OGC® DOCUMENT: 20-004R1

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

Open Geospatial Consortium

OGC API - RECORDS - PART 1: CORE

STANDARD
Implementation

APPROVED

Version: 1.0
Submission Date: 2025-01-08
Approval Date: 2025-04-07
Publication Date: 2025-05-02
Editor: Panagiotis (Peter) A. Vretanos, Tom Kralidis, Angelos Tzotsos, Charles Heazel

License Agreement

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

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

Copyright notice

Note

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

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

CONTENTS

LIST OF TABLES

LIST OF RECOMMENDATIONS

ABSTRACT

The OGC API family of Standards is being developed to make it easy for anyone to provide geospatial data over the web. These Standards build upon the legacy of the OGC Web Service Standards (WMS, WFS, WCS, WPS, CSW, etc.), but define resource-centric APIs that take advantage of modern web development practices.

This document defines the OGC API — Records — Part 1: Core Standard. In this document "Records API" refers to the OGC API — Records — Part 1: Core Standard.

The content of a Record makes a resource discoverable by providing summary information (metadata) about the resource. In this context, resources are things that would be useful to a user or developer, such as features, coverages, tiles / maps, styles, assets, machine models, services, processes, widgets, notebooks etc.

An implementation of the OGC API — Records Standard provides a way to browse or search a curated collection of records known as a catalog. The Records API Standard envisions deploying a catalog using one of the following patterns:

- a collection of static files,
- a collection of records accessed via an API.

A catalog can be deployed as a static collection of records stored in web-accessible files and typically co-located with the resources each record is describing. Such a deployment is amenable to browsing using a web browser, being crawled by a search engine crawler, or traversed by automated tools.

A catalog can also be deployed as an implementation of a Web API consisting of well-known endpoints for retrieving information about the catalog, retrieving records from the catalog and searching the catalog for subsets of records that satisfy user-defined search criteria (area of interest, temporal extent, keywords, etc.).

To enable this kind of flexible deployment of a collection of records, the Records API Standard defines the following set of common components:

- a record,
- a catalog (a collection of records),
- a set of common query parameters,
- the requirements and characteristics for implementation instances of a Records search API (hereafter termed the "search API").

A record is the atomic unit of information in a catalog. It contains descriptive information (metadata) about a resource such as:

- a title,
- a human readable description of the resource,
- the nature or genre of the resource,
- keywords associated with the resource,
- information about the publisher or provider of the resource,
- links to access the resource,
- etc.

This information is generic and can be used to describe a wide variety of resources. The Records API Standard expects and encourages implementations to enrich the information content of a record to suit specific use cases, requirements, domains, or communities of interest.

A collection of records is itself described by a record that provides descriptive information about the catalog. The collection also provides access to the records of the collection either by explicitly linking to each record that is an i.e., of the collection or by providing a search endpoint that allows the collection of records to be searched.

An implementation of a search API allows collections of records to be searched using logically connected spatial, temporal and scalar predicates. This Standard uses the OGC API — Features — Part 1: Core Standard for specifying requirements of the core of the Records search API. Additional parts from the OGC API — Features suite of standards can also be implemented to enhance the capabilities of the search API. The following table lists the endpoints of the core OGC API — Records API that allow a collection of records to be searched:

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

RESOURCE	PATH	HTTP METHOD	DOCUMENT REFERENCE
Landing page	/	GET	API landing page
Conformance declaration	/conformance	GET	Conformance
Record collections	/collections	GET	Record collections
Record collection	/collections/{collectionId}	GET	Record collection
Records	/collections/{collectionId}/items	GET	Records access
Record	/collections/{collectionId}/items/{recordId}	GET	Record Core

A special use case of a catalog deployed as an implementation of a Web API is the *local resources catalog*. A local resources catalog is a catalog deployed as an implementation of a Web API that

makes local resources offered by a provider searchable using the catalog information model and accessed via the API. An example of a local resources catalog is the `/collections` endpoint of OGC API — Common — Part 2: Geospatial data or OGC API — Features — Part 1: Core. Adding catalog capabilities to this endpoint enables a client to search for collections that satisfy a specified filter expression. Any API endpoint defined in an OGC API Standard whose function is to provide descriptive metadata about other sub-resources accessible via the API can be extended to behave as a local resources catalog. The `/collections` endpoint is an examples of such an endpoint. The `/collections` endpoint provides metadata about the collections that a server offers. One can well imagine the benefits of being able to search for a specific subset of collection descriptions on a server that offers thousands of data collections.

KEYWORDS

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

ogcdoc, OGC document, OGC API, catalog, record, records, resource, metadata, discovery, CSW, API, GeoJSON, HTML, schema.org, JSON-LD

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 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.

SECURITY CONSIDERATIONS

Given the dependencies on the OGC API — Common and OGC API — Features Standard, aspects of security are discussed in those Standards.

Metadata record encryption is out of scope for this Standard. However communities who wish to implement record encryption are free to do so as needed.

V SUBMITTING ORGANIZATIONS

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

- CubeWerx Inc.
- Meteorological Service of Canada
- Open Source Geospatial Foundation
- Geonovum

VI SUBMITTERS

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

Name	Affiliation
Panagiotis (Peter) A. Vretanos (editor)	CubeWerx Inc.
Tom Kralidis (editor)	Meteorological Service of Canada
Angelos Tzotsos (editor)	Open Source Geospatial Foundation
Charles Heazel (editor)	HeazelTech
Linda van den Brink	Geonovum

VII ACKNOWLEDGEMENTS

This specification acknowledges and remembers Douglas Nebert. As part of USGS and FGDC, Doug was internationally recognized as a champion of metadata, discovery and interoperability. He was one of the editors of the OGC Catalogue Services specification and a longtime champion of spatial data infrastructure initiatives including the US data.gov efforts, GEOSS and beyond. Doug's vision and expertise will always be remembered and appreciated by the specification editors and community.

1 SCOPE

1 SCOPE

The OGC Records API Standard specifies requirements classes for a set of common components that can be assembled in various ways to deploy a collection of related descriptive information (metadata) about resources called a catalog. The atomic unit of information in a catalog is the record.

This Standard specifies the information content of a record. A record contains summary descriptive information (metadata) about a resource that a provider wishes to make discoverable. A record represents resource characteristics that can be presented for evaluation and further processing by both humans and software. Examples of resources include a data collection, a service, a process, a style, an Earth observation asset, a machine learning model, a code list and so on.

Records are organized into collections called *catalogs*. The Records API Standard describes how catalogs can be crawled or searched. Crawling a collection of records involves following embedded links from one record in a catalog to the next. Searching a collection of records involves specifying query predicates that define a subset of records.

2 CONFORMANCE

2 CONFORMANCE

2.1. Overview

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

The standardization target of the conformance classes:

- Record Core
- Record collection

is “Document model.”

The standardization target of the conformance classes:

- Record query parameters
- Profile query parameter
- Records API
- Sorting
- Filtering
- Searchable Catalog
- Searchable Catalog — Filtering
- Searchable Catalog — Sorting
- Crawlable Catalog
- Local Resources Catalog
- Local Resources Catalog — Query Parameters
- Local Resources Catalog — Filtering
- Local Resources Catalog — Sorting
- Autodiscovery

- OpenAPI 3.0

is “Web APIs.”

The standardization target of the conformance classes:

- JSON
- HTML

is “Document encoding.”

The Conformance Classes implemented in an API are advertised through the `/conformance` path on the landing page. Each Conformance Class has an associated Requirements Class. The Requirements Classes define the functional requirements which will be tested through the associated Conformance Test.

2.2. Common components

This standard also identifies ten (11) requirements classes for common components:

- Record Core
- Record collection
- Record query parameters
- Profile query parameter
- Records API
- Sorting
- Filtering
- JSON
- HTML
- OpenAPI 3.0
- Autodiscovery

These common components are not intended to be implemented independently but rather implemented as components of one or more catalog deployment types.

The *Record Core* conformance class defines the requirements for the core schema of a catalog record which is a set of properties (core properties) that can be used to make any resource

discoverable via a catalog. This core set of properties can be extended as required by specific communities of interest and/or use cases.

The *Record Collection* conformance class defines requirements for the metadata used to describe a collection of related records referred to as a *catalog* in this Standard.

The *Record query parameters* conformance class defines a minimum set of query parameters that all queryable catalogs should provide.

The *Profile query parameter* conformance class add support for the `profile` parameter which allows a client to request one or more variations of a resource representation (e.g., requesting a GeoJSON document that is a catalog record).

The *Records API* conformance class defines the requirements for an API implementation for searching catalogs based on logically connected predicates that can include spatial and/or temporal predicates. The Records API Standard is based on OGC API — Feature — Part 1: Core with extensions specific to OGC API — Records. OGC API — Features provides a common foundation for other OGC API Standards for feature access. Therefore, this conformance class should be viewed as an extension to OGC API — Features. Conformance to this class requires demonstrated conformance to the applicable Conformance Classes define in the of OGC API — Features Standard.

The *Sorting* conformance class defines the requirements to support sorting of records in a query response.

The *Filtering* conformance class defines the requirements to support record filtering using CQL2.

The *JSON* conformance class defines the requirements for a JSON representation of a catalog object and a GeoJSON (see RFC 7946) representation of a standard catalog record.

The *HTML* conformance class defines the requirements for an HTML representation of a standard catalog record.

The OpenAPI 3.0 conformance class defines the requirements for server that use an OpenAPI 3.0 document to define their API.

The Autodiscovery conformance class defines requirements that allow catalogs that conform to this Standard and are associated with a web page or web site to be automatically discovered.

2.3. Catalog deployments

Using the above-mentioned common components, this Standard identifies the following catalog requirements classes:

- Crawlable Catalog
- Searchable Catalog
    - Searchable Catalog — Sorting

    - Searchable Catalog — Filtering

- Local Resources Catalog
    - Local Resources Catalog — Query Parameters
    - Local Resources Catalog — Sorting
    - Local Resources Catalog — Filtering

Each of the top-level requirements classes, Crawlable Catalog, Searchable Catalog and Local Resources Catalog represent an implementable catalog composed of aggregations of the common components described above.

The Crawlable Catalog conformance class defines the core requirements for a catalog composed of a collection of web-accessible files. The files of a crawlable catalog may be static web pages or dynamically generated but regardless, they exist at a fixed URL and can be retrieved without the use of query parameters.

The Searchable Catalog, Searchable Catalog — Sorting, Searchable Catalog — Filtering conformance classes define the requirements for a catalog composed of a collection of records that is searchable via an API.

The Local Resources Catalog, Local Resources Catalog — Query Parameters, Local Resources Catalog — Sorting, Local Resources Catalog — Filtering conformance classes define the requirements for a local resources catalog which is a catalog composed of a list of resources offered by an OGC API deployment. The `/collections` endpoint is an example of a local resources catalog but other endpoints may exist in an OGC API deployment as well.

Table 2 — Required common components by catalog deployment type

COMMON COMPONENT	DEPLOYMENT TYPE		
	Crawlable	***Searchable***	***Local Resources catalog***
Record Core	Mandatory	Mandatory	Mandatory
Record collection	Mandatory	Mandatory	Mandatory
Record query parameters	N/A	Mandatory	Optional
Profile query parameter	N/A	Optional	Optional
Records API	N/A	Mandatory	N/A
Sorting	N/A	Optional	Optional

COMMON COMPONENT	DEPLOYMENT TYPE		
Filtering	N/A	Optional	Optional
JSON	Optional	Optional	Optional
HTML	Optional	Optional	Optional
OpenAPI 3.0	N/A	Optional	Optional

2.4. Implementations

Implementors of this Standard select one or more of the catalog deployment requirements classes they wish to implement and then implement the required common component requirements classes.

2.5. Conformance testing

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

Table 3 — Catalog Deployment Conformance class URIs

CONFORMANCE CLASS	URI
Crawlable Catalog	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/crawlable-catalog
Searchable Catalog	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog
Searchable Catalog — Filtering	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog-filtering
Searchable Catalog — Sorting	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog-sorting
Local Resources Catalog	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog
Local Resources Catalog — Query Parameters	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog-query-parameters

CONFORMANCE CLASS	URI
Local Resources Catalog — Filtering	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog-filtering
Local Resources Catalog — Sorting	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog-sorting

Table 4 — Common Component Conformance class URIs

CONFORMANCE CLASS	URI
Record Core	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core
Record Collection	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-collection
Record Core Query Parameters	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core-query-parameters
Profile Query Parameter	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/query-param-profile
Records API	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-api
Sorting	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/sorting
Filtering	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/filtering
JSON	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/json
HTML	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/html
OpenAPI 3.0	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/oas30
Autodiscovery	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/autodiscovery

3 NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

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

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee: IETF RFC 2616, *Hypertext Transfer Protocol — HTTP/1.1*. RFC Publisher (1999). https://www.rfc-editor.org/info/rfc2616.

E. Rescorla: IETF RFC 2818, *HTTP Over TLS*. RFC Publisher (2000). https://www.rfc-editor.org/info/rfc2818.

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

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

M. Duerst, M. Suignard: IETF RFC 3987, *Internationalized Resource Identifiers (IRIs)*. RFC Publisher (2005). https://www.rfc-editor.org/info/rfc3987.

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

T. Bray (ed.): IETF RFC 7159, *The JavaScript Object Notation (JSON) Data Interchange Format*. RFC Publisher (2014). https://www.rfc-editor.org/info/rfc7159.

H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, T. Schaub: IETF RFC 7946, *The GeoJSON Format*. RFC Publisher (2016). https://www.rfc-editor.org/info/rfc7946.

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

Clemens Portele, Panagiotis (Peter) A. Vretanos, Charles Heazel: OGC 17-069r4, *OGC API — Features — Part 1: Core corrigendum*. Open Geospatial Consortium (2022). http://www.opengis.net/doc/IS/ogcapi-features-1/1.0.1.

Panagiotis (Peter) A. Vretanos, Clemens Portele: OGC 19-079r2, *OGC API — Features — Part 3: Filtering*. Open Geospatial Consortium (2024). http://www.opengis.net/doc/IS/ogcapi-features-3/1.0.

Panagiotis (Peter) A. Vretanos, Clemens Portele: OGC 21-065r2, *Common Query Language (CQL2)*. Open Geospatial Consortium (2024). http://www.opengis.net/doc/IS/cql2/1.0.

Open API Initiative: OpenAPI Specification 3.0.3, https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md

WHATWG: HTML, Living Standard [online, viewed 2020-03-16]. Available at https://html.spec.whatwg.org/

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

Nottingham, M.: IETF, The Link-Template HTTP Header Field, draft-ietf-httpapi-link-template-latest

4 TERMS AND DEFINITIONS

4 TERMS AND DEFINITIONS

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

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

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

4.1. catalog; record collection

a collection of related records

4.2. conformance module; conformance test module

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

Note 1 to entry: When no ambiguity is possible, the word `test` may be omitted. i.e., conformance test module is the same as conformance module. Conformance modules may be nested in a hierarchical way.

