of the text strings.
- For numeric values:
  - "enum": The list of actual values, if the values are from a finite set (integer only); and
  - "minimum", "maximum", "exclusiveMinimum", "exclusiveMaximum": The actual range of the values.

- For geometric values:
  - "format": The format value that matches the actual geometry types (e.g., "geometry-point").

Which profile is most useful depends on the semantics of the schema resource.

- The "actual-domain" profile is in particular useful for queryables to guide a client which values can be used to filter the current data. The same applies for the returnables to guide a client how to process or visualize the data (e.g., restrict a map legend to actual values). The "valid-domain" profile is still useful for these resources, but includes values that are not in the current data.
- The "valid-domain" profile is important for receivables so that a client knows the values that the API accepts for new or updates resources. The "actual-domain" profile is in general not useful for receivables.
- For sortables, the domain is in general less relevant and domain keywords could also be dropped altogether.

Whether an API generates a schema with valid domains or actual domains is the choice of the API and can depend on the type of the schema resource.

Providing the actual domain is more expensive than providing the valid domain, as it requires analyzing the data.

The valid domain is static and can be cached for longer periods of time. Depending on the update frequency of the dataset, the actual domain can also be cached for a certain period of time.

Requirements Class	http://www.opengis.net/spec/ogcapi-common-3/1.0/req/profile-domains
Target type	Schema for a collection of data
Dependency	Requirements Class "Schemas"

Requirement 36	/req/profile-domains/consistency
A	All properties with constraint keywords in a schema SHALL use the same profile.

Recommendation 8	/rec/profile-domains/profile-link-header
A	The response SHOULD include a link to the applied domain profile in the HTTP response headers.

Requirement 37	/req/profile-domains/actual-domain

A	If the profile "actual-domain" (http://www.opengis.net/def/profile/OGC/0/actual-domain) is applied, • "format" values starting with "geometry-" SHALL be restricted to the actual geometry types in the dataset at the time of the request; • "enum" values SHALL be restricted to values in the dataset at the time of the request; • "minLength" and "maxLength" values SHALL be restricted to the actual range of the length of the text strings in the dataset at the time of the request; • "minimum", "maximum", "exclusiveMinimum", and "exclusiveMaximum" values SHALL be restricted to the actual numeric range of the values in the dataset at the time of the request;

Permission 4	/rec/profile-domains/options
A	A resource MAY support only one domain profile.
B	If more than one profile is supported, any domain profile MAY be the default profile for a schema resource.
C	The schema resource MAY support the "profile" query parameter to allow clients to request a specific profile supported by the API.

Example 12. Profile "valid-domain"

Request: `GET /collections/CulturePnt/queryables?f=json`

In this case, the API returns the valid domain of the properties by default.


```
{
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$id" : "https://demo.ldproxy.net/daraa/collections/CulturePnt/queryables",
  "type" : "object",
  "title" : "Cultural (Points)",
  "properties" : {
    "ZI037_REL" : {
      "title" : "Religious Designation",
      "x-ogc-codelistUri" :
"https://demo.ldproxy.net/daraa/codelists/daraa%2Frel",
      "enum" : [ -999999, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 ],
      "type" : "integer"
    },
    "geometry" : {
      "title" : "geometry",
      "x-ogc-role" : "primary-geometry",
```

```
      "format" : "geometry-point-or-multipoint"
    },
    "F_CODE" : {
      "title" : "Feature Type Code",
      "x-ogc-codelistUri" :
"https://demo.ldproxy.net/daraa/codelists/daraa%2Ff_code",
      "x-ogc-role" : "type",
      "enum" : [ "AK121", "AL012", "AL030", "AL130", "BH075" ],
      "type" : "string"
    },
    "ZI001_SDV" : {
      "title" : "Last Change",
      "x-ogc-role" : "primary-instant",
      "format" : "date-time",
      "type" : "string"
    }
  },
  "additionalProperties" : false
}
```


Example 13. Profile "actual-domain"

Request: `GET /collections/CulturePnt/queryables?f=json&profile=actual-domain`

The API supports the query parameter "profile", which is used in this request to ask for the actual domain of the properties.


```
{
  "$schema" : "https://json-schema.org/draft/2020-12/schema",
  "$id" : "https://demo.ldproxy.net/daraa/collections/CulturePnt/queryables"
  "type" : "object"
  "title" : "Cultural (Points)",
  "properties" : {
    "ZI037_REL" : {
      "title" : "Religious Designation",
      "x-ogc-codelistUri" :
"https://demo.ldproxy.net/daraa/codelists/daraa%2Frel",
      "enum" : [ 3, 5, 7, 13 ],
      "type" : "integer"
    },
    "geometry" : {
      "title" : "geometry",
      "x-ogc-role" : "primary-geometry",
      "format" : "geometry-point"
    },
    "F_CODE" : {
      "title" : "Feature Type Code",
      "x-ogc-codelistUri" :
"https://demo.ldproxy.net/daraa/codelists/daraa%2Ff_code",
```

```
      "x-ogc-role" : "type",
      "enum" : [ "AL012", "BH075" ],
      "type" : "string"
    },
    "ZI001_SDV" : {
      "title" : "Last Change",
      "x-ogc-role" : "primary-instant",
      "format" : "date-time",
      "type" : "string"
    }
  },
  "additionalProperties" : false
}
```

Chapter 17. Media Types

The media type for JSON Schema documents is "application/schema+json".

Chapter 18. Security Considerations

See the following document clauses.

- OGC API - Features - Part 1: Core, Clause 11;
- JSON Schema: A Media Type for Describing JSON Documents, Clause 13;
- JSON Schema Validation: A Vocabulary for Structural Validation of JSON, Clause 10.

Annex A: Abstract Test Suite (Normative)

This test suite uses the Given-When-Then notation to specify the tests.

In order to execute tests against a Web API under test, the Web API needs to support a well-known format. Since all known implementations at this time support JSON, the relevant Conformance Classes expect support for JSON and add a dependency to the Conformance Class "GeoJSON" in OGC API - Features - Part 1: Core.

NOTE	In the future, an alternative dependency can be added to OGC API - Common - Part 1, Conformance Class "JSON" in combination with OGC API - Common - Part 2.

A Web API under test can require authorization. Any Executable Test Suite implementing this test suite should implement the following security schemes supported by OpenAPI 3.1: HTTP Authentication schemes "basic" and "bearer", API keys, and OAuth2 flow "authorizationCode".

A.1. Conformance Class "Schemas"

Conformance Class	
http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/schemas	
Target type	Schema for a collection of geospatial data
Requirements class	Requirements Class "Schemas"
Dependency	OGC API - Common - Part 1: Core, Conformance Class "JSON" or OGC API - Features - Part 1: Core, Conformance Class "GeoJSON"

A.1.1. Conformance Test 1

Test id:	/conf/schemas/basic-schema-inspection
Requirements:	/req/schemas/json-schema, /req/schemas/properties, /req/schemas/additional-keywords
Test purpose:	Inspect the schema

Test method:	Given: • the schema that is the subject under test; • a JSON Schema validator; When: • the URI in the "$id" member is dereferenced with a HTTP header "Accept: application/schema+json"; • validate the schema using the JSON Schema validator against the schema at https://json-schema.org/draft/2020-12/schema; Then: • assert that the value of the "$schema" member is "https://json-schema.org/draft/2020-12/schema"; • assert that the value of the "$id" member is a URI in the "http" or "https" scheme; • assert that the value of the "$id" member is a URI without a query parameter; • assert dereferencing the URI in the "$id" member with a HTTP header "Accept: application/schema+json" results in a 200-response with "Content-Type: application/schema+json"; • assert that the value of the "type" member is "object"; • assert that the schema is a valid JSON Schema using the JSON Schema validator by testing it against the schema at https://json-schema.org/draft/2020-12/schema; • assert that each member in "$.properties" has an object as its value and the object either includes a "type" member or no "type" and no "$ref" member a "format" member whose value starts with "geometry-"; • review the members in "$.properties" and check that if property is a temporal instant that has type "string" and "format: date" or "format: date-time"; • review the members in "$.properties" and check that if property is a temporal interval that has type "string" and "format: interval-iso8601" or type "array" and "format: interval-array" with two items of type "string"; • review the members in "$.properties" and check that if property has a value domain that is a named subset of a number or string that the "format" keyword is provided to indicate the value domain; • store the key of an arbitrary property with a "writeOnly: true" member, if one exists; • store the key of an arbitrary property with a "readOnly: true" member, if one exists;

A.1.2. Conformance Test 2

Test id:	/conf/schemas/role
Requirements:	/req/schemas/role
Test purpose:	Inspect uses of the "x-ogc-role" keyword
Test method:	Given: • the schema that is the subject under test; When: • n/a; Then: • assert that all values of "x-ogc-role" are a string.

A.1.3. Conformance Test 3

Test id:	/conf/schemas/role-id
Requirements:	/req/schemas/role-id
Test purpose:	Inspect uses of the "x-ogc-role" keyword with value "id"
Test method:	Given: • the schema that is the subject under test; When: • select the properties that include the "x-ogc-role" keyword with value "id"; Then: • assert that exactly one property is selected; • assert that the property has a "type" member with value "string" or "integer".

A.1.4. Conformance Test 4

Test id:	/conf/schemas/property-seq
Requirements:	/req/schemas/property-seq
Test purpose:	Inspect uses of the "x-ogc-propertySeq" keyword

Test method:	Given: • the schema that is the subject under test; When: • select the properties that include the "x-ogc-propertySeq" keyword; Then: • assert that all values of "x-ogc-propertySeq" are an integer.

A.1.5. Conformance Test 5

Test id:	/conf/schemas/unit
Requirements:	/req/schemas/unit
Test purpose:	Inspect uses of the "x-ogc-unit" keyword
Test method:	Given: • the schema that is the subject under test; When: • select the properties that include the "x-ogc-unit" or "x-ogc-unitLang" keywords; Then: • assert that all values of "x-ogc-unit" and "x-ogc-unitLang" are a string; • assert that all selected properties have "type" "number" or "integer"; • assert that all selected properties without "x-ogc-unitLang" have the keyword "x-ogc-unit" with a value that is a valid UCUM unit of measure in the case sensitive UCUM representation ("c/s"); • assert that all selected properties with a "x-ogc-unitLang" value of "UCUM" have the keyword "x-ogc-unit" with a value that is a valid UCUM unit of measure in the case sensitive UCUM representation ("c/s"); • assert that all selected properties with a "x-ogc-unitLang" value of "QUDT" have the keyword "x-ogc-unit" with a value that is a valid QUDT URI in the QUDT Unitx Vocabulary.