4.3. conformance class; conformance test class

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

Note 1 to entry: When no ambiguity is possible, the word `test` may be omitted, so conformance test class may be called a conformance class.

4.4. dataset

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

4.5. deployment; deployment type

a set of components implemented to execute the runtime functions of a catalog

EXAMPLE: a crawlable catalog deployment composed of static, web-accessible files

EXAMPLE: a searchable catalog deployment composed of an API that can be used to dynamically search a collection of catalog records

4.6. distribution

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

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

4.7. executable test suite (ETS)

set of code (e.g., Java and CTL) that provides runtime tests for the assertions defined by the ATS. Test data required to do the tests are part of the ETS (OGC 08-134)

4.8. local resources endpoint

the path from which a catalog of local resources can be retrieved (e.g., `/collections`)

4.9. local resources catalog

an instance of the document or resource that is retrieved from a local resources endpoint

4.10. local resources item

an instance of an object that provides summary information about a discoverable local resource retrieved from a local resources catalog (e.g., `/collections{/collectionID}`)

4.11. OGC API resource tree

a collection of resources defined by the set of OGC API Standards that are hierarchically organized and exposed by implementations of the OGC API Standards through a root node that is common to all OGC APIs

4.12. record

atomic unit of information of a catalog that is used to provide information (i.e., metadata) about a particular resource that the publisher of that resources wishes to make discoverable

4.13. recommendation

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

4.14. **requirement**

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

4.15. **requirements class**

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

4.16. **requirements module**

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

4.17. **resource**

an asset of interest that may be described by metadata (i.e., a record) in a catalog for the purpose of discovery and access (i.e., binding)

4.18. **searchable catalog endpoint**

an endpoint in the OGC API resource tree that is designed to support catalog queries

Note 1 to entry: For searchable catalogs, the searchable catalog endpoint is at `/items`. For local resources catalogs, an example of a searchable catalog endpoint might be `/collections` or `/processes`.

4.19. standardization target

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

Note 1 to entry: The standardization target is the entity which may receive a certificate of conformance for a requirements class.

5 CONVENTIONS

5 CONVENTIONS

The following conventions will be used in this document. Examples of conventions are symbols, abbreviations, use of XML schema, use of JSON schema, or special notes regarding how to read the document.

5.1. Identifiers

The normative provisions in this Standard are denoted by the URI `http://www.opengis.net/spec/ogcapi-records-1/1.0`.

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

5.2. Use of HTTPS

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

5.3. Link relations

To express relationships between resources, RFC 8288 (Web Linking) is used.

The following IANA registered link relation types are used in this document.

- **alternate**: Refers to a substitute for this context.
- **collection**: The target IRI points to a resource which represents the collection resource for the context IRI.
- **describes**: Refers to a resource that is described by this context (i.e., the record).
- **icon**: Refers to an icon representing this context.
- **item**: Refers to a resource that is a member of the collection represented by this context.
- **license**: Refers to a license associated with this context.

- **next**: Indicates that the link's context is a part of a series, and that the next in the series is the link target.
- **prev**, **previous**: Indicates that the link's context is a part of a series, and that the previous in the series is the link target.
- **self**: Conveys an identifier for this context.
- **service-desc**: Identifies service description for this context that is primarily intended for consumption by machines. API definitions are considered service descriptions.
- **service-doc**: Identifies service documentation for this context that is primarily intended for human consumption.

In addition, the following link relation types are used for which no applicable registered link relation type could be identified.

- **catalog**: Refers to a resource that is a catalog associated with this context.
- **child**: Indicates that the link's context is part of a hierarchy, and that a child resource in the hierarchy is the link target.
- **conformance**: Refers to a resource that identifies the specifications that this context conforms to.
- **items**: Refers to a resource that is comprised of members of the collection represented by this context.
- **parent**: Indicates that the link's context is part of a hierarchy, and that the parent resource in the hierarchy is the link target.
- **root**: Indicates that the link's context is part of a hierarchy, and that the root resource of the hierarchy is the link target.
- **`http://www.opengis.net/def/rel/ogc/1.0/sortables`**: Refers to a resource, for this context, that provides a list of properties that may be used to create a sortby expression.
- **http://www.opengis.net/def/rel/ogc/1.0/queryables**: Refers to a resource, for this context, that provides a list of properties that may be used to create a filter expression.
- **http://www.opengis.net/def/rel/ogc/1.0/ogc-catalog**: Refers to a resource, for this context, that points to an associated catalog.

Each resource representation includes an array of links. Implementations are free to add additional links for all resources provided by the API. For example, an enclosure link could reference a bulk download of a collection. Or a related link on a record could reference a related record.

5.4. Profile URIs

The following profile URIs are defined and used in this document.

- **http://www.opengis.net/def/profile/OGC/0/ogc-catalog**: Refers to a profile of a base resource type (e.g., `application/json`) that is a catalog.
- **http://www.opengis.net/def/profile/OGC/0/ogc-record**: Refers to a profile of a base resource type (e.g., `application/geo+json`) that is a record of a catalg.

5.5. Examples

Most of the examples provided in this Standard are encoded in JSON. JSON was chosen because it is widely understood by implementers and easy to include in a text document. This convention should NOT be interpreted as a requirement that JSON must be used. Implementers are free to use any format they desire as long as there is a Conformance Class for that format and the deployed API advertises its support for the associated Conformance Class.

5.6. Schemas

OpenAPI 3.0 Schema objects are used throughout this Standard to define the structure of resources. These schema are typically represented using YAML encoding. This convention is for the ease of the user. It does not prohibit the use of another schema language or encoding. Nor does it indicate that OpenAPI 3.0 Schema objects are required. Implementations should use a schema language and encoding appropriate for the format of the resource. Note that for property values in JSON for which `null` is not explicitly supported/required, server implementations are recommended to drop the property (as opposed to specifying the property with a value of `null`).

6 OVERVIEW

6 OVERVIEW

6.1. Role of catalogs

Historically, catalogs have been used as large repositories of formal metadata such as ISO 19115, ISO 19119, FGDC, etc. More recently, catalogs are migrating to a more distributed model where the metadata is closely coupled with the resource that it describes. A current example is SpatioTemporal Asset Catalogs (STAC) which relies on well-defined common components that can be integrated in different ways.

Although OGC API — Records is built upon the legacy of the OGC Catalog Service Web (CSW) Standard, which was primarily focused on accessing a large repository of metadata documents, this Standard also supports other catalog deployment patterns such as a catalog composed of a distributed set of static records.

In the modern scenario, the catalog records can be distributed and linked together so that they can be crawled, or they can be harvested into a service endpoint to make them searchable. In addition, a searchable catalog can harvest not only a metadata document describing the resource but, in some cases, the resource itself (e.g., harvesting the collections object from an OGC API). The implications of this are subtle yet important. In the traditional cataloging approach (e.g., based on CSW) a client could find the record and the record might include binding information to access the resource. Using the new approach, a client interacts more intimately with the catalog and the catalog becomes integral to accessing the resource. An example is the cloud native ecosystem: HTTP range requests to Cloud Optimized GeoTIFF (COG) data cannot be used properly unless there is a STAC i.e., telling the client in what ranges the bands exists. Therefore the catalog becomes an integral component (discovery, evaluation) in the process of accessing a resource.

6.2. Records and catalogs

The atomic unit of information in a catalog is the *record*.

A *catalog* is a collection of related *records*. The terms **collection of records** and **catalog** are used interchangeably in this Standard.

A record provides a summary description (metadata) of a resource that a provider of the resource wishes to make discoverable. The record schema defined in this Standard includes a small number of properties that can be used to provide a summary description of any resource. It is anticipated, and encouraged, that this list of properties be extended to enhance the information content of a record as required by specific communities of interest and/or use cases.

6.3. Common components

This Standard defines the following common components that may be used to deploy a catalog.

1. The *core* schema of a record.
2. The definition of a *collection* resource that groups and describes a set of related records (i.e., a catalog).
3. The definition of a core set of query parameters for searchable catalogs.
4. An API that allows a catalog to be searched using logically connected predicates that can include spatial and/or temporal predicates.

Using these common components a catalog can be deployed as:

- a set of web-accessible records that can be crawled by a search engine crawler, using (for example) a web browser, or by using automated tools; or
- a searchable catalog with records accessible through an API endpoint.

A special use case of the *searchable catalog* is the ***local resources catalog***. The OGC API resource tree has several endpoints that provide lists of resources. Examples of such endpoints include the `/collections` and the `/processes` endpoints. One can readily imagine a large deployment of APIs implementing OGC API building blocks with thousands of collections where the ability to search would enhance the client's interaction with the server. Using the common components defined in this Standard, these endpoints can be extended to support catalog-like searching using filter expressions that may also include spatial and/or temporal predicates. Implementations of OGC API endpoints extended in this way behave like a catalog of local resources.

6.4. Record Schema

The core set of record properties that may be used to describe a resource include the *id*, *type*, *title*, *description* and *geometry*. A complete definition of a record can be found in clause Core properties.

The choice of which properties to include in the set of core properties was primarily informed by the following record models:

- The core catalog schema from the OGC® Catalogue Services 3.0 suite of standards; and
- The Data Catalog Vocabulary (DCAT) version 2 and the GeoDCAT-AP version 3.0.0 specifications.

It is expected that the information necessary to populate this core set of properties is readily available for any resource that is being made discoverable by a record.

The expectation is that this *core* set of properties will be extended to describe specific resource types (e.g., datasets, dataset series, Earth observation products, machine models, services, etc.). The core can also be extended by information communities wishing to enrich the information content of the record to suit their needs. Extending the information content of a record to suit specific needs is allowed, expected, and encouraged by this Specification.

Although this Standard does not mandate any particular encoding for a record or a catalog, it does define conformance classes for two encodings:

- A JSON, and
- An HTML encoding.

Other encodings are allowed but are not described in this Standard.

Accessing a catalog deployed as static files is described in the Crawlable records deployment clause.

Accessing a catalog through an implementations of the Records API is described in the Records access clause.

6.5. Record collection (catalog)

A record collection or catalog is an object that groups and describes a set of related records. The catalog is the primary access point from which a deployed set of records can be accessed. Having found a catalog a client can, by following the appropriate hypermedia controls contained therein, navigate to the records of the collection.

Depending on the deployment pattern, the catalog may provide a link to each individual record of the collection or a link to a search access point for retrieving subsets of records.

The core set of properties that may be used to describe a catalog include *id*, *type*, *title*, *description*, *extent* and *links*. A complete definition of a catalog can be found in clause Record collection. It should be noted that this list of properties is very similar to the list of properties defined for a catalog record. This is intentional since the catalog can be considered a *record* that describes a related set of records.

It is anticipated that this set of properties will be extended to enrich the information content of the collection metadata to suit specific needs.

6.6. Records API

6.6.1. Overview

An implementation of the Records API Standard enables retrieving a subset of records from a catalog using a logically connected set of predicates that may include spatial and/or temporal predicates.

The Records API extends the OGC API — Features — Part 1: Core Standard to:

1. Provide API patterns and encodings to facilitate further lowering the barrier to finding the existence of spatial resources on the Web; and
2. Provide functionality comparable to that of the OGC Catalogue Service (CSW) Standard so that a facade can be created over legacy services thus allowing them to participate in the new OGC API Standards ecosystem.

To facilitate access to records in a catalog, the OGC API — Feature — Part 1: Core Standard has been:

- Extended with additional parameters at the `/collections/{collectionId}/items` endpoint, and
- Constrained to a single information model (i.e., the record).

Table 5 summarizes the access paths and relation types defined in this Standard.

Table 5 — Records API Paths

PATH TEMPLATE	RELATION	RESOURCE
	Common	
/		Landing page
/api	`service-desc` or `service-doc`	API Description (optional)
/conformance	`conformance`	Conformance Classes
/collections	`data`	Metadata describing the catalogs available from this API implementation instance.
/collections/{catalogId}	`collection`	Metadata describing a catalog which has the unique identifier `{catalogId}`

PATH TEMPLATE	RELATION	RESOURCE
	Records	
/collections/{catalogId}/items	`items`	Search results based on querying the service for records satisfying 0..n query parameters.
/collections/{catalogId}/items/{recordId}	`item`	Record of metadata which has the unique identifier `{recordId}`.

Where:

- `{catalogId}` = An identifier for a specific catalog; and
- `{recordId}` = An identifier for a specific record within a collection.

6.6.2. API Behaviour Model

The Records API Standard is designed to be compatible but not conformant with the OGC Catalogue Service for the Web (CSW) Standard. This allows OGC API — Records implementations and CSW implementations to co-exist in a single processing environment.

The OGC Catalogue Service Standard version 3 provides an abstract core model of metadata (data about data) describing a number of different information types (datasets, services, styles, processes, etc.) on which the classic operations (GetCapabilities, DescribeRecord, GetRecords, and GetRecordById) can be explained naturally. This model consists of 1 .. n catalog collections residing in a CSW backend repository. The model holds service metadata describing service qualities (identification, contact, operations, filtering capabilities, etc.). In principle, a catalog may provide discovery services to any number of metadata repositories. The core catalog model is based on an extension of Dublin Core (CSW Record). Application profiles can be developed to target specific metadata information models (such as ISO 19115/19139, etc.).

Discussion has shown that the Records API model also assumes underlying service and object descriptions, so a convergence seems possible. In any case, it will be advantageous to have a similar “mental model” of the server store organization on hand to explain the various functionalities introduced below.

6.6.3. Search

This Standard defines three levels of search capability of increasing complexity and capability.

The first or core level of search capability is based on OGC API — Features and thus supports:

- bounding box searches,
- time instant or time period searches, and

- equality predicates (i.e., *property=value*).

The Records API Standard extends these core search capabilities to include:

- keyword searches,
- searches based on the type of resource,
- searches based on one or more record identifiers, and
- searches based on one or more external identifiers of a resource.

The second level of search capability extends the search API so that it is compatible with the OGC OpenSearch Geo and Time Extensions (OpenSearch Geo). OpenSearch Geo gives the user more control over the kinds of geometries, beyond a bounding box, that can be used to define an area of interest. OGC API — Records — Part 2: OpenSearch will define the requirements for a catalog that supports OpenSearch.

The third level of search capability, defined by the *Filtering* requirements class, supports complex filter expressions using a rich set of logically connected query predicates.

6.6.4. Dependencies

The search API requirements and conformance classes defined in this Standard for accessing records from a searchable catalog is an extension of the OGC API — Features — Part 1: Core Standard. An implementation of a searchable catalog API must first satisfy the appropriate Requirements Classes from OGC API — Features — Part 1: Core. [req-mappings], identifies these requirements.