A.1.6. Conformance Test 6

Test id:	/conf/schemas/definition
Requirements:	/req/schemas/definition
Test purpose:	Inspect uses of the "x-ogc-definition" keyword

Test method:	Given: • the schema that is the subject under test; When: • select the properties that include the "x-ogc-definition" keyword; Then: • assert that all values of "x-ogc-definition" are a string that is a valid URI.

A.1.7. Conformance Test 7

Test id:	/conf/schemas/nullvalues
Requirements:	/req/schemas/nullvalues
Test purpose:	Inspect uses of the "x-ogc-nullValues" keyword
Test method:	Given: • the schema that is the subject under test; When: • select the properties that include the "x-ogc-nullValues" keyword; Then: • assert that all values of "x-ogc-nullValues" are an array of strings (if the value of the property is a string) or an array of numbers (if the value of the property is a number).

A.2. Conformance Class "Advanced property roles"

Conformance Class	
http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/advanced-property-roles	
Target type	Schema for a collection of geospatial data
Requirements class	Requirements Class "Advanced property roles"
Dependency	Conformance Class "Schemas"

A.2.1. Conformance Test 8

Test id:	/conf/advanced-property-roles/role-primary-geometry
Requirements:	/req/advanced-property-roles/role-primary-geometry
Test purpose:	Inspect uses of the "x-ogc-role" keyword with value "primary-geometry"

Test method:	Given: • the schema that is the subject under test; When: • select the properties that include the "x-ogc-role" keyword with value "primary-geometry"; Then: • assert that at most one property is selected; • assert that the property has no "type" and no "$ref" member, but a "format" member that starts with "geometry-"; • assert that if no property is selected, the schema has only a single spatial property (with a "format" member that starts with "geometry-").

A.2.2. Conformance Test 9

Test id:	/conf/advanced-property-roles/role-primary-temporal
Requirements:	/req/advanced-property-roles/role-primary-instant, /req/advanced-property-roles/role-primary-interval-start, /req/advanced-property-roles/role-primary-interval-end, /req/advanced-property-roles/role-primary-interval, /req/advanced-property-roles/primary-temporal-constraints
Test purpose:	Inspect uses of the "x-ogc-role" keyword with value "primary-instant", "primary-interval-start", "primary-interval-end", or "primary-interval"
Test method:	Given: • the schema that is the subject under test; When: • select the properties that include the "x-ogc-role" keyword with value "primary-instant", "primary-interval-start", "primary-interval-end", or "primary-interval"; Then: • assert that at most one property is selected with the exception of "primary-interval-start" and "primary-interval-end" which can be used together; • assert for role "primary-instant", "primary-interval-start" and "primary-interval-end" that each property is a string with a "format" member that is either "date" or "date-time"; • assert for role "primary-interval" that the property is either a string with "format: interval-iso8601" or an array with "format: interval-array" containing two string items.

A.2.3. Conformance Test 10

Test id:	/conf/advanced-property-roles/role-type
Requirements:	/req/advanced-property-roles/role-type
Test purpose:	Inspect uses of the "x-ogc-role" keyword with value "type"
Test method:	Given: • the schema that is the subject under test; When: • select the properties that include the "x-ogc-role" keyword with value "type"; Then: • assert that at most one property is selected; • assert that the property has "type" "string".

A.3. Conformance Class "References"

Conformance Class	
http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/references	
Target type	Schema for a collection of geospatial data
Requirements class	Requirements Class "References"
Dependency	Conformance Class "Schemas"

A.3.1. Conformance Test 11

Test id:	/conf/references/role-reference
Requirements:	/req/references/role-reference
Test purpose:	Inspect uses of the "x-ogc-role" keyword with value "reference"

Test method:	Given: • the schema that is the subject under test; • the schemas of the collections that are referenced via "x-ogc-collectionId"; When: • select the properties that include the "x-ogc-role" keyword with value "reference"; Then: • assert that each selected property has "type" "string" or "integer"; • assert that each selected property has either the keyword "x-ogc-uriTemplate" or "x-ogc-collectionId"; • assert that each selected property with the keyword "x-ogc-uriTemplate" has a value that is a URI template with the variable "resourceId"; • assert that each selected property with the keyword "x-ogc-collectionId" has the same type as the property with the "x-ogc-role" keyword with value "id" in the schema of the referenced collection.