Table 6 — Required OGC API — Features — Part 1: Core Requirements Classes for Records Access

API — Features Requirements Classes
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/fc-op
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/fc-bbox-definition
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/fc-bbox-response
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/fc-limit-definition
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/fc-limit-response
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/fc-time-definition
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/fc-time-response

http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/query-param-invalid
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/query-param-unknown
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/fc-response
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/fc-links/req/core/fc-rel-type
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/fc-timeStamp
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/fc-numberMatches
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/fc-numberReturned,

Table 7 — Required OGC API — Features — Part 1: Core Requirements Classes for Record Access

API — Features Requirements Classes
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/f-op
http://www.opengis.net/spec/ogcapi_features-1/1.0/req/core/f-success

7

COMMON COMPONENTS

7 COMMON COMPONENTS

7.1. Overview

This clause specifies requirements classes that define common components that can be used to define various catalog deployments.

7.2. Requirements Class "Record Core" (Common Component)

7.2.1. Overview

REQUIREMENTS CLASS 1	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core`
TARGET TYPE	Document model
CONFORMANCE CLASS	Conformance class A.1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core`
NORMATIVE STATEMENTS	Requirement 1: `/req/record-core/mandatory-properties-record` Requirement 2: `/req/record-core/time-instant` Requirement 3: `/req/record-core/time-interval` Requirement 4: `/req/record-core/time-instant-interval` Requirement 5: `/req/record-core/time-zone` Requirement 6: `/req/record-core/contact` Requirement 7: `/req/record-core/license` Requirement 8: `/req/record-core/links` Requirement 9: `/req/record-core/templated-link` Requirement 10: `/req/record-core/templated-link-header` Requirement 92: `/req/record-core/default-mediatype`

The `Record` Requirements Class defines the requirements for a catalog record which is the atomic unit of information of a catalog.

It is expected that the information necessary to populate the fields/properties of a catalog record is readily available for any resource that is being registered in the catalog.

The expectation is that the schema of a record will be extended to describe specific resource types (e.g., datasets, services, etc.) as well as also extended by information communities wishing to enrich the information content of the record to suit their needs.

Although this Standard does not mandate any particular encoding for a record, this document does define conformance classes for two encodings:

- a GeoJSON record encoding, and
- an HTML encoding.

Other encodings are allowed but are not described in this document.

7.2.2. Core properties

Table 8 and Table 9 define the set of properties, called the *core properties*, that may be included in a record.

Table 8 — Core properties related to the catalog record

PROPERTY	REQUIREMENT	DESCRIPTION	GEOJSON KEY
id	**required**	A unique record identifier assigned by the server.	`id`
created	optional	The date this record was created in the server.	`properties.created`
updated	optional	The most recent date on which the record was changed.	`properties.updated`
conformsTo	optional	The extensions/conformance classes used in this record.	`conformsTo`
language	optional	The language used for textual values (i.e., titles, descriptions, etc.) of this record.	`properties.language`
languages	optional	The list of other languages in which this record is available.	`properties.languages`
links	optional	A list of links related to this record.	`links`
linkTemplates	optional	A list of link templates related to this record.	`linkTemplates`

Table 9 — Core properties related to the resource

PROPERTY	REQUIREMENT	DESCRIPTION	GEOJSON KEY
type	optional	The nature or genre of the resource described by this record.	`properties.type`
title	optional	A human-readable name given to the resource described by this record.	`properties.title`
description	optional	A free-text description of the resource described by this record.	`properties.description`
geometry	optional	A spatial extent associated with the resource described by this record. Can be `null` if there is no associated spatial extent.	`geometry`
time	optional	A temporal extent associated with the resource described by this record. Can be `null` if there is no associated temporal extent.	`time`
keywords	optional	A list of free-form keywords or tags associated with the resource described by this record.	`properties.keywords`
themes	optional	A knowledge organization system used to classify the resource described by this resource.	`properties.themes`
resource Languages	optional	The list of languages in which the resource described by this record can be retrieved.	`properties.resourceLanguages`
externalIds	optional	One or more identifiers, assigned by an external entity, for the resource described by this record.	`properties.externalIds`
formats	optional	A list of available distributions for the resource described by this record.	`properties.formats`
contacts	optional	A list of contacts qualified by their role(s) in association to the record or the resource described by this record.	`properties.contacts`
license	optional	The legal provisions under which the resource described by this record is made available.	`properties.license`
rights	optional	A statement that concerns all rights not addressed by the license such as a copyright statement.	`properties.rights`

REQUIREMENT 1	
IDENTIFIER	`/req/record-core/mandatory-properties-record`

REQUIREMENT 1	
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core`
A	Each record in a response SHALL contain all the mandatory properties listed in Table 8.
B	The identifier of a record (`id`) cannot be NULL or the empty string.

PERMISSION 1	
IDENTIFIER	`/per/record-core/common-mark`
A	For any `description` member in a record, CommonMark MAY be used for a rich text representation.

PERMISSION 2	
IDENTIFIER	`/per/record-core/additional-properties-record`
A	Each record in a response MAY contain zero or more of the optional properties listed in Table 8 and Table 9.
B	Each record in a response MAY contain any number of additional properties not listed in Table 8 and Table 9. The meaning of these additional properties is not specified in this document.

7.2.3. Record extensions

In Table 8 and Table 9 this Standard defines a set of core properties for describing discoverable resources. It is anticipated that this set of core properties will be extended to suit specific use cases or requirements. For example, a catalog record may be extended to include additional properties from the GeoDCAT vocabulary. The fact that a catalog record has been extended in this way can be advertised using the `conformsTo` property. The `conformsTo` property is a list of identifiers that indicate each extension used in a record. This Standard does not define the specific identifiers that are used to identify specific extensions; it simply provides a place where these identifiers can be listed.

RECOMMENDATION 1	
IDENTIFIER	`/rec/record-core/extensions`
A	If the core record is extended with additional properties, the requirements of the extension SHOULD be formally published.

RECOMMENDATION 1	
B	The formal publication SHOULD define an identifier that can be included in the `conformsTo` list to allow clients to identify that the record has been extended according to the requirements defined in the publication.
C	The identifier SHOULD be an active identifier (e.g., an URL) that resolves to this publication or eventually leads to it through intermediate resources.
D	The formal publication SHOULD include requirements about any formal vocabularies that SHOULD be used to populate core and extension record properties.

In the case where all the records of a catalog use the same set of extensions, and to prevent unnecessary duplication, there is also a `conformsTo` property at the catalog level that may be used to declare which extensions are used in all records of the catalog.

NOTE: Extensions listed at the catalog and record levels are cumulative.

7.2.4. Type

In order to facilitate the ability to discover resources of a particular type(s), this Standard makes the following recommendations:

RECOMMENDATION 2	
IDENTIFIER	`/rec/record-core/type`
A	A record SHOULD include the `type` property to indicate the nature or genre of the resource that the record is describing.
B	The list of codes that can occur in records of a catalog for the property `type` SHOULD use a formal, controlled vocabulary defined by an organization (e.g., ISO), or a community of interest/domain (e.g., WMO).
C	The list of codes that can occur in records of a catalog for the property `type` SHOULD be listed as enum values in the API definition of the record schema.

PERMISSION 3	
IDENTIFIER	`/per/record-core/type`
A	The list of codes that can occur in records of a catalog for the property `type` MAY use vocabularies found in the OGC Definitions server. A subset of common examples can be found in Annex C.

In addition, links to external resources describing the record type vocabulary used to populate the `type` property are recommended, as described in Associations and Links.

7.2.5. Title and description

In order to facilitate the discovery of resources utilizing keyword or free-text searches, this Standard makes the following recommendation:

RECOMMENDATION 3	
IDENTIFIER	`/rec/record-core/record-title`
A	A record SHOULD include the `title` property to provide a human-readable name for the resource that the record describes and also to provide a target of possible text searches of the catalog.

RECOMMENDATION 4	
IDENTIFIER	`/rec/record-core/record-description`
A	A record SHOULD include the `description` property to provide a free-text description of the resource that the record describes.

7.2.6. Spatial information

The spatial characteristic of the resource described by a record is specified using the optional `geometry` property.

PERMISSION 4	
IDENTIFIER	`/per/record-core/geometry`
A	Specific communities of interest and/or use cases MAY alter this obligation and make the `geometry` property mandatory.

For example, the JSON requirements class makes the `geometry` member mandatory because the record encoding (i.e., GeoJSON) requires that the `geometry` member be mandatory.

7.2.7. Temporal information

7.2.7.1. Overview

Many records have a geometry property that provides information about the primary spatial characteristics of the resource described by the record. Records often also have temporal information. In most cases time is either an instant (e.g., an event) or an interval (e.g., an activity or a temporal validity). In OGC API Records the temporal property is reflected in the datetime parameter which supports temporal filtering.

This Standard added support for the most common temporal case: Associating a resource with a single temporal instant or interval in the Gregorian calendar. More complex cases and other temporal coordinate reference systems are out-of-scope for this Standard and might be specified in future extensions.

Records can have multiple properties that describe the temporal characteristics of the resource(s) that the record describes.

- In many records all temporal properties are instants (represented by a date or a timestamp) and intervals are described using two temporal instants, one for the start and one for the end.
- Multiple temporal properties are sometimes used to describe different temporal characteristics of the resource(s) the record describes. For example, the time instant or interval when the information in the record is valid (sometimes called “valid time”) and the time when the record was recorded in the catalog (sometimes called “transaction time”). Another example is the Observations & Measurements standard, where an observation has multiple temporal properties including “phenomenon time,” “result time,” and “valid time.”

This Standard does not place any constraints on the number of additional temporal properties that may be added to a record to characterize temporal aspects of the resource being described by a record. This can make it difficult for a naive client to recognize and utilize temporal information stored in the record. For this reason, this Standard includes the `time` property in a record.

RECOMMENDATION 5	
IDENTIFIER	`/rec/record-core/time`
A	A record SHOULD include a `time` property that indicates the temporal aspect of the resource that the record describes.
B	If temporal information about the resource is not available then the `time` property SHOULD, even though it is optional, still be included in the record and its value SHOULD be set to `null`.

This `time` property is set by the publisher of the record and describes characteristic temporal information (an instant or an interval) associated with the resource described by a record. This `time` property can thus be used by naive clients without the need to inspect or understand the schema of an extended record. If included in the record, the value of the `time` property is either `null` (no temporal information) or an object encoding a time instant or interval.

Table 10 — Properties of the "time" object

PROPERTY	TYPE	DESCRIPTION
`date`	string	An instant with the granularity of a date. See below for more details about instants.
`timestamp`	string	An instant with the granularity of a timestamp. See below for more details about instants.
`interval`	[string]	An interval, described by an array of the two instants (start and end). See below for more details about intervals.

If both values interest, including both an instant and an interval is valid. In this case, clients should use the interval property and may use the date or timestamp property to determine the temporal characteristics of the resource being described by the record.

The "time" object may be extended with additional properties. Clients processing a "time" object must be prepared to parse additional properties. Clients should ignore properties that they do not understand. For example, in cases where the "time" property neither includes an "instant" nor "interval", a client may process the record as a record without temporal information.

NOTE: The data publisher decides how additional temporal properties are encoded in a record. The schema for the `time` property does not imply a recommendation that temporal properties reuse the same schema. For example, it is expected that a date-valued record properties will in most cases be represented as a string with an RFC 3339 date value.

7.2.7.2. Instants

An instant is a value that conforms to RFC 3339 (Date and Time on the Internet: Timestamps) and is consistent with one of the following production rules of the ISO 8601 profile specified in the RFC:

- `full-date` (e.g., `"1969-07-20"`)
- `date-time` (e.g., `"1969-07-20T20:17:40Z"`)

Conceptually, an instant is a "temporal entity with zero extent or duration" [Time Ontology in OWL]. In practice, the temporal position of an instant is described using data types where each value has some duration or granularity. The value should be described with a granularity that is sufficient for the intended use of the data.

In the case of a timestamp the granularity is a second or smaller. All timestamps must be in the time zone UTC ("Z").

In the case of a date the granularity is a day and dates as instants will be used when a granularity of a day independent of its timezone is sufficient for the intended use of the data. If that is not the case and, for example, the timezone is important for the intended use, the temporal information should be provided as an interval with start and end timestamps.

NOTE 1: This Standard only provides guidance as to how record data is represented in JSON. It is out-of-scope for this Standard to provide guidance how to cast record data to other data types. The Common Query Language (CQL2) standard uses the same model of instants and intervals as used here and includes additional guidance on how to compare values.

Example 1 — A date

```
"time" : {
    "date": "1969-07-20"
}
```

Example 2 — A timestamp

```
"time" : {
    "timestamp": "1969-07-20T20:17:40Z"
}
```

Dates and timestamps are the initial range of instant values. The range may be extended in the future to support additional use cases. Clients processing values of `time` must be prepared to receive other values. Clients may ignore values that they do not understand.

REQUIREMENT 2	
IDENTIFIER	`/req/record-core/time-instant`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core`
A	If the `time` object in a record includes a `date` member, the value SHALL conform to RFC 3339 (Date and Time on the Internet: Timestamps) and match the production rule `full-date`.
B	If the `time` object in a record includes a `timestamp` member, the value SHALL conform to RFC 3339 (Date and Time on the Internet: Timestamps) and match the production rule `date-time`.

NOTE 2: `date` and `timestamp` are different datatypes and so comparing them will require a cast. This Standard does not prescribe how types are cast. The evaluation of comparison expressions that involve `date` and `timestamp` values will, therefore, be system dependent.

PERMISSION 5	
IDENTIFIER	`/per/record-core/temporal-comparison`

PERMISSION 5	
A	`date` and `timestamp` are different datatypes and when attempting to compare them, a server MAY either return an error or it MAY cast the value(s) to compatible data types for comparison.

7.2.7.3. Intervals

An interval is described by start and end instants. Both start and end instants are included in the interval, i.e., the interval is closed.

The end of unbounded intervals is represented by a double-dot string (`..`) for the start/end. This follows the convention of ISO 8601-2 for an open start or end.

Example 1 — An interval with dates

```
"time" : {
    "interval": [
        "1969-07-16",
        "1969-07-24"
    ]
}
```

Example 2 — An interval with timestamps

```
"time" : {
    "interval": [
        "1969-07-16T05:32:00Z",
        "1969-07-24T16:50:35Z"
    ]
}
```

Example 3 — An half-bounded interval

```
"time" : {
    "interval": [
        "2014-04-24T10:50:18Z",
        ".."
    ]
}
```

The options described above are the initial range of interval values — the granularity is either days or (sub-)seconds and interval ends may be unbounded. The value range may be extended in the future to support additional use cases. Clients processing values of `time` must be prepared to receive other values. Clients may ignore values that they do not understand.

REQUIREMENT 3	
IDENTIFIER	`/req/record-core/time-interval`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core`

REQUIREMENT 3	
A	If the `time` object in a record includes an `interval` member, each array i.e., SHALL be a string that is a double-dot ("..") or conforms to RFC 3339 (Date and Time on the Internet: Timestamps) and match one of the following production rules: `full-date` (a date) or `date-time` (a timestamp).
B	If the start is a date, the end SHALL be a date, too, or "..".
C	If the start is a timestamp, the end SHALL be a timestamp, too, or "..".

7.2.7.4. Instants and intervals

REQUIREMENT 4	
IDENTIFIER	`/req/record-core/time-instant-interval`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core`
A	If the `time` object in a record includes both a `date` and a `timestamp` member, the `full-date` parts SHALL be identical.
B	If the `time` object in a record includes both a `timestamp` and an `interval` member with start/end dates, the interval SHALL contain the date of the timestamp.
C	If the `time` object in a record includes both a `timestamp` and an `interval` member with start/end timestamps, the interval SHALL contain the timestamp.
D	If the `time` object in a record includes both a `date` and an `interval` member with start/end dates, the interval SHALL contain the date.
E	If the `time` object in a record includes both a `date` and an `interval` member with start/end timestamps, the interval SHALL include timestamps on the date.

7.2.7.5. Time zones

REQUIREMENT 5	
IDENTIFIER	`/req/record-core/time-zone`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core`
A	Timestamps in the `time` member in a record SHALL use UTC ("Z") as the time zone.

7.2.8. Keywords and Themes

A record allows for one `keywords` and one to many `themes` properties. Keywords are free form terms or tags associated with the resource(s) that the record describes. Themes are concepts associated with the resource(s) that a record describes taken from one or more formal knowledge organization systems or schemes. The list of knowledge organization systems or schemes used in a searchable or local resources catalog can be determined by inspecting the server's OpenAPI document or, if the server implements CQL2, by exposing a queryable (e.g., named `scheme`) and enumerating the list of schemes used in the queryable's schema definition.

RECOMMENDATION 6	
IDENTIFIER	`/rec/record-core/keywords-themes`
A	Implementations SHOULD use the `keywords` property to provide free-form terms or tags associated with the resource(s) described by the current record.
B	Implementations SHOULD use the `themes` property to enumerate 1..n concepts, and their respective knowledge organization system/controlled vocabulary, associated with the resource(s) described by the current record.

The following provides an example of using keywords for free form terms (e.g., tags) as well as themes to articulate terms from formal vocabularies or classification systems.

```
{
  .
  .
  .
  "keywords": [
    "total",
    "ozone",
    "level 1.0",
    "column",
    "dobson",
    "brewer",
    "saoz"
  ],
  "themes": [
    {
      "concepts": [
        {
          "id": "dobson",
          "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
dobson"
        },
        {
          "id": "brewer",
          "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
brewer"
        },
        {
          "id": "vassey",
```

```
          "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
vassey"
        },
        {
          "id": "pion",
          "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_pion"
        },
        {
          "id": "microtops",
          "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
microtops"
        },
        {
          "id": "spectral",
          "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
spectral"
        },
        {
          "id": "hoelper",
          "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
hoelper"
        },
        {
          "id": "saoz",
          "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_saoz"
        },
        {
          "id": "filter",
          "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
filter"
        }
      ],
      "scheme": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode"
    },
    {
      "concepts": [
        {
          "id": "atmosphericComposition",
          "url": "https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_
CategoryCode_atmosphericComposition"
        },
        {
          "id": "pollution",
          "url": "https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_
CategoryCode_pollution"
        },
        {
          "id": "observationPlatform",
          "url": "https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_
CategoryCode_observationPlatform"
        },
        {
          "id": "rocketSounding",
          "url": "https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_
CategoryCode_rocketSounding"
        }
      ],
      "scheme": "https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_
CategoryCode"
    }
  ]
  .
  .
```

```
  .
}
```

Listing 1 — Keywords and themes Example

7.2.9. Formats

The `formats` property indicates the list of available distribution formats for the resource that a record describes. These can include both physical and digital distribution formats.

The following example illustrates a resources that is a available is several formats.

```
"formats": [
  { "name": "TK50" },
  { "name": "GRIB", "mediaType": "application/x-grib" },
  { "mediaType": "application/zip"}
]
```

Listing 2 — Formats Example

RECOMMENDATION 7	
IDENTIFIER	`/rec/record-core/format`
A	The value of the `mediaType` property should be taken from the list of IANA media types.

7.2.10. Language handling

7.2.10.1. Overview

A catalog record can include language information about:

- The resource that the record describes;
- The language used for text values in the current record representation; and
- Alternate languages in which the current record can be represented.

7.2.10.2. Language of the record

RECOMMENDATION 8	
IDENTIFIER	`/rec/record-core/langs`

RECOMMENDATION 8	
A	The language used to express text values in a record SHOULD be specified using the `language` property of the record.
B	The list of other languages in which a record may be retrieved SHOULD be specified in the record using the `languages` property.
C	If one or more language-specific links are included in the links section of a record (path: `links`) pointing to alternate language representations of this record, the language used as the `hreflang` value SHOULD be one of the languages listed as the value of the `languages` property of the record.

The link object and the templated link object are based on RFC 8288 (Web Linking) which includes a `hreflang` attribute that can be used to state the language of a referenced resource. This can be used to include links to the same record in different languages.

A server that wants to use language-specific links will have to support a mechanism to mint language-specific URIs to express links to the same record in other languages or to express links to the resource described by record in multiple languages. This document does not mandate any approach how such a capability is supported by a server.

NOTE 1: Two common approaches are:

- An additional path element for each language-specific encoding of a record or resource (this can be expressed, for example, using a language-specific path element like …`/en-ca/`…); and
- An additional query parameter (for example, "language" or "l") that overrides the Accept-Language header of the HTTP request.

NOTE 2: There is no provision in the record schema to allow a response containing text fields simultaneously presented in multiple languages (e.g., multiple titles in different languages). To obtain a record in multiple languages would require multiple requests to the server each negotiating to a desired language.

7.2.10.3. Language of the resource

RECOMMENDATION 9	
IDENTIFIER	`/rec/record-core/resource-langs`
A	If there are one or more languages associated with the resource that a catalog record describes, those languages SHOULD be listed in the record using the `resourceLanguages` property.

7.2.11. Contacts

REQUIREMENT 6	
IDENTIFIER	`/req/record-core/contact`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core`
A	If the `contacts` property is present and the `logo` property is included in the contact object then the link relation SHALL be `icon` and the media type SHALL be an `image` media type.
B	If the `contacts` property is included and the `links` property is included in the contact object then the link relation `about` SHOULD be used for each listed link and the media type SHALL indicate the content type of the link (e.g., `text/html` for a company's web page versus `text/vcard` for a virtual contact file).

RECOMMENDATION 10	
IDENTIFIER	`/rec/record-core/contact`
A	A contact `country` property SHOULD use ISO 3166-1 notation.

7.2.12. License

A record may have one `license` property. If present, the value of this property should be a code, convenient for filtering records.

REQUIREMENT 7	
IDENTIFIER	`/req/record-core/license`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core`
A	If the resource is being made available under a common license then a SPDX license identifier SHALL be used as the value of the `license` property.
B	If the resource is being made available under: • more than one common license with a SPDX license identifier, • one or more licenses that have not been assigned a SPDX license identifier, • one or more custom licenses
STATEMENT	then the value of the `license` property SHALL be the value `other` and one or more links (relation: `license`) SHALL be included in the `links` section of the record pointing to the file(s) that contain the text of the licenses.

NOTE: There is also the case of a resource that is private or unpublished and is thus unlicensed. In this case do not register such a resource in the catalog in the first place since there is no point in making such a resource discoverable.

7.2.13. Associations and Links

A record must contain a `links` section and may contain a `linkTemplates` section. These sections contain navigation links and/or association links.

Navigation links are meant to encode relationships between catalog entities (i.e., collections and records). Navigation links in the `links` section can include links to the collection (relation: `collection`) of which a record is a member, alternative representations (relation: `alternate`) of the record, and — in the context of an API — navigation links to previous (relation: `prev` or `previous`) or next (relation: `next`) records in a chain of records. Links in the `links` section may also be used to organize records in a hierarchy (relation: `root`, relation: `parent`, relation: `child`).

REQUIREMENT 8	
IDENTIFIER	`/req/record-core/links`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core`
A	Each record SHALL, if it is a member of a catalog, include a link (relation: `collection`) pointing to the catalog of which this record is a member.
B	Only a single link (relation: `collection`) SHALL be included in a record. This implies that a record can only be a member of a single catalog.

RECOMMENDATION 11	
IDENTIFIER	`/rec/record-core/links`
A	Each record SHOULD include one or more links (relation: `describes`) pointing to the resource(s) that this record describes.
B	Each record SHOULD include a link (relation: `self`) to itself.
C	Each record SHOULD, if they exist, include typed links (relation: `alternate`) to each alternative representation of this record.

NOTE 1: The term “typed link” means a link that specifies a value for the `type` attribute.

Links in the `links` section may also encode canonical URIs for record properties such as `type` and `license`.

RECOMMENDATION 12

IDENTIFIER	`/rec/record-core/links-type`
A	If there is a cannonical URI for the value of the `type` property of a record, a link (relation: `type`) to that canonical URI SHOULD be included in the `links` section of the record.

RECOMMENDATION 13

IDENTIFIER	`/rec/record-core/links_iana`
A	Links relations for links in the `links` or `linkTermplates` sections SHOULD be taked from the official IANA Link Relation Types registry.

Association links point to the resource(s) being described by a record. Such links allow binding to a resource once it has been discovered by searching a catalog. Depending on how the target resource of a record is represented and how it may be accessed, there may be one or more association links. For example, if the target resource is an Earth observation imagery file, links may point to the individually accessible bands of the image file.

Association links are also meant to point to other records related to the containing record. For example, consider a record that describes a set of vector features and another record that describes a rendering style for that feature set. These two records may, in their respective `links` sections point to each other to encode this “styles”/“styled-by” relationship.

Finally, association links are meant to point to other resources external to the catalog that are related to the containing record or the resource(s) that the record describes.

RECOMMENDATION 14

IDENTIFIER	`/rec/record-core/associations_relate`
A	Implementations SHOULD use links in the `links` sections to represent associations to other internal or external resources related to the current record or the resource the record is describing.

In some instances, additional context may be desirable or required to access (or bind to) the resource(s) being described by a record. This is the case when, for example, access to the resource is through a service interface that requires mandatory parameters to be provided. An implementation of the OGC Web Map Service (WMS) Standard is an example of such a service. It may also be the case that efficient access to the resource would be facilitated by providing additional context. Accessing data from a large data repository such as a data cube would be more efficient if the context of the query used to discover the resource (e.g., bbox, datetime) were passed down to the link used to access the data.

RECOMMENDATION 15

IDENTIFIER	`/rec/record-core/associations-templated`
A	If access to the resource(s) requires or would be facilitated by additional context (e.g., a bounding box) then templated links SHOULD be used.

NOTE 2: This Standard does not define any specific link relations that should be used for links in the `links` or `linkTemplates` sections. Such relationships are typically domain specific and so it is left to communities of interest to standardize the relations that links should use.

The following example uses links and link templates to point to resources related to a record. The `describes` links can be used to retrieve the resources described by the record. Other links, such as `enclosure`, can be used to retrieve the same information in archived formats.


```
{
  .
  .
  .
  "links": [
    {
      "rel": "collection",
      "href": "https://woudc.org/data/dataset_info.php"
    },
    {
      "rel": "describes",
      "title": "OGC Web Map Service (WMS)",
      "href": "https://geo.woudc.org/ows?service=WMS&request=GetCapabilities"
    },
    {
      "rel": "describes",
      "title": "OGC Web Feature Service (WFS)",
      "href": "https://geo.woudc.org/ows?service=WFS&request=GetCapabilities"
    },
    {
      "rel": "preview",
      "type": "image/png",
      "title": "Total Ozone Preview Image",
      "href": "https://woudc.org/data/preview.png"
    },
    {
      "rel": "enclosure",
      "type": "text/html",
      "title": "Web Accessible Folder (WAF)",
      "href": "https://woudc.org/archive/Archive-NewFormat/TotalOzone_1.0_1",
      "created": "2015-01-23T00:00:00Z",
      "updated": "2015-01-23T00:00:00Z"
    },
    {
      "rel": "enclosure",
      "type": "application/zip",
      "title": "Static dataset archive file",
      "href": "https://woudc.org/archive/Summaries/dataset-snapshots/totalozone.
zip",
      "created": "2015-01-23T00:00:00Z",
      "updated": "2015-01-23T00:00:00Z"
    },
    {
      "rel": "search",
```

```
      "type": "text/html",
      "title": "Data Search / Download User Interface",
      "href": "https://woudc.org/data/explore.php?dataset=totalozone"
    },
    {
      "rel": "license",
      "href": "https://woudc.org/about/data-policy.php"
    }
  ],
  "linkTemplates": [
    {
      "rel": "describes",
      "title": "World Ozone and Ultraviolet Radiation Data Centre (WOUDC)
stations",
      "uriTemplate": "https://geo.woudc.org/ows?service=WMS&version=
1.3.0&request=GetMap&crs={crs}&bbox={bbox}&layers=stations&width={width}&height=
{height}&format={format}",
      "variables": {
        "bbox": {
          "description": "...",
          "type": "array",
          "items": {
            "type": "number",
            "format": "double"
          },
          "minItems": 4,
          "maxItems": 4
        },
        "crs": {
          "description": "...",
          "type": "string",
          "enum": [
            "EPSG:4326",
            "EPSG:3857"
          ]
        },
        "width": {
          "description": "...",
          "type": "number",
          "format": "integer",
          "minimum": 600,
          "maximum": 5000
        },
        "height": {
          "description": "...",
          "type": "number",
          "format": "integer",
          "minimum": 600,
          "maximum": 5000
        },
        "format": {
          "type": "string",
          "enum": [
            "application/vnd.google-earth.kml+xml",
            "application/vnd.google-earth.kmz",
            "image/png",
            "image/jpeg",
            "image/gif",
            "image/png; mode=8bit",
            "application/x-pdf",
            "image/svg+xml",
            "image/tiff"
          ]
```

```
        }
      }
    },
    {
      "rel": "describes",
      "title": "World Ozone and Ultraviolet Radiation Data Centre (WOUDC)
stations",
      "uriTemplate": "https://geo.woudc.org/ows?service=WFS&version=
1.1.0&request=GetFeature&typeName=woudc:totalozone&maxFeatures=
{maxFeatures}&outputFormat={outputFormat}",
      "variables": {
        "maxFeatures": {
          "type": "number",
          "default": 10
        },
        "outputFormat": {
          "type": "string",
          "enum": [
            "text/xml; subtype=gml/3.1.1",
            "text/xml; subtype=gml/2.1.2; driver=ogr",
            "application/json; subtype=geojson",
            "application/vnd.google-earth.kml+xml",
            "application/vnd.shp",
            "text/plain",
            "text/csv"
          ],
          "default": "text/xml; subtype=gml/2.1.2; driver=ogr"
        }
      }
    }
  ]
}
```