A.4. Conformance Class "Returnables and Receivables"

Conformance Class	
http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/returnables-and-receivables	
Target type	Web API
Requirements class	Requirements Class "Returnables and Receivables"
Dependency	OGC API - Features - Part 1: Core, Conformance Class "GeoJSON"
Indirect Dependency	Conformance Class "Schemas"

A.4.1. Conformance Test 12

Test id:	/conf/returnables-and-receivables/get-conformance
Requirements:	n/a
Test purpose:	Check that the API declares support for the conformance class

Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); • authentication credentials (optional); When: • the request for the Conformance Declaration is executed ◦ method: GET ◦ path: "{apiURI}/conformance" ◦ header: "Accept: application/json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200", "Content-Type" header is "application/json"); • assert that "$.conformsTo" is a string array that includes the values "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/schemas" and "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/returnables-and-receivables"; • store the values of "$.conformsTo" that match the URI template "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/{cc}";

A.4.2. Conformance Test 13

Test id:	/conf/returnables-and-receivables/link
Requirements:	/req/returnables-and-receivables/link
Test purpose:	Check that a link to the schema resource exists

Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); When: • a request is executed to determine the list of collections ◦ method: GET ◦ path: "{apiURI}/collections"" ◦ header: "Accept: application/json" ◦ authentication, if authentication credentials are provided • a request is executed to each collection found in the Collections resource ◦ method: GET ◦ path: "{apiURI}/collections/{collectionId}" ◦ header: "Accept: application/json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200"); • assert that each Collection response includes a link in the JSON payload with "rel" set to "http://www.opengis.net/def/rel/ogc/1.0/schema"; • store the "href" value as the schema URI for the collection ("{schemaUri}"); • assert that the schema URI is "{apiUri}/collections/{collectionId}/schema"; • store, if the collection supports features (itemType is not set or set to "features" or "records").

A.4.3. Conformance Test 14

Test id:	/conf/returnables-and-receivables/op-and-response
Requirements:	/req/returnables-and-receivables/op, /req/returnables-and-receivables/response
Test purpose:	Check that the schema resource exists and conforms to dependencies

Test method:	Given: • the URIs of the schema resources for each collection ("{schemaUri}"); When: • a request is executed for each schema resource ◦ method: GET ◦ path: "{schemaURI}" ◦ header: "Accept: application/schema+json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200", "Content-Type" header is "application/schema+json"); • assert that each schema passes the conformance class "Schemas"; • if the conformance declaration includes the URI "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/advanced-property-roles", assert that each schema passes the conformance class "Advanced property roles"; • if the conformance declaration includes the URI "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/references", assert that each schema passes the conformance class "References";

A.4.4. Conformance Test 15

Test id:	/conf/returnables-and-receivables/response-1
Requirements:	/req/returnables-and-receivables/response
Test purpose:	Check that features are valid against the schema

Test method:	Given: • the schema resource of a collection that supports features; When: • requests are executed for the feature collection ◦ method: GET ◦ path: "{apiURI}/collections/{collectionId}/items" and "{apiURI}/collections/{collectionId}/items/{featureId}" ◦ header: "Accept: application/geo+json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution of the feature requests (status code is "200", "Content-Type" header is "application/geo+json"); • assert that each property is valid according to the schema ("id" is validated against the schema with "x-ogc-role: id", "geometry" is validated against the schema with "x-ogc-role: primary-geometry", all other properties are validated against the schema with the same name in "properties" except "writeOnly" properties, which are ignored); • if the schema includes "additionalProperties: false", assert that no properties are returned that are not specified in the schema; • if the schema includes "additionalProperties: false", assert that no properties are returned that are specified in the schema with "writeOnly: true".

A.4.5. Conformance Test 16

Test id:	/conf/returnables-and-receivables/response-2
Requirements:	/req/returnables-and-receivables/response
Test purpose:	Check that features conforming to the schema are accepted in CRUD operations

Test method:	Given: • the schema resource of a collection that supports features; • the OGC Web API supports CRUD operations on the feature collection; When: • requests are executed for each feature collection ◦ method: PUT, POST, where applicable ◦ path: "{apiURI}/collections/{collectionId}/items" (POST) and "{apiURI}/collections/{collectionId}/items/{featureId}" (PUT) ◦ header: "Content-Type: application/geo+json" ◦ body: a valid GeoJSON Feature, both requests that conform and do not conform to the schema ◦ authentication, if authentication credentials are provided Then: • assert successful execution for valid requests (status code is "201" for POST and "200" or "204" for PUT); • assert unsuccessful execution of invalid requests (status code is "400"); • if the schema includes "additionalProperties: false", assert that the request is rejected if the body contains a property that is not specified in the schema or that is specified as "readOnly".

A.5. Conformance Class "Queryables"

Conformance Class	
http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/queryables	
Target type	Web API
Requirements class	Requirements Class "Queryables"
Dependency	OGC API - Features - Part 1: Core, Conformance Class "GeoJSON"
Dependency	OGC API - Features - Part 3: Filtering, Conformance Class "Queryables as Query Parameters" or "Features Filter"
Indirect Dependency	Conformance Class "Schemas"

A.5.1. Conformance Test 17

Test id:	/conf/queryables/get-conformance
Requirements:	n/a

Test purpose:	Check that the API declares support for the conformance class
Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); • authentication credentials (optional); When: • the request for the Conformance Declaration is executed ◦ method: GET ◦ path: "{apiURI}/conformance" ◦ header: "Accept: application/json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200", "Content-Type" header is "application/json"); • assert that "$.conformsTo" is a string array that includes the values "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/schemas" and "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/queryables"; • store the values of "$.conformsTo" that match the URI template "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/{cc}";

A.5.2. Conformance Test 18

Test id:	/conf/queryables/link
Requirements:	/req/queryables/link
Test purpose:	Check that a link to the schema resource exists

Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); When: • a request is executed to determine the list of collections ◦ method: GET ◦ path: "{apiURI}/collections"" ◦ header: "Accept: application/json" ◦ authentication, if authentication credentials are provided • a request is executed to each collection found in the Collections resource ◦ method: GET ◦ path: "{apiURI}/collections/{collectionId}" ◦ header: "Accept: application/json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200"); • assert that each Collection response includes a link in the JSON payload with "rel" set to "http://www.opengis.net/def/rel/ogc/1.0/queryables"; • store the "href" value as the schema URI ("{schemaUri}"); • assert that the schema URI is "{apiUri}/collections/{collectionId}/queryables"; • store, if the collection supports features (itemType is not set or set to "features" or "records").

A.5.3. Conformance Test 19

Test id:	/conf/queryables/op-and-response
Requirements:	/req/queryables/op, /req/queryables/response
Test purpose:	Check that the schema resource exists and conforms to dependencies

Test method:	Given: • the URIs of the schema resources ("{schemaUri}"); When: • a request is executed for each schema resource ◦ method: GET ◦ path: "{schemaURI}" ◦ header: "Accept: application/schema+json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200", "Content-Type" header is "application/schema+json"); • assert that each schema passes the conformance class "Schemas"; • if the conformance declaration includes the URI "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/advanced-property-roles", assert that each schema passes the conformance class "Advanced property roles"; • if the conformance declaration includes the URI "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/references", assert that each schema passes the conformance class "References"; • store the key of selected properties with a "type" member; • store the key of an arbitrary property of the object as the spatial queryable of the filterable resource, if the value of member is an object that includes no type member and a format member with a value "geometry-{type}"" where "{type}"" is one of "point", "multipoint", "linestring", "multilinestring", "polygon", "multipolygon", "geometrycollection", "any", "point-or-multipoint", "linestring-or-multilinestring", or "polygon-or-multipolygon"; • store the value of the additionalProperties member or "true", if it is not provided; • if the conformance declaration includes the URI "http://www.opengis.net/spec/ogcapi-features-3/1.0/conf/queryables-query-parameters", assert that the OGC Web API passes the conformance class "Queryables as Query Parameters" for each collection that supports features; • if the conformance declaration includes the URI "http://www.opengis.net/spec/ogcapi-features-3/1.0/conf/features-filter", assert that the OGC Web API passes the conformance class "Features Filter" for each collection that supports features;

A.6. Conformance Class "Sortables"

Conformance Class	
http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/sortables	
Target type	Web API
Requirements class	Requirements Class "Sortables"
Dependency	OGC API - Features - Part 1: Core, Conformance Class "GeoJSON"
Indirect Dependency	Conformance Class "Schemas"

A.6.1. Conformance Test 20

Test id:	/conf/sortables/get-conformance
Requirements:	n/a
Test purpose:	Check that the API declares support for the conformance class
Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); • authentication credentials (optional); When: • the request for the Conformance Declaration is executed ◦ method: GET ◦ path: "{apiURI}/conformance" ◦ header: "Accept: application/json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200", "Content-Type" header is "application/json"); • assert that "$.conformsTo" is a string array that includes the values "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/schemas" and "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/sortables"; • store the values of "$.conformsTo" that match the URI template "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/{cc}";

A.6.2. Conformance Test 21

Test id:	/conf/sortables/link

Requirements:	/req/sortables/link
Test purpose:	Check that a link to the schema resource exists
Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); When: • a request is executed to determine the list of collections ◦ method: GET ◦ path: "{apiURI}/collections"" ◦ header: "Accept: application/json" ◦ authentication, if authentication credentials are provided • a request is executed to each collection found in the Collections resource ◦ method: GET ◦ path: "{apiURI}/collections/{collectionId}" ◦ header: "Accept: application/json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200"); • assert that each Collection response includes a link in the JSON payload with "rel" set to "http://www.opengis.net/def/rel/ogc/1.0/sortables"; • store the "href" value as the schema URI ("{schemaUri}"); • assert that the schema URI is "{apiUri}/collections/{collectionId}/sortables"; • store, if the collection supports features (itemType is not set or set to "features" or "records").

A.6.3. Conformance Test 22

Test id:	/conf/sortables/op-and-response
Requirements:	/req/sortables/op, /req/sortables/response
Test purpose:	Check that the schema resource exists and conforms to dependencies

Test method:	Given: • the URIs of the schema resources ("{schemaUri}"); When: • a request is executed for each schema resource ◦ method: GET ◦ path: "{schemaURI}" ◦ header: "Accept: application/schema+json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200", "Content-Type" header is "application/schema+json"); • assert that each schema passes the conformance class "Schemas"; • if the conformance declaration includes the URI "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/advanced-property-roles", assert that each schema passes the conformance class "Advanced property roles"; • if the conformance declaration includes the URI "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/references", assert that each schema passes the conformance class "References"; • assert that each property has a "type" member; • assert that no property has a "type" member with value "array" or "object";

NOTE	Tests that assert that the sortable properties can be used for sorting will be included in OGC API - Features - Part 8: Sorting.

A.7. Conformance Class "Profile query parameter"

Conformance Class	
http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/profile-parameter	
Target type	Web API
Requirements class	Requirements Class "Profile query parameter"
Dependency	OGC API - Features - Part 1: Core, Conformance Class "GeoJSON"

A.7.1. Conformance Test 23

Test id:	/conf/profile-parameter/get-conformance

Requirements:	n/a
Test purpose:	Check that the API declares support for the conformance class
Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); • authentication credentials (optional); When: • the request for the Conformance Declaration is executed ◦ method: GET ◦ path: "{apiURI}/conformance" ◦ header: "Accept: application/json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200", "Content-Type" header is "application/json"); • assert that "$.conformsTo" is a string array that includes the value "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/profile-parameter".