Listing 3 — Example of Links and Link-Templates in a Record

The following example shows links and link-templates specified in the HTTP header of a response.

```
Link-Template: <https://geo.woudc.org/ows?service=WMS&version=1.3.0&request=
GetMap&crs={crs}&bbox={bbox}&layers=stations&width={width}&height=
{height}&format={format}>; rel="describes"; title="World Ozone and Ultraviolet
Radiation Data Centre (WOUDC) stations"; varBase="https://geo.woudc.org/variables
Link-Template: <https://geo.woudc.org/ows?service=WFS&version=1.1.0&request=
GetFeature&typeName=woudc:totalozone&maxFeatures={maxFeatures}&outputFormat=
{outputFormat}; rel="describes"; title="World Ozone and Ultraviolet Radiation
Data Centre (WOUDC) stations"; varBase="https://geo.woudc.org/variables
Link: <https://woudc.org/data/dataset_info.php>; rel="collection"
Link: <https://geo.woudc.org/ows?service=WMS&request=GetCapabilities>; rel=
"describes"; title="OGC Web Map Service (WMS)"
Link: <https://geo.woudc.org/ows?service=WFS&request=GetCapabilities>; rel=
"describes"; title="OGC Web Feature Service (WFS)"
Link: <https://woudc.org/data/preview.png>; rel="preview"; type="image/png";
title="Total Ozone Preview Image"
Link: <https://woudc.org/archive/Archive-NewFormat/TotalOzone_1.0_1>; rel=
"enclosure"; type="text/html", title="Web Accessible Folder (WAF)"; created=
"2015-01-23T00:00:00Z"; updated="2015-01-23T00:00:00Z"
Link: <https://woudc.org/archive/Summaries/dataset-snapshots/totalozone.zip>;
rel="enclosure"; type="application/zip", title="Static dataset archive file";
created="2015-01-23T00:00:00Z"; updated="2015-01-23T00:00:00Z"
Link: <https://woudc.org/data/explore.php?dataset=totalozone">; rel="search";
type="text/html"; title="Data Search / Download User Interface"
```

```
Link: <https://woudc.org/about/data-policy.php>; rel="license"
```

Listing 4 — Example of Links and Link-Templates in HTTP Headers

NOTE 3: Care should be exercised when there is a large number of links that need to be presented. In such cases, it would be prudent to avoid encoding all the links in the HTTP header and instead including them in the body of the response in the link and/or link templates sections.

7.2.14. Templated links with variables

The OGC API — Records Standard uses links to express associations between resources including links that bind to the resource(s) being described by a record. In some situations, a static link does not adequately express all the information necessary to retrieve the referenced resource OR does not allow for the resource to be retrieved in an efficient manner.

Consider the case where a record describing a coverage resource is discovered by searching the catalog using a spatial constraint (e.g., a bbox). Further consider that the found record contains a link to retrieve the coverage. Using a static link would only allow for retrieval of the entire coverage or a subset unrelated to the user's original catalog query. Using a templated link, however, would allow the context of the catalog query (i.e., the bbox) to be passed to the retrieval link as a substitution variable. The templated link with all substitution variables resolved would thus retrieve the subset of the coverage that corresponds to the spatial constraint used to discover the record in the first place which presumably represents the area of interest.

Templated links may also be used to bind to resources that may require additional information in order to access the resources. For example, simply knowing the URL of a legacy OGC service instance such as for WMS or WFS implementations is not sufficient to access resources offered by those services. Additional information in the form of request parameters and values is required in order to have a complete link to a resource. A templated link with variables can provide this additional context.

This Standard defines a new schema, linkTemplate.yaml, based on the schema for a link that includes substitution variables who's values are filled in at runtime prior to resolving the link.

Information about the substitution variables can be obtained in one of two ways. The `variable` property may be used to encode the definitions of substitution variables in-line in the link. Alternatively, the `varBase` property can be used to specify a base URI to which a variable name may be appended to retrieve the definition of the specified substitution variable.

REQUIREMENT 9	
IDENTIFIER	`/req/record-core/templated-link`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core`

A	A templated link SHALL be defined by the following schema fragment:
	`---`
	`allOf:`

REQUIREMENT 9

```
    - $ref: 'linkBase.yaml'
    - type: object
      required:
        - uriTemplate
      properties:
        uriTemplate:
          type: string
          description:
            Supplies a resolvable URI to a remote resource
            (or resource fragment).
        varBase:
          type: string
          description:
            The base URI to which the variable name can be
            appended to retrieve the definition of the
            variable as a JSON Schema fragment.
          format: uri-reference
        variables:
          type: object
          description:
            This object contains one key per substitution
            variable in the templated URL.  Each key defines
            the schema of one substitution variable using a
            JSON Schema fragment and can thus include things
            like the data type of the variable, enumerations,
            minimum values, maximum values, etc.
```

B	The `uriTemplate` member SHALL supply the templated link URI.
C	Substitution variables in the templated link URI SHALL have the form {*variable name*}.

RECOMMENDATION 16

IDENTIFIER	`/rec/record-core/templated-link`
A	A dictionary of substitution variable definitions SHOULD be defined for templated links.
B	The dictionary MAY either be specified in-line in the link using the `variables` member or substitution variable definitions MAY be retrieved by appending the name of the variable to a base URI specified using the `varBase` member.

REQUIREMENT 10

IDENTIFIER	`/req/record-core/templated-link-header`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core`
A	A templated link that appears in a HTTP header SHALL be encoded according to The Link-Template HTTP Header Field specification.

The following example illustrates a link template to a legacy OGC Web service. The example includes both an in-line and referenced dictionaries of substitution variables.

```
{
  "rel": "describes",
  "title": "World Ozone and Ultraviolet Radiation Data Centre (WOUDC) stations",
  "uriTemplate": "https://geo.woudc.org/ows?service=WMS&version=1.3.0&request=
GetMap&crs={crs}&bbox={bbox}&layers=stations&width={width}&height=
{height}&format={format}",
  "variables": {
    "bbox": {
      "description": "...",
      "type": "array",
      "items": {
        "type": "number",
        "format": "double"
      },
      "minItems": 4,
      "maxItems": 4
    },
    "crs": {
      "description": "...",
      "type": "string",
      "enum": [
        "EPSG:4326",
        "EPSG:3857"
      ]
    },
    "width": {
      "description": "...",
      "type": "number",
      "format": "integer",
      "minimum": 600,
      "maximum": 5000
    },
    "height": {
      "description": "...",
      "type": "number",
      "format": "integer",
      "minimum": 600,
      "maximum": 5000
    },
    "format": {
      "type": "string",
      "enum": [
        "application/vnd.google-earth.kml+xml",
        "application/vnd.google-earth.kmz",
        "image/png",
        "image/jpeg",
        "image/gif",
        "image/png; mode=8bit",
        "application/x-pdf",
        "image/svg+xml",
        "image/tiff"
      ]
    }
  }
}
```

Listing 6 — Templated link with in-line dictionary of substitution variables:

```
{
  "rel": "item",
  "type": "image/png",
  "title": "World Ozone and Ultraviolet Radiation Data Centre (WOUDC) stations",
  "uriTemplate": "https://geo.woudc.org/ows?service=WMS&version=1.3.0&request=
GetMap&crs={crs}&bbox={bbox}&layers=stations&width={width}&height=
{height}&format=image/png",
  "varBase": "variables.json"
}
```

Listing 7 — Templated link with `varBase` URI for retrieving substitution variable definitions:

```
CLIENT                                                    SERVER
   |                                                        |
   |   GET variables/bbox HTTP/1.1                          |
   |   Host: example.org                                    |
   |   Accept: application/json                             |
   |------------------------------------------------------->|
   |                                                        |
   |   HTTP/1.1 200 OK                                      |
   |   Content-Type: application/json                       |
   |                                                        |
   |   {                                                    |
   |     "bbox": {                                          |
   |       "description": "...",                            |
   |       "type": "array",                                 |
   |       "items": {                                       |
   |         "type": "number",                              |
   |         "format": "double"                             |
   |       },                                               |
   |       "minItems": 4,                                   |
   |       "maxItems": 4                                    |
   |     }                                                  |
   |   }                                                    |
   |<-------------------------------------------------------|
```

Listing 8 — Using `varBase` to retrieve the definition of the variable bbox

7.2.15. Resource timestamps

A record contains information, encoded in the links and link templates sections, for accessing one or more resources described by the record. Since a resource may have multiple representations there may be multiple links pointing to each representation of a resource. Furthermore, representations of a resource may be created and updated at different times. In order to capture this life cycle information of resource representations, the link schema is extended with the addition of two properties named `created` and `updated`. The values of these properties are used to indicate the creation and last updated dates of the resource representation pointed to by the link.

7.2.16. Record encodings

This Standard defines requirements classes for JSON and HTML encoding of a record. See Encodings.

7.3. Requirements Class “Record Collection” (Common Component)

7.3.1. Overview

REQUIREMENTS CLASS 2	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
TARGET TYPE	Document model
CONFORMANCE CLASS	Conformance class A.2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-collection`
PREREQUISITE	https://docs.ogc.org/is/17-069r4/17-069r4.html#_response_5
NORMATIVE STATEMENTS	Requirement 11: `/req/record-collection/mandatory-properties-collection` Requirement 12: `/req/record-collection/itemType` Requirement 13: `/req/record-collection/contact` Requirement 14: `/req/record-collection/license` Requirement 15: `/req/record-collection/links-record` Requirement 16: `/req/record-collection/links-records` Requirement 17: `/req/record-collection/records` Requirement 18: `/req/record-collection/records-array-name` Requirement 19: `/req/record-collection/links-catalog-related` Requirement 20: `/req/record-collection/links-catalog-series` Requirement 21: `/req/record-collection/links-catalog-hierarchy` Requirement 22: `/req/record-collection/links` Requirement 93: `/req/record-collection/default-mediatype`

The Record Collection requirements class defines the requirements for a record collection.

A *record collection* is an object that provides information about and access to a set of related records. Such a collection of records is also referred to as a *catalog*.

The schema for the catalog is an extension of the collection schema defined in OGC API — Features Standard.

While OGC API — Features — Part 1: Core defines a specific location for the collection resource (path: `/collections/{collectionId}`), OGC API — Records only fixes the location of the catalog in the Records API conformance class. Otherwise, the catalog resource can live anywhere the provider wishes to place it including as a static file in web space (e.g., a web-accessible directory or an S3 bucket).

Although this Standard does not mandate any particular encoding for a catalog, two conformance classes are defined:

- a JSON catalog encoding, and
- an HTML encoding.

Other encodings are allowed but are not described in this document.

7.3.2. Catalog schema

Table 11 defines the set of properties that may be used to describe a catalog.

Table 11 — Catalog properties

PROPERTY	REQUIREMENT	DESCRIPTION
id	**required**	A unique identifier for this catalog.
created	optional	The date this collection was created.
updated	optional	The more recent date on which this collection was changed.
conformsTo	optional	The extensions/conformance classes used in this catalog object.
type	**required**	Fixed value of “Collection”.
itemType	optional	Fixed value of “record”, “catalog” or both.
title	optional	A human-readable name given to this catalog.
description	optional	A free-text description of this catalog.
extent	optional	The spatiotemporal coverage of this catalog.
crs	optional	A list of coordinate reference systems used for spatiotemporal values.
keywords	optional	A list of free-form keywords or tags associated with this collection.
themes	optional	A knowledge organization system used to classify this collection.
language	optional	The language used for textual values (i.e., titles, descriptions, etc.) of this collection object.
languages	optional	The list of other languages in which this collection object is available.

PROPERTY	REQUIREMENT	DESCRIPTION
record Languages	optional	The list of languages in which records from the collection can be represented.
contacts	optional	A list of contacts qualified by their role(s).
license	optional	The legal provisions under which this collection is made available.
rights	optional	A statement that concerns all rights not addressed by the license such as a copyright statement.
recordsArray Name	optional	The name of the array property in the catalog used to encode records in-line. The default value is `records`.
records	optional	An array of records encoded in-line in the catalog.
links	**required**	A list of links related to this catalog.
linkTemplates	optional	A list of link templates related to this catalog.
schemes	optional	A list of schemes related to this catalog.

NOTE: The properties *id*, *itemType*, *title*, *description*, *extent*, *crs* and *links* are inherited from OGC API — Features — Part 1: Core.

REQUIREMENT 11

IDENTIFIER	`/req/record-collection/mandatory-properties-collection`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
A	A catalog SHALL include all the mandatory properties listed in Table 11.

PERMISSION 6

IDENTIFIER	`/per/record-collection/common-mark`
A	For any `description` member in a catalog, CommonMark MAY be used for a rich text representation.

PERMISSION 7	
IDENTIFIER	`/per/record-collection/additional-properties`
A	A catalog MAY contain zero or more of the optional properties listed in Table 11.
B	A catalog MAY include any number of additional properties not listed in Table 11. The meaning of these additional properties is not defined in this standard.

7.3.3. Catalog extensions

In Clause 7.3.2 this Standard defines a set of core properties for describing a catalog. It is anticipated that this set of properties will be extended to suit specific use cases or requirements. The `conformsTo` property is a list of identifiers that indicate each extension used in the description of the catalog. This Standard does not define the specific identifiers that are used to identify specific extensions; it simply provides a place where these identifiers can be listed.

In the case where all the records of a catalog use the same set of extensions, and to prevent unnecessary duplication, the `conformsTo` property of the catalog can also be used to declare which extensions are used that apply to all records.

See Record extensions.

7.3.4. i.e., type

A catalog can include references to records, other catalogs or both. A catalog can also contain an array of catalog records.

REQUIREMENT 12	
IDENTIFIER	`/req/record-collection/itemType`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
A	If a catalog homogeneously references or contains records then, its `itemType` property SHALL be a string with the fixed value of `record`.
B	If a catalog homogeneously references other catalogs then, its `itemType` property SHALL be a string with the fixed value of `catalog`.
C	If a catalog references or contains records and references other catalogs then, its `itemType` property SHALL be an array of strings whose elements enumerate the types of resources referenced/contained by this catalog (i.e., `record` and `catalog`).

7.3.5. Title and description

In order to facilitate the discovery of resources utilizing keyword or free-text searches, this Standard makes the following recommendation:

RECOMMENDATION 17	
IDENTIFIER	`/rec/record-core/catalog-title`
A	A catalog SHOULD include the `title` property.

RECOMMENDATION 18	
IDENTIFIER	`/rec/record-core/catalog-description`
A	A catalog SHOULD include the `description` property.

7.3.6. Keywords and Themes

RECOMMENDATION 19	
IDENTIFIER	`/rec/record-collection/keywords-themes`
A	Implementations SHOULD use the `keywords` property to provide free-form terms or tags associated with this catalog.
B	Implementations SHOULD use the `themes` property to enumerate 1..n concepts, and their respective knowledge organization system/controlled vocabulary, associated with the catalog.