A.7.2. Conformance Test 24

Test id:	/conf/profile-parameter/profile-param
Requirements:	/req/profile-parameter/profile-param
Test purpose:	Check that the parameter "profile" is defined and supported

Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); • the path to resources that should support the "profile" query parameter ("{pathToProfiledResource}"); • authentication credentials (optional); When: • a request is executed for the API definition ◦ method: GET ◦ path: "{apiURI}/api" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200"); • assert that the "profile" query parameter is defined for each profiled resource as an optional list of comma-separated strings; • assert that the allowed values for the "profile" query parameter are either HTTP(S) URI or a non-URI value that can be converted to a URI using the URI template "http://www.opengis.net/def/profile/OGC/0/{profileId}" and that can be de-referenced (a GET or HEAD request returns a 200 status code); • store the allowed values for each profiled resource.

A.7.3. Conformance Test 25

Test id:	/conf/profile-parameter/profile-param-response
Requirements:	/req/profile-parameter/{conf-test
Test purpose:	Check that the parameter "profile" is defined and supported

Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); • the path to resources that should support the "profile" query parameter ("{pathToProfiledResource}"); • the allowed values for the "profile" query parameter for each profiled resource; • authentication credentials (optional); When: • a request is executed for each profiled resource ◦ method: GET ◦ path: "{apiURI}/{pathToProfiledResource}" ◦ query parameters: ▪ without a query parameter ▪ with query parameter "profile" with a value that is in the list of allowed values ▪ with query parameter "profile" with a value that is not in the list of allowed values ▪ with query parameter "profile" with two or more values that are in the list of allowed values ▪ with query parameter "profile" with two or more values where at least one is not in the list of allowed values ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200"), if no "profile" query parameter is provided or if the "profile" query parameter is provided with values that are in the list of allowed values; • assert unsuccessful execution (status code is "400"), if the "profile" query parameter is provided with values that are not in the list of allowed values; • assert that if the "profile" parameter was provided and the response payload includes links (e.g. a GeoJSON feature or feature collection), at least one link is included with "rel" set to "profile".

A.8. Conformance Class "Profiles for references"

Conformance Class
http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/profile-references

Target type	Collection of data
Requirements class	Requirements Class "Profiles for references"
Indirect Dependency	OGC API - Features - Part 1: Core, Conformance Class "GeoJSON"
Indirect Dependency	Conformance Class "Returnables and Receivables"
Indirect Dependency	Conformance Class "Profile query parameter"
Indirect Dependency	Conformance Class "References

A.8.1. Conformance Test 26

Test id:	/conf/profile-references/get-conformance
Requirements:	n/a
Test purpose:	Check that the API declares support for the conformance class
Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}") that contains the collection of data; • authentication credentials (optional); When: • the request for the Conformance Declaration is executed ◦ method: GET ◦ path: "{apiURI}/conformance" ◦ header: "Accept: application/json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200", "Content-Type" header is "application/json"); • assert that "$.conformsTo" is a string array that includes the value "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/profile-references".

A.8.2. Conformance Test 27

Test id:	/conf/profile-references/ref-profiles

Requirements:	/req/profile-references/ref-profiles
Test purpose:	Check that the parameter "profile" is defined and supported
Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); • the IDs of collections ("{collectionId}") that include references in their returnables and receivables schema; • the path to sub-resources that should support the "profile" query parameter ("{subpathToProfiledResource}"); • authentication credentials (optional); When: • a request is executed for profiled resources ◦ method: GET ◦ path: "{apiURI}/collections/{collectionId}/{subpathToProfiledResource}" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200"); • assert that the API passes the Conformance Class "Returnables and Receivables" and that the schema passes the Conformance Class "References" for the identified collections; • assert that the API passes the Conformance Class "Profile query parameter" for the path "collections/{collectionId}/{subpathToProfiledResource}" and the allowed values "rel-as-key", "rel-as-uri", and "rel-as-link";

A.8.3. Conformance Test 28

Test id:	/conf/profile-references/rel-as-key
Requirements:	/req/profile-references/rel-as-key
Test purpose:	Check the profile "rel-as-key" for references

Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); • the IDs of collections ("{collectionId}") that include references in their returnables and receivables schema; • the path to sub-resources that should support the "profile" query parameter ("{subpathToProfiledResource}"); • authentication credentials (optional); When: • a request is executed for profiled resources ◦ method: GET ◦ path: "{apiURI}/collections/{collectionId}/{subpathToProfiledResource}" ◦ query parameter: "profile=rel-as-key" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200"); • if the response includes a link with "rel" set to "profile" and "href" set to "http://www.opengis.net/def/profile/OGC/0/rel-as-key", ◦ assert that all properties with "x-ogc-role" set to "reference" and which include the keyword "x-ogc-collectionId" are either a string or an integer.

A.8.4. Conformance Test 29

Test id:	/conf/profile-references/rel-as-uri
Requirements:	/req/profile-references/rel-as-uri
Test purpose:	Check the profile "rel-as-uri" for references

Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); • the IDs of collections ("{collectionId}") that include references in their returnables and receivables schema; • the path to sub-resources that should support the "profile" query parameter ("{subpathToProfiledResource}"); • authentication credentials (optional); When: • a request is executed for profiled resources ◦ method: GET ◦ path: "{apiURI}/collections/{collectionId}/{subpathToProfiledResource}" ◦ query parameter: "profile=rel-as-uri" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200"); • if the response includes a link with "rel" set to "profile" and "href" set to "http://www.opengis.net/def/profile/OGC/0/rel-as-uri", ◦ assert that all properties with "x-ogc-role" set to "reference" are a URI in the HTTP or HTTPS scheme.

A.8.5. Conformance Test 30

Test id:	/conf/profile-references/rel-as-link
Requirements:	/req/profile-references/rel-as-link
Test purpose:	Check the profile "rel-as-link" for references

Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); • the IDs of collections ("{collectionId}") that include references in their returnables and receivables schema; • the path to sub-resources that should support the "profile" query parameter ("{subpathToProfiledResource}"); • authentication credentials (optional); When: • a request is executed for profiled resources ◦ method: GET ◦ path: "{apiURI}/collections/{collectionId}/{subpathToProfiledResource}" ◦ query parameter: "profile=rel-as-link" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200"); • if the response includes a link with "rel" set to "profile" and "href" set to "http://www.opengis.net/def/profile/OGC/0/rel-as-link", ◦ assert that all properties with "x-ogc-role" set to "reference" are an object with a "href" member with a URI as the value.

A.9. Conformance Class "Profiles for codelists"

Conformance Class	
http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/profile-codelists	
Target type	Schema for a collection of data
Requirements class	Requirements Class "Profiles for codelists"
Indirect Dependency	OGC API - Features - Part 1: Core, Conformance Class "GeoJSON"
Indirect Dependency	Conformance Class "Schemas

A.9.1. Conformance Test 31

Test id:	/conf/profile-codelists/get-conformance
Requirements:	n/a

Test purpose:	Check that the API declares support for the conformance class
Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}") that contains the collection of data; • authentication credentials (optional); When: • the request for the Conformance Declaration is executed ◦ method: GET ◦ path: "{apiURI}/conformance" ◦ header: "Accept: application/json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200", "Content-Type" header is "application/json"); • assert that "$.conformsTo" is a string array that includes the value "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/profile-codelists".

A.9.2. Conformance Test 32

Test id:	/conf/profile-codelists/codelists
Requirements:	/req/profile-codelists/consistency, /req/profile-codelists/codelists-inline, /req/profile-codelists/codelists-ref
Test purpose:	Check that all codelists use the same profile and that responses are consistent with the declared profile

Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); • the IDs of collections ("{collectionId}") that include codelists in their queryables or returnables and receivables schema ("{subpathToSchema}"); • authentication credentials (optional); When: • a request is executed for schema that is the subject under test ◦ method: GET ◦ path: "{apiURI}/collections/{collectionId}/{subpathToSchema}" ◦ header: "Accept: application/schema+json" ◦ query parameters: ▪ without a query parameter ▪ with query parameter "profile=codelists-inline" ▪ with query parameter "profile=codelists-ref" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200", "Content-Type: application/schema+json"); • assert that the response passes the Conformance Class "Schemas"; • assert that the API passes the Conformance Class "Profile query parameter" for the path "collections/{collectionId}/{subpathToSchema}" and the potentially allowed values "codelists-inline" and "codelists-ref"; • assert that all properties that specify enumerated values (either using the "enum" keyword or using a "oneOf" schema with a "const" member in each schema) follow the same pattern: ◦ if a property uses type "oneOf" with a "const" member in every schema, assert that no property uses the "enum" keyword and the "x-ogc-codelistUri" keyword; ◦ if a property uses type "string" or "integer" and both the "enum" and the "x-ogc-codelistUri" keyword, assert that no property uses type "oneOf" with a "const" member in every schema; ◦ if a property uses type "oneOf" with a "const" member in every schema, assert that the response does not include a link with "rel" set to "profile" with "href" set to "http://www.opengis.net/def/profile/OGC/0/codelists-ref"; ◦ if a property uses type "string" or "integer" and both the "enum" and the "x-ogc-codelistUri" keyword, assert that the response does not

A.10. Conformance Class "Profiles for value domains"

Conformance Class	
http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/profile-domains	
Target type	Schema for a collection of data
Requirements class	Requirements Class "Profiles for value domains"
Indirect Dependency	OGC API - Features - Part 1: Core, Conformance Class "GeoJSON"
Indirect Dependency	Conformance Class "Schemas

A.10.1. Conformance Test 33

Test id:	/conf/profile-domains/get-conformance
Requirements:	n/a
Test purpose:	Check that the API declares support for the conformance class
Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}") that contains the collection of data; • authentication credentials (optional); When: • the request for the Conformance Declaration is executed ◦ method: GET ◦ path: "{apiURI}/conformance" ◦ header: "Accept: application/json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200", "Content-Type" header is "application/json"); • assert that "$.conformsTo" is a string array that includes the value "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/profile-domains".