See Keywords and Themes.

7.3.7. Language handling

RECOMMENDATION 20	
IDENTIFIER	`/rec/record-collection/langs`

RECOMMENDATION 20	
A	The language used to express text values in a catalog SHOULD be specified using the `language` property.
B	The list of other languages in which the catalog may be retrieved SHOULD be specified using the `languages` property.
C	If one or more language-specific links are included in the links section of a catalog (path: `links`) pointing to alternate language representations of this catalog, the language used as the `hreflang` value SHOULD be one of the languages listed as the value of the `languages` property of the catalog.

RECOMMENDATION 21	
IDENTIFIER	`/rec/record-collection/record-langs`
A	The languages in which the records of the catalog can be represented SHOULD be listed using the `recordLanguages` property.

See Language handling.

7.3.8. Contacts

REQUIREMENT 13	
IDENTIFIER	`/req/record-collection/contact`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
A	See requirement /req/record-core/contact.

7.3.9. License

REQUIREMENT 14	
IDENTIFIER	`/req/record-collection/license`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
A	See requirement /req/record-core/license. Replace all instances of the term `resource` with the term `catalog`.

7.3.10. Records and RecordsArrayName

7.3.10.1. Overview

PERMISSION 8	
IDENTIFIER	`/per/record-collection/records`
A	The records of a catalog MAY be: 1. accessed individually via one or more links in the catalog's links section, 2. accessed via an API whose access endpoint is identified by a link in the catalog's links section, 3. accessed from an array encoded directly in-line in the catalog.

PERMISSION 9	
IDENTIFIER	`/per/record-collection/record-access-modes`
A	A catalog MAY simultaneously include all three of the aforementioned access modes.

Thus a catalog may simultaneously include links to individual records, a link to an API endpoint for accessing records and also include records encoded in-line in the catalog.

7.3.10.2. Referencing individual catalog records

REQUIREMENT 15	
IDENTIFIER	`/req/record-collection/links-record`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
A	A link (relation: `item`) SHALL be included in the links section of the catalog for each individually referenced record that is a member of this catalog.

A crawlable catalog is an example of catalog where the records can be individually accessed via links (relation: item) in the catalog's links section.

7.3.10.3. Referencing records accessible via an API

REQUIREMENT 16	
IDENTIFIER	`/req/record-collection/links-records`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
A	A link (relation: `items`) SHALL be included in the links section of the catalog pointing to an endpoint for accessing the records of this catalog via an API.

A searchable catalog is an example of a catalog where the records are accessed from an API endpoint identified using a link (relation: items) in the catalog's links section.

7.3.10.4. Encoding records in-line

REQUIREMENT 17	
IDENTIFIER	`/req/record-collection/records`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
A	Records encoded in-line in the catalog SHALL be encoded using an array property named `records`.

PERMISSION 10	
IDENTIFIER	`/per/record-collection/records-array-name`
A	The name of the in-line records array property in the catalog (default: `records`) MAY be changed to something else.

A local-resources-catalog is an example of a catalog where its records are encoded in-line in the catalog. In the case of a local resources catalog, however, the name of the array property containing the records is typically named something other than *"records"*.

REQUIREMENT 18	
IDENTIFIER	/req/record-collection/records-array-name
INCLUDED IN	Requirements class 2: http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection
A	If the name of the in-line records array property in the catalog is changed to something other than records then that name SHALL be indicated in the catalog using a property named recordsArrayName.

For example, a local resources catalog at the /collections endpoint uses an array property named collections to encode the records of the catalog and so the value of the recordsArrayName property in this case would be:

```
"recordsArrayName": "collections"
"collections": [
  {... record describing a collection ... },
  {... record describing a collection ... },
  .
  .
  .
  ]
```

Listing 9

Similarly, a local resources catalog at the /processes endpoint uses an array named processes to encode the records of the catalog and so the value of the recordsArrayName property would be:

```
"recordsArrayName": "processes"
"processes": [
  { ... record describing a process ... },
  { ... record describing a process ... },
  .
  .
  .
  ]
```

Listing 10

7.3.11. Other catalogs

7.3.11.1. Overview

PERMISSION 11	
IDENTIFIER	/per/record-collection/catalogs
A	A catalog MAY include zero or more references to other related catalogs.

Related catalogs may be organized as:

- an unorganized set of related catalogs,
- as a series of related catalogs (e.g., for paging),
- as a hierarchy of related catalogs.

7.3.11.2. Related catalogs

REQUIREMENT 19	
IDENTIFIER	`/req/record-collection/links-catalog-related`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
A	A catalog referencing another related catalog in an unorganized set of catalogs SHALL include a link (relation: `related`) pointing to the related catalog.

7.3.11.3. Series of related catalogs

REQUIREMENT 20	
IDENTIFIER	`/req/record-collection/links-catalog-series`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
A	A catalog referencing another catalog in a series of related catalogs SHALL include a link (relation: `next`) pointing to the next related catalog in the series.

RECOMMENDATION 22	
IDENTIFIER	`/rec/record-collection/links-catalog-series`
A	A catalog referencing another catalog in a series of related catalogs SHOULD include a link (relation: `prev` or `previous`) pointing to the previous related catalog in the series.
B	A catalog referencing another catalog in a series of related catalogs SHOULD include a link (relation: `first`) pointing to the first catalog in the series.

7.3.11.4. Hierarchy if related catalog

REQUIREMENT 21	
IDENTIFIER	`/req/record-collection/links-catalog-hierarchy`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
A	A catalog referencing sub-ordinate catalogs SHALL include a link (relation: `child`) pointing to each immediate sub-ordinate catalog in the hierarchy.

RECOMMENDATION 23	
IDENTIFIER	`/rec/record-collection/links-catalog-hierarchy`
A	A child catalog SHOULD include a link (relation: `parent`) pointing to its immediate parent catalog in the hierarchy.
B	A child catalog SHOULD include a link (relation: `root`) pointing to the root catalog of the hierarchy.

7.3.12. Links

REQUIREMENT 22	
IDENTIFIER	`/req/record-collection/links`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
A	The catalog SHALL include a link (relation: `self`) that points to itself.
B	The catalog SHALL include one link (relation: `alternate`) for each alternative representation of itself.

PERMISSION 12	
IDENTIFIER	`/per/record-collection/links`

PERMISSION 12	
A	The links in the `links` section of a catalog may be typed (i.e., include the `type` member) to allow for the possibility of alternate representations for each record and/or subordinate catalog.

7.3.13. Templated links with variables

See Templated links with variables.

7.3.14. Schemes

In various places in a catalog or catalog record property values can be qualified by an authority, scheme or namespace. Typically such schemes are represented as a URI. In order to alleviate the need to keep repeating such long URI strings, the schemes mechanism can be used to assign a short identified to each scheme and that identified can, instead, be used to qualify the value.

Consider the value `atmosphericComposition` that is qualified with the schema `https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_CategoryCode`. Rather than, for example, specifying this value as `https//wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_CategoryCode:atmosphericComposition` in an expression, the value `categoryCode` can be associated with the URI `https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_CategoryCode` and the value can be specified as `categoryCode:atmosphericComposition`. The schemes mechanism provides the means to associated the identifier `categoryCode` with the URI `https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_CategoryCode` as shown in the following JSON fragment:

```
  "schemes": [
    .
    .
    .
    {
      "scheme-id": "categoryCode",
      "namespace" : "https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_
CategoryCode"
    },
    .
    .
    .
  ]
```

Listing 11

Once defined in the `schemes` section of a catalog, the identifier `categoryCode` can be used wherever the long namespace `https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_CategoryCode` would normally be used.

7.3.15. Encodings

This Standard defines requirements for JSON and HTML encoding of a catalog. See Encodings.

7.4. Requirements Class “Record Core Query Parameters” (Common Component)

7.4.1. Overview

REQUIREMENTS CLASS 3	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters`
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.4: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core-query-parameters`
NORMATIVE STATEMENTS	Requirement 23: `/req/record-core-query-parameters/bbox` Requirement 24: `/req/record-core-query-parameters/datetime` Requirement 25: `/req/record-core-query-parameters/limit` Requirement 26: `/req/record-core-query-parameters/q-definition` Requirement 27: `/req/record-core-query-parameters/q-response` Requirement 28: `/req/record-core-query-parameters/type-definition` Requirement 29: `/req/record-core-query-parameters/type-response` Requirement 30: `/req/record-core-query-parameters/ids-definition` Requirement 31: `/req/record-core-query-parameters/ids-response` Requirement 32: `/req/record-core-query-parameters/externalIds-definition` Requirement 33: `/req/record-core-query-parameters/externalIds-response`

The Record Core Query Parameters requirements class defines a minimum set of query parameters that should be implemented at a searchable catalog endpoint. Table 12 lists this set of query parameters.

The parameters `bbox`, `datetime`, `limit` and `ids` are inherited from OGC API Features — Part 1: Core. The remaining parameters are defined in this Standard.

Table 12 — Core query parameters

PARAMETER NAME	DESCRIPTION
bbox	A bounding box. If the spatial extent of the record intersects the specified bounding box, then the record shall be presented in the response document.
datetime	A time instance or time period. If the temporal extent of the record intersects the specified date/time value, then the record shall be presented in the response document.

PARAMETER NAME	DESCRIPTION
limit	The number of records to be presented in a response document.
q	A comma-separated list of search terms. If any server-chosen text field in the record contains 1 or more of the terms listed, then this record shall appear in the response set.
type	An equality predicate consisting of a comma-separated list of resource types. Only records of the listed type shall appear in the response set.
ids	An equality predicate consisting of a comma-separated list of record identifiers. Only records with the specified identifiers shall appear in the response set.
externalIds	An equality predicate consisting of a comma-separated list of external resource identifiers. Only records with the specified external identifiers shall appear in the response set.
prop=value	Equality predicates with any queryable not already listed in this table.

7.4.2. Query parameters

7.4.2.1. Parameter bbox

REQUIREMENT 23	
IDENTIFIER	`/req/record-core-query-parameters/bbox`
INCLUDED IN	Requirements class 3: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters`
A	A searchable endpoint SHALL support the Bounding Box Search (`bbox`) parameter as defined in the OGC API — Features — Part 1: Core Standard.
B	All references to the term “features” in the OGC API — Features — Part 1: Core Standard SHALL be replaced by the term “records” or “local resources” as the context may indicate.

7.4.2.2. Parameter datetime

REQUIREMENT 24	
IDENTIFIER	`/req/record-core-query-parameters/datetime`
INCLUDED IN	Requirements class 3: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters`

REQUIREMENT 24	
A	A searchable endpoint SHALL support a Temporal Searching (`datetime`) parameter as defined in the OGC API — Features — Part 1: Core Standard.
B	All references to the term “features” in the OGC API — Features — Part 1: Core Standard SHALL be replaced by the term “records” or “local resources” as the context may indicate.

7.4.2.3. Parameter limit

REQUIREMENT 25	
IDENTIFIER	`/req/record-core-query-parameters/limit`
INCLUDED IN	Requirements class 3: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters`
A	A searchable endpoint SHALL support a Paging (`limit`) parameter as defined in the OGC API — Features — Part 1: Core Standard.
B	All references to the term “features” in the OGC API — Features — Part 1: Core Standard SHALL be replaced by the term “records” or “local resources” as the context may indicate.

7.4.2.4. Parameter q

The `q` parameter provides a simple, easy-to-implement keyword search capability across one or more text fields in a record.

REQUIREMENT 26	
IDENTIFIER	`/req/record-core-query-parameters/q-definition`
INCLUDED IN	Requirements class 3: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters`
A	A searchable endpoint SHALL support the Text Search (q) parameter for the operation.

B	The `q` parameter SHALL have the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`---`
	`name: q`
	`in: query`
	`description: |-`
	` The optional q parameter supports keyword searching. Only records whose text fields contain one or more of the specified search terms are selected. The specific set of text keys/fields/properties of a record to which the q operator is applied is up to the discretion of the server. Implementations should, however, apply the q operator to the title, description and keywords keys/fields/properties.`

REQUIREMENT 26

```
required: false
schema:
  type: array
  items:
    type: string
explode: false
style: form
```

C	Search terms that may appear together (logical OR) in a record SHALL be separated by literal commas.
D	Search terms that must appear together, and in the order specified, in a record SHALL be separated by one or more white space characters.
E	Keyword searches using the q parameter SHALL be case insensitive.
F	The specific set of text keys/fields/properties of a record to which the q operator is applied SHALL be left to the discretion of the implementation.

The specific text fields in a record that are to be searched is not mandated in this Standard.

RECOMMENDATION 24

IDENTIFIER	`/rec/record-core-query-parameters/param-q`
A	The q operator SHOULD, at least, be applied to the following core properties: • title • description • keywords

REQUIREMENT 27

IDENTIFIER	`/req/record-core-query-parameters/q-response`
INCLUDED IN	Requirements class 3: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters`
A	If a single search term is specified, then only records that contain that search term in one or more of the searched text fields SHALL be in the result set.
B	For multiple search terms that are comma separated (logical OR), only records that contain one or more the specified search terms in one or more of the searched text fields SHALL be in the result set.
C	For multiple search terms that are white space separated, only records that contain all the search terms specified, in the order specified and separated by any number of white spaces in one or more of the searched text fields SHALL be in the result set.

For example, the following `q` parameter specification:

```
q=ocean,climate%20%09change,desalination
```

Listing 13

represents the query predicate:

```
(anytext CONTAINS 'ocean') OR
(anytext CONTAINS 'climate\s+change') OR
(anytext CONTAINS 'desalination')`
```

Listing 14

NOTES:

- The term `anytext` represents the set of text fields that are searched by the operation.
- The `CONTAINS` operator used above is just an example of an operator that searches a text field(s) for specific search terms. How this operator is implemented and what it is named in the system backing a catalog implementation is beyond the scope of this Standard.
- The regular expression `\s+` represents one or more white spaces.

7.4.2.5. Parameter type

REQUIREMENT 28	
IDENTIFIER	`/req/record-core-query-parameters/type-definition`
INCLUDED IN	Requirements class 3: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters`
A	A searchable endpoint SHALL support the search by Type (`type`) parameter for the operation.
B	The `type` parameter SHALL have the following characteristics (using an OpenAPI Specification 3.0 fragment):

STATEMENT	`---`
	`name: type`
	`in: query`
	`description: |-`
	`  The optional type parameter allows a specific list of records, identified`
	`  by their resource type, to be fetched from a catalog.  Only records whose`
	`  resource type matches one of the values listed for this parameter shall`
	`  appear in the response.`
	`required: false`
	`schema:`
	`  type: array`
	`  items:`
	`  type: string`
	`explode: false`
	`style: form`

RECOMMENDATION 25	
IDENTIFIER	`/rec/record-core-query-parameters/param-type-definition`
A	The definition of the `type` parameter SHOULD be extended to enumerate the list of known record or resource types.