A.10.2. Conformance Test 34

Test id:	/conf/profile-domains/domains
Requirements:	/req/profile-domains/consistency, /req/profile-domains/actual-domain

Test purpose:	Check that all value domains use the same profile and that responses are consistent with the declared profile

Test method:	Given: • the landing page URI of the OGC Web API ("{apiURI}"); • the IDs of collections ("{collectionId}") that include properties with "enum", "pattern", "minLength", "maxLength", "minimum, "maximum", "exclusiveMinimum", "exclusiveMaximum", "format" keywords in their queryables or returnables and receivables schema ("{subpathToSchema}"); • authentication credentials (optional); When: • a request is executed for schema that is the subject under test ◦ method: GET ◦ path: "{apiURI}/collections/{collectionId}/{subpathToSchema}" ◦ header: "Accept: application/schema+json" ◦ query parameters: ▪ without a query parameter ▪ with query parameter "profile=actual-domain" ▪ with query parameter "profile=valid-domain" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200", "Content-Type: application/schema+json"); • assert that the response passes the Conformance Class "Schemas"; • assert that the API passes the Conformance Class "Profile query parameter" for the path "collections/{collectionId}/{subpathToSchema}" and the potentially allowed values "actual-domain" and "valid-domain"; • if the response includes a link with "rel" set to "profile" with "href" set to "http://www.opengis.net/def/profile/OGC/0/actual-domain", assert through spot checks that valid values for properties with "enum", "pattern", "minLength", "maxLength", "minimum, "maximum", "exclusiveMinimum", "exclusiveMaximum", and "format" keywords can be found in the data of the collection; • if the response includes a link with "rel" set to "profile" with "href" set to "http://www.opengis.net/def/profile/OGC/0/valid-domain" and if the OGC Web API supports CRUD operations on the collection, assert through spot checks that valid values for properties with "enum", "pattern", "minLength", "maxLength", "minimum, "maximum", "exclusiveMinimum", "exclusiveMaximum", and "format" keywords can be submitted to create new resources in the collection.

input: URI of resources, list of values

check definition of the query parameter profile on GET

check allowed values

test without profile query parameter, 200

check links to profiles

---

conditional check

rel-as-key, rel-as-uri, rel-as-link

- The URI of the schema resource in the API ("{apiURI}/{relativePathToSchema}");

NOTE	The Conformance Class "JSON" in Common Core has a dependency to the Conformance Classes "Core" and "Landing Page"; that is, testing against "JSON" will automatically test against the dependencies.

The following table lists input given to all tests in this conformance class:

- The landing page URI of the OGC Web API ("{apiURI}");
- The URI of the schema resource in the API ("{apiURI}/{relativePathToSchema}");
- Authentication credentials (optional).

A.10.3. Conformance Test 35

Test id:	/conf/profile-domains/get-conformance
Requirements:	n/a
Test purpose:	Check that the API declares support for the conformance class

Test method:	Given: • n/a When: • the request for the Conformance Declaration is executed ◦ method: "GET" ◦ path: "{apiURI}/conformance" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200", "Content-Type" header is "application/json"); • assert that "$.conformsTo" is a string array that includes the value "http://www.opengis.net/spec/ogcapi-common-3/1.0/conf/profile-domains".

A.10.4. Conformance Test 36

Test id:	/conf/profile-domains/get-queryables-uris
Requirements:	/req/profile-domains/queryables-link
Test purpose:	Check that a link to the schema resource exists
Test method:	Given: • the schema resource ("{apiURI}/{relativePathToSchema}"); When: • a request is executed for the schema resource ◦ method: "HEAD" (if "HEAD" results in a 405 response, use "GET" instead) ◦ path: "{apiURI}/{relativePathToSchema}" ◦ header: "Accept: application/schema+json" ◦ authentication, if authentication credentials are provided Then: • assert successful execution (status code is "200"); • assert that the response includes a "Link" header with "rel" set to "http://www.opengis.net/def/rel/ogc/1.0/queryables"; • store the "href" value as the Queryables URI for the filterable resource ({queryablesUri}).

Annex B: Revision History

Date	Release	Editor	Primary clauses modified	Description
2023-10-22	1.0.0-SNAPSHOT	C. Portele	all	initial version
2024-03-07	1.0.0-draft.1	C. Portele	all	draft for OAB review
2024-06-04	1.0.0-draft.2	P. Vretanos, C. Portele	all	draft for Public Comment
2025-09-22	1.0.0-draft.3	P. Vretanos, C. Portele	all	draft for OGC approval process, co-branding as OGC API - Common - Part 3

Annex C: Bibliography

Internet Assigned Numbers Authority (IANA). **Link Relation Types** [online, viewed 2023-10-14], Available at https://www.iana.org/assignments/link-relations/link-relations.xml

Open Geospatial Consortium (OGC). **OGC Link Relation Type Register** [online, viewed 2023-10-14], Available at http://www.opengis.net/def/rel

OpenAPI Initiative (OAI). **OpenAPI Specification 3.1** [online, viewed 2025-07-14]. 2024. Available at http://spec.openapis.org/oas/v3.1.1

Internet Engineering Task Force (IETF). RFC 6906: **The 'profile' Link Relation Type**. Edited by E. Wilde. 2013. Available at https://www.rfc-editor.org/rfc/rfc6906.html

Internet Engineering Task Force (IETF). RFC 8288: **Web Linking**. Edited by M. Nottingham. 2017. Available at https://www.rfc-editor.org/rfc/rfc8288.html