REQUIREMENT 29	
IDENTIFIER	`/req/record-core-query-parameters/type-response`
INCLUDED IN	Requirements class 3: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters`
A	Only records whose type, as indicated by the value of the `type` core queryable, is equal to one of the listed values specified using the `type` parameter SHALL be in the result set.

7.4.2.6. Parameter ids

REQUIREMENT 30	
IDENTIFIER	`/req/record-core-query-parameters/ids-definition`
INCLUDED IN	Requirements class 3: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters`
A	A Records API implementation SHALL support the query by Ids (`ids`) parameter for the operation.

B	The following OpenAPI 3.0 schema fragment SHALL define the characteristics of the `ids` parameter:
	`---`
	`name: ids`
	`in: query`
	`description:`
	`  The optional ids parameter allows a specific list of records, identified`
	`  by their identifiers, to be fetched from a catalog.  Only records whose`
	`  identifier matches one of the values listed for this parameter shall appear`
	`  in the response.`
	`required: false`
	`schema:`
	`  type: array`
	`  items:`
	`    type: string`
	`explode: false`
	`style: form`

REQUIREMENT 31	
IDENTIFIER	/req/record-core-query-parameters/ids-response
INCLUDED IN	Requirements class 3: http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters
A	Only records whose identifier match one of the identifiers specified using ids parameter SHALL be in the results set.

7.4.2.7. Parameter externalIds

REQUIREMENT 32	
IDENTIFIER	/req/record-core-query-parameters/externalIds-definition
INCLUDED IN	Requirements class 3: http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters
A	A searchable endpoint SHALL support search by External Identifiers (externalIds) parameter for the operation.

B	The externalIds parameter SHALL have the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`---`
	`name: externalIds`
	`in: query`
	`description: |-`
	`  The optional externalIds parameter allows a specific list of records,`
	`  identified by their external identifiers, to be fetched from a catalog.`
	`  Only records where one of their associated external identifiers equals`
	`  one of the values listed for this parameter shall appear in the response.`
	`required: false`
	`schema:`
	`  type: array`
	`  items:`
	`  type: string`
	`  pattern: ([^:]+:)?[^:]+`
	`explode: false`
	`style: form`

REQUIREMENT 33	
IDENTIFIER	/req/record-core-query-parameters/externalIds-response
INCLUDED IN	Requirements class 3: http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters
A	Only records that have an external identifier that matches one of the values listed using the externalIds parameter SHALL be in the result set.

REQUIREMENT 33	
B	If the search value is qualified with a scheme then both the scheme and the value of the record's external identifier SHALL match in order for the record to be in the result set.
C	If the search value is not qualified with a scheme then only the value of the record's external identifier SHALL match in order for the record to be in the result set.
D	If the search value is only the scheme then the scheme component of the record's external identified SHALL match in order for the record to be in the result set.

7.4.2.8. Additional query parameters

This Standard defines the following URL query parameters based on the set of core properties to support record selection based on the resource type and/or an external identifier associated with the resource.

- type
- externalIds

RECOMMENDATION 26	
IDENTIFIER	`/rec/record-core-query-parameters/additional-query-parameters`
A	If any additional property (see additional record properties) is expected to be useful for applications using at a searchable catalog endpoint, a parameter with the name of that property SHOULD be supported.

B	If the parameter has a simple value (e.g., a string or integer) then it SHOULD have the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: <name of property>`
	`in: query`
	`required: false`
	`style: form`
	`explode: false`

C	If the parameter is an array of simple values then it SHOULD have the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: <name of property>`
	`in: query`
	`required: false`
	`schema:`
	`  type: array`
	`explode: false`

D	The `schema` property SHOULD be the same as the definition of the property from which the parameter is derived (see additional record properties).

7.5. Requirements Class “Records API” (Common Component)

7.5.1. Overview

REQUIREMENTS CLASS 4	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/records-api`
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.3: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/records-api`
PREREQUISITES	http://www.opengis.net/spec/ogcapi-features-1/1.0/req/core Records Core Query Parameters Record Collection Record Core
NORMATIVE STATEMENTS	Requirement 34: `/req/records-api/features-api` Requirement 35: `/req/records-api/resource-name-mapping` Requirement 36: `/req/records-api/catalog-response` Requirement 37: `/req/records-api/catalogs-response` Requirement 38: `/req/records-api/records-op` Requirement 39: `/req/records-api/mandatory-params` Requirement 40: `/req/records-api/query-params` Requirement 41: `/req/records-api/record-op` Requirement 42: `/req/records-api/record-response`

REQUIREMENT 34	
IDENTIFIER	`/req/records-api/features-api`
INCLUDED IN	Requirements class 4: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/records-api`
A	Implementations of this requirements class SHALL implement the Core requirements class of OGC API — Features — Part 1: Core.

The core requirements for an API that enables searching a catalog are defined by the OGC API — Features — Part 1: Core Standard.

REQUIREMENT 35	
IDENTIFIER	`/req/records-api/resource-name-mapping`
INCLUDED IN	Requirements class 4: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/records-api`
A	Each reference to `features` or `feature` in the narrative text of OGC API — Features — Part 1: Core Standard SHALL be replaced by the terms `records` or `record` when interpreting an OGC API — Features — Part 1: Core requirement for the Records API.

NOTE: This requirement does not imply that the terms `features` or `feature` in the schemas or code examples of OGC API — Features — Part 1: Core should be replaced by the terms `records` or `record`. For example, the GeoJSON encoding of a record and the GeoJSON encoding of a feature both include a top-level field named `type` whose value is fixed to `Feature` as defined in the GeoJSON specification. This requirement does not imply that the value `Feature` should be changed to `Record` in this Standard since such a change would invalidate the GeoJSON document.

7.5.2. Encodings

While this Standard does not specify any mandatory response encoding, support for the following encodings is recommended.

RECOMMENDATION 27	
IDENTIFIER	`/rec/records-api/html`
A	To support browsing the catalog and its records with a web browser and to enable search engines to crawl and index the catalog, implementations SHOULD consider supporting a HTML encoding.

RECOMMENDATION 28	
IDENTIFIER	`/rec/records-api/json`
A	If the records can be represented for the intended use in GeoJSON, implementations SHOULD consider supporting GeoJSON as an encoding for records and catalogs (i.e., record collections).
B	If a catalog (i.e., record collection) can be represented for the intended use in JSON, implementations SHOULD consider supporting JSON as an encoding for a catalog (i.e., record collection).

The encoding of a server response is determined according to the mechanisms defined in Clause 12 of the HTTP 1.1 Standard.

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

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

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

NOTE: Two common approaches are:

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

7.5.3. Record collection response

REQUIREMENT 36	
IDENTIFIER	`/req/records-api/catalog-response`
INCLUDED IN	Requirements class 4: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/records-api`

A	For an operation that selects a collection object, the body of the response SHALL contain a description of the catalog based upon the following schema fragment which extends the collection.yaml schema defined in the OGC API — Features — Part 1: Core standard:
	`---`
	`allOf:`
	`  - $ref: 'catalog.yaml'`
	`  - type: object`
	`  properties:`
	`  itemType:`
	`  description:`
	`  For a collection of records, the itemType is`
	`  fixed to "record".`
	`  type: string`
	`  const: record`

B	The collection SHALL homogeneously reference records.
C	The value of the `itemType` property SHALL be a string with a fixed value of `record`.

7.5.4. Record collections response

REQUIREMENT 37	
IDENTIFIER	`/req/records-api/catalogs-response`
INCLUDED IN	Requirements class 4: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/records-api`
A	For an operation that selects a list of collection objects, only collections where the `itemType` property (JSONPath: `$.collections[*].itemType`) is a string with the value `record` SHALL be considered to be catalogs.
B	The schema of each catalog in the response SHALL be based upon the schema defined in the /req/records-api/catalog-response requirement.

7.5.5. Records access

7.5.5.1. Operation

REQUIREMENT 38	
IDENTIFIER	`/req/records-api/records-op`
INCLUDED IN	Requirements class 4: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/records-api`
A	For every catalog identified in the collections response (path: `/collections`), the server SHALL support the HTTP GET operation at the path `/collections/{collectionId}/items`.
B	The parameter `collectionId` is each `id` property in the collections response (JSONPath: `$.collections[*].id`) where the `itemsType` property (JSONPath: `$.collections[*].itemType`) is a string with the value `record`.

REQUIREMENT 39	
IDENTIFIER	`/req/records-api/mandatory-params`
INCLUDED IN	Requirements class 4: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/records-api`
A	An implementation of the Records API SHALL support the parameters listed in Table 12 at the `/collections/{collectionId}/items` endpoint.

REQUIREMENT 39	
B	The parameter `collectionId` is each `id` property in the collections response (JSONPath: `$.collections[*].id`) where the value of the `itemType` (JSONPath `$.collections[*].itemType`) is `record`. `id` is a local identifier of the record.

PERMISSION 13	
IDENTIFIER	`/per/records-api/query-params`
A	Any combination of query parameters from the Table of Query Parameters and any additional query parameters MAY be specified on the operation for the purpose of selecting a subset of catalog records (i.e., filtering).

7.5.5.2. Response

REQUIREMENT 40	
IDENTIFIER	`/req/records-api/query-params`
INCLUDED IN	Requirements class 4: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/records-api`
A	Only records satisfying all the predicates specified by the included query parameters and additional query parameters SHALL be in the result set.
B	The implied logical operation between query parameters/additional query parameters SHALL be **"AND"**.

7.5.5.3. Response encoding

Table 13 lists specific requirements based on the negotiated representation of the response.

Table 13 — Negotiated representation of response

REPRESENTATION	REQUIREMENTS
JSON/GeoJSON	Requirements Class JSON
HTML	Requirements Class HTML

7.5.5.4. Examples

The following is an example of a response to an OGC API — Records query operation (path: `/collections/{collectionId}/items`). This example shows a JSON response and the response container is a GeoJSON feature collection. Specific records are not presented in this example to illustrate the structure of the response container in GeoJSON.


```
{
  "type": "FeatureCollection",
  "timeStamp": "2021-03-06T07:10:00Z",
  "numberReturned": 10,
  "numberMatched": 347,
  "features": [
     { ... record ... },
     { ... record ... },
     { ... record ... },
     { ... record ... },
     { ... record ... },
     .
     .
     .
  ],
  "links": [
    {
       "href": "https://www.someserver.com/ogcapi/collections/tepcat?f=json...",
       "rel": "collection",
       "type": "application/geo+json",
       "title": "Link to the containing collection in JSON"
    },
    {
       "href": "https://www.someserver.com/ogcapi/collections/tepcat?f=html...",
       "rel": "collection",
       "type": "text/html",
       "title": "Link to the containing collection in HTML"
    },
    {
       "href": "https://www.someserver.com/ogcapi/collections/tepcat/items?f=
json&...",
       "rel": "self",
       "type": "application/geo+json",
       "title": "Link to this response"
    },
    {
       "href": "https://www.someserver.com/ogcapi/collections/tepcat/items?f=
html&...",
       "rel": "alternate",
       "type": "text/html",
       "title": "Link to this response in HTML"
    },
    {
       "href": "https://www.someserver.com/ogcapi/collections/tepcat/items?f=
json&startIndex=20&limit=10...",
       "rel": "next",
       "type": "application/geo+json",
       "title": "Link to the next set of records in GeoJSON"
    },
    {
       "href": "https://www.someserver.com/ogcapi/collections/tepcat/items?f=
html&startIndex=20&limit=10...",
```

```
      "rel": "next",
      "type": "text/html",
      "title": "Link to the next set of records in HTML"
    }
  ]
}
```


Listing 21 — Records Example in GeoJSON

The following is an example of a single catalog record encoded in GeoJSON. Such a record might appear in the `features` array of a query response or it may be the response when a specific, single record is accessed (path: `/collections/{collectionId}/items/{recordId}`).


```
{
  "id": "urn:x-wmo:md:int.wmo.wis::https://geo.woudc.org/def/data/ozone/total-
column-ozone/totalozone",
  "type": "Feature",
  "time": {
    "interval": [
      "1924-08-17T00:00:00Z",
      ".."
    ],
    "resolution": "P1D"
  },
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [
        [
          -180,
          -90
        ],
        [
          -180,
          90
        ],
        [
          180,
          90
        ],
        [
          180,
          -90
        ],
        [
          -180,
          -90
        ]
      ]
    ]
  },
  "conformsTo": [
    "http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core"
  ],
  "properties": {
    "created": "2021-02-08T00:00:00Z",
    "updated": "2021-02-08T00:00:00Z",
    "type": "dataset",
    "title": "Total Ozone - daily observations",
    "description": "A measurement of the total amount of atmospheric ozone in
a given column from the surface to the edge of the atmosphere. Ground based
```

```
instruments such as spectrophotometers and ozonemeters are used to measure
results daily",
    "keywords": [
      "total",
      "ozone",
      "level 1.0",
      "column",
      "dobson",
      "brewer",
      "saoz"
    ],
    "language": {
      "code": "en-CA",
      "name": "English (Canada)"
    },
    "languages": [
      {
        "code": "en-CA",
        "name": "English (Canada)"
      },
      {
        "code": "fr-CA",
        "name": "French (Canada)"
      }
    ],
    "externalIds": [
      {
        "scheme": "WMO:WIS",
        "value": "urn:x-wmo:md:int.wmo.wis::https://geo.woudc.org/def/data/ozone/
total-column-ozone/totalozone"
      }
    ],
    "contacts": [
      {
        "name": "World Ozone and Ultraviolet Radiation Data Centre",
        "links": [
          {
            "href": "https://woudc.org",
            "rel": "about",
            "type": "text/html"
          }
        ],
        "contactInstructions": "SEE PAGE: https://woudc.org/contact.php",
        "roles": [
          "publisher"
        ]
      }
    ],
    "themes": [
      {
        "concepts": [
          {
            "id": "dobson",
            "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
dobson"
          },
          {
            "id": "brewer",
            "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
brewer"
          },
          {
            "id": "vassey",
```

```
            "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
vassey"
          },
          {
            "id": "pion",
            "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
pion"
          },
          {
            "id": "microtops",
            "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
microtops"
          },
          {
            "id": "spectral",
            "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
spectral"
          },
          {
            "id": "hoelper",
            "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
hoelper"
          },
          {
            "id": "saoz",
            "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
saoz"
          },
          {
            "id": "filter",
            "url": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode_
filter"
          }
        ],
        "scheme": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode"
      },
      {
        "concepts": [
          {
            "id": "atmosphericComposition",
            "url": "https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_
CategoryCode_atmosphericComposition"
          },
          {
            "id": "pollution",
            "url": "https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_
CategoryCode_pollution"
          },
          {
            "id": "observationPlatform",
            "url": "https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_
CategoryCode_observationPlatform"
          },
          {
            "id": "rocketSounding",
            "url": "https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_
CategoryCode_rocketSounding"
          }
        ],
        "scheme": "https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_
CategoryCode"
      }
    ],
```

```
  "formats": [
    {
      "name": "TEXT",
      "mediaType": "text/plain"
    },
    {
      "name": "CSV",
      "mediaType": "text/csv"
    },
    {
      "name": "GML2",
      "mediaType": "text/xml; subtype=gml/2.1.2"
    },
    {
      "name": "GML3",
      "mediaType": "text/xml; subtype=gml/3.1.1"
    },
    {
      "name": "SHAPEFILE",
      "mediaType": "application/vnd.shp"
    },
    {
      "name": "KML",
      "mediaType": "application/vnd.google-earth.kml+xml"
    },
    {
      "name": "KMZ",
      "mediaType": "application/vnd.google-earth.kmz"
    },
    {
      "name": "GeoJSON",
      "mediaType": "application/geo+json"
    },
    {
      "name": "PNG",
      "mediaType": "image/png"
    },
    {
      "name": "JPEG",
      "mediaType": "image/jpeg"
    },
    {
      "name": "GIF",
      "mediaType": "image/gif"
    },
    {
      "name": "PDF",
      "mediaType": "application/x-pdf"
    },
    {
      "name": "SVG",
      "mediaType": "image/svg+xml"
    },
    {
      "name": "TIFF",
      "mediaType": "image/tiff"
    }
  ],
  "license": "other"
},
"linkTemplates": [
  {
    "rel": "describes",
```

```
      "title": "World Ozone and Ultraviolet Radiation Data Centre (WOUDC)
stations",
      "uriTemplate": "https://geo.woudc.org/ows?service=WMS&version=
1.3.0&request=GetMap&crs={crs}&bbox={bbox}&layers=stations&width={width}&height=
{height}&format={format}",
      "variables": {
        "bbox": {
          "description": "...",
          "type": "array",
          "items": {
            "type": "number",
            "format": "double"
          },
          "minItems": 4,
          "maxItems": 4
        },
        "crs": {
          "description": "...",
          "type": "string",
          "enum": [
            "EPSG:4326",
            "EPSG:3857"
          ]
        },
        "width": {
          "description": "...",
          "type": "number",
          "format": "integer",
          "minimum": 600,
          "maximum": 5000
        },
        "height": {
          "description": "...",
          "type": "number",
          "format": "integer",
          "minimum": 600,
          "maximum": 5000
        },
        "format": {
          "type": "string",
          "enum": [
            "application/vnd.google-earth.kml+xml",
            "application/vnd.google-earth.kmz",
            "image/png",
            "image/jpeg",
            "image/gif",
            "image/png; mode=8bit",
            "application/x-pdf",
            "image/svg+xml",
            "image/tiff"
          ]
        }
      }
    },
    {
      "rel": "describes",
      "title": "World Ozone and Ultraviolet Radiation Data Centre (WOUDC)
stations",
      "uriTemplate": "https://geo.woudc.org/ows?service=WFS&version=
1.1.0&request=GetFeature&typeName=woudc:totalozone&maxFeatures=
{maxFeatures}&outputFormat={outputFormat}",
      "variables": {
        "maxFeatures": {
```

```
            "type": "number",
            "default": 10
          },
          "outputFormat": {
            "type": "string",
            "enum": [
              "text/xml; subtype=gml/3.1.1",
              "text/xml; subtype=gml/2.1.2; driver=ogr",
              "application/json; subtype=geojson",
              "application/vnd.google-earth.kml+xml",
              "application/vnd.shp",
              "text/plain",
              "text/csv"
            ],
            "default": "text/xml; subtype=gml/2.1.2; driver=ogr"
          }
        }
      }
    ],
    "links": [
      {
        "rel": "collection",
        "href": "https://woudc.org/data/dataset_info.php"
      },
      {
        "rel": "describes",
        "title": "OGC Web Map Service (WMS)",
        "href": "https://geo.woudc.org/ows?service=WMS&request=GetCapabilities"
      },
      {
        "rel": "describes",
        "title": "OGC Web Feature Service (WFS)",
        "href": "https://geo.woudc.org/ows?service=WFS&request=GetCapabilities"
      },
      {
        "rel": "preview",
        "type": "image/png",
        "title": "Total Ozone Preview Image",
        "href": "https://woudc.org/data/preview.png"
      },
      {
        "rel": "enclosure",
        "type": "text/html",
        "title": "Web Accessible Folder (WAF)",
        "href": "https://woudc.org/archive/Archive-NewFormat/TotalOzone_1.0_1",
        "created": "2015-01-23T00:00:00Z",
        "updated": "2015-01-23T00:00:00Z"
      },
      {
        "rel": "enclosure",
        "type": "application/zip",
        "title": "Static dataset archive file",
        "href": "https://woudc.org/archive/Summaries/dataset-snapshots/totalozone.
zip",
        "created": "2015-01-23T00:00:00Z",
        "updated": "2015-01-23T00:00:00Z"
      },
      {
        "rel": "search",
        "type": "text/html",
        "title": "Data Search / Download User Interface",
        "href": "https://woudc.org/data/explore.php?dataset=totalozone"
      },
```

```
    {
      "rel": "license",
      "href": "https://woudc.org/about/data-policy.php"
    }
  ]
}
```

Listing 22 — Record Example in GeoJSON

7.5.6. Record access

7.5.6.1. Operation

REQUIREMENT 41	
IDENTIFIER	`/req/records-api/record-op`
INCLUDED IN	Requirements class 4: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/records-api`
A	For every record in a catalog the server SHALL support the HTTP GET operation at the `/collection/{collectionId}/items/{recordId}` path.
B	The parameter `collectionId` is each `id` property in the collections response (JSONPath: `$.collections[*].id`) where the value of the `itemType` (JSONPath `$.collections[*].itemType`) is `record`. `id` is a local identifier of the record.

7.5.6.2. Response

REQUIREMENT 42	
IDENTIFIER	`/req/records-api/record-response`
INCLUDED IN	Requirements class 4: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/records-api`

A	The body of the response SHALL contain a record based on the following schema:
	`---`
	`type: object`
	`required:`
	`  - id`
	`  - type`
	`  - geometry`
	`  - properties`
	`properties:`
	`  id:`
	`    type:`
	`      - string`
	`      - integer`
	`    description:`

REQUIREMENT 42

```
        A unique identifier of the catalog record.
    type:
      type: string
      const: Feature
    time:
      oneOf:
        - type: null
        - $ref: 'time.yaml'
    geometry:
      oneOf:
        - type: null
        - $ref: 'https://schemas.opengis.net/ogcapi/features/part1/1.0/openapi/
schemas/geometryGeoJSON.yaml'
    conformsTo:
      type: array
      description:
        The extensions/conformance classes used in this record.
      items:
        type: string
    properties:
      oneOf:
        - type: null
        - allOf:
          - type: object
          - $ref: 'recordCommonProperties.yaml'
    links:
      type: array
      items:
        $ref: 'link.yaml'
    linkTemplates:
      type: array
      items:
        $ref: 'linkTemplate.yaml'
```

B	The `links` section of the record SHALL contain the following links: • a link to this response document (relation: self), • a typed link to the response document in every other media type supported by the service (relation: alternate), and • a link to the record collection that contains this record (relation: collection).
C	All links SHALL include the `rel` parameter.

NOTE: The term “typed link” means a link that specifies a value for the `type` attribute.

7.6. Requirements Class “Sorting” (Common Component)

7.6.1. Overview

REQUIREMENTS CLASS 5	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/sorting`
TARGET TYPE	Document model
CONFORMANCE CLASS	Conformance class A.7: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/sorting`
NORMATIVE STATEMENTS	Requirement 43: `/req/sorting/sortby-param` Requirement 44: `/req/sorting/sortby-response` Requirement 45: `/req/sorting/get-sortables-op` Requirement 46: `/req/sorting/get-sortables-success` Requirement 47: `/req/sorting/defaultSortOrder-definition`

The `Sorting` Requirements Class defines the requirements for specifying how records in a response should be ordered for presentation.

7.6.2. Parameter sortby

REQUIREMENT 43	
IDENTIFIER	`/req/sorting/sortby-param`
INCLUDED IN	Requirements class 5: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/sorting`

A	At the searchable catalog endpoint, the operation SHALL support a parameter `sortby` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: sortby`
	`in: query`
	`required: false`
	`schema:`
	`  type: array`
	`  minItems: 1`
	`  items:`
	`  type: string`
	`  pattern: '[+|-]?[A-Za-z_].*'`
	`explode: false`
	`style: form`

B	The default sort order SHALL be ascending (i.e., +).

NOTE 1: The core definition of the `sortby` parameter only defines a single directive that controls the sort order of the corresponding property. It is anticipated extensions would add additional search facets such as directives for:

1. Handling missing values,

2. Specifying a high value indicating that the corresponding property be sorted as if it were the highest possible value,
3. Specifying a low value indicating that the corresponding property be sorted as if it were the lowest possible value,
4. Allowing records to be omitted from the result set based on their sort order,
5. Specify a fixed value and a fixed value that sorts the corresponding property as if it were the specified fixed value.

REQUIREMENT 44	
IDENTIFIER	`/req/sorting/sortby-response`
INCLUDED IN	Requirements class 5: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/sorting`
A	If the `sortby` parameter is specified, then the records in a response SHALL be ordered by the keys and sort directions (i.e., ascending or descending) specified.
B	The specific set of keys that may be used for sorting SHALL be specified by the `/collections/{collectionId}/sortables` resource.

REQUIREMENT 45	
IDENTIFIER	`/req/sorting/get-sortables-op`
INCLUDED IN	Requirements class 5: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/sorting`
A	For every catalog, the server SHALL support the HTTP GET operation at the sortables endpoint and the media type `application/schema+json`.
B	The sortables resource SHALL be referenced from the catalog with a link with the link relation type `http://www.opengis.net/def/rel/ogc/1.0/sortables`.

REQUIREMENT 46	
IDENTIFIER	`/req/sorting/get-sortables-success`
INCLUDED IN	Requirements class 5: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/sorting`
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.

REQUIREMENT 46	
B	For responses that use `application/schema+json` as the `Content-Type` of the response, the response SHALL have the following characteristics: • The property `$schema` is `http://json-schema.org/draft-07/schema#` or `https://json-schema.org/draft/2019-09/schema`. • The property `$id` is the URI of the resource without query parameters. • The `type` is `object` and each property is a sortable. • The property `$schema` is `https://json-schema.org/draft/2019-09/schema` or `https://json-schema.org/draft/2020-12/schema`. • The property `$id` is the URI of the resource without query parameters. • The `type` is `object` and each property is a queryable.
C	Each property SHALL include a `type` member, except for spatial properties.
D	Each spatial property SHALL not include a `type` or `$ref` member.
E	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 with "any" as the wildcard for any geometry type. In other words, the values for the Simple Feature geometry types are: `geometry-any`, `geometry-point`, `geometry-multipoint`, `geometry-linestring`, `geometry-multilinestring`, `geometry-polygon`, `geometry-multipolygon`, and `geometry-geometrycollection`.
F	If included, the `additionalProperties` member SHALL have a value of `false`.

PERMISSION 14	
IDENTIFIER	`/per/sorting/synthetic`
A	The response MAY include properties that do not actually exist in the information model of the record but are somehow derived or synthesized by the system.

The response is returned as a JSON Schema document that describes a single JSON object where each property is a sortable. Note that the sortables schema does not specify a schema of any object that can be retrieved from a Records API endpoint. JSON Schema is used for the sortables to have a consistent approach for describing schema information and JSON Schema is/will be used in other parts of OGC API — Features suite of standards to describe schemas for GeoJSON feature content including in OpenAPI documents.

NOTE 2: The OGC Features API Standards Working Group has a schema harmonization task that could lead to a future OGC API Standard that will deprecate the approach for the sortables resource specified in this document.

To support clients, providing additional detail about the meaning of the sortables is recommended:

RECOMMENDATION 29	
IDENTIFIER	`/rec/sorting/sortables-schema`
A	Each property SHOULD have a human readable title (`title`) and where necessary a description (`description`).
B	Each property SHOULD have a single type (`type`).

In an OpenAPI 3.0 document, the Sortables resource has the following schema:

```
---
description:
  A list of properties by which the server response may be sorted.
content:
  application/json:
    schema:
      type: object
      description:
         A JSON Schema document that defines all the sortables.
  text/html:
    schema:
      type: string
```

Listing 25 — Schema for the list of sortable properties (Sortables.yaml)

7.6.3. Declaring default sort order

This Specification does not mandate a specific default sort order for records fetched from a searchable catalog. However, servers can declare a default sort order.

REQUIREMENT 47	
IDENTIFIER	`/req/sorting/defaultSortOrder-definition`
INCLUDED IN	Requirements class 5: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/sorting`

A	The information model of a catalog SHALL include the `defaultSortOrder` member with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`---`
	`type: array`
	`items:`
	`  type: object`
	`  required:`
	`  - field`
	`  - direction`
	`  properties:`
	`  field:`
	`  type: string`
	`  direction:`

REQUIREMENT 47

```
                type: string
                enum:
                  - asc
                  - desc
```

B	Each object in the array SHALL include the properties `field` containing the name of the sortable and `direction` indicating the direction of sorting
C	If the value of the `direction` property is `asc` this SHALL indicate that the sort order is ascending.
D	If the value of the `direction` property is `desc` this SHALL indicate that the sort order is descending.
E	If the `defaultSortOrder` member is not present in the response, then no sorting information SHALL be assumed.

7.6.4. Sorting by relevance (informative)

A common practice is to sort search results by relevance.

This capability is accomodated in this Standard by using synthetic sortables. That is, using a sortable that is derived or synthesized by the server.

A server implementing relevance ranking might include, in its list of sortables, a sortable named `relevance` that is dynamically computed for each result in a result set. How exactly the value of the `relevance` score is computed is left to the server's discretion. The value of the synthetic `relevance` sortable could, for example, be between 0 and 10 with zero indicating least relevance and 10 indicating most relevance. A `sortBy` clause specified on a request specifying a descending sort by the `relevance` property would present search results from most to least relevant.

7.6.5. Examples

```
{
  .
  .
  .
  "defaultSortOrder": [
    {
      "field": "updated",
      "direction": "desc"
    },
    {
      "field": "area",
      "direction": "desc"
    }
  ],
  .
  .
  .
```

```
}
```

Listing 27 — Default Sort Order Example

The following examples assume the following set of sortables retrieved from the servers `/collection/{catalogId}/sortables` endpoint:

```
{
  "$schema": "https://json-schema.org/draft/2019-09/schema",
  "$id": "https://data.example.com/collections/abc/sortables",
  "type": "object",
  "properties": {
    "id": {
      "title": "Record Identifier",
      "description": "A unique record identifier assigned by the server.",
      "type": "string"
    },
    "updated": {
      "title": "Updated",
      "description": "The more recent date on which the resource was changed.",
      "type": "string",
      "format": "date"
    },
    "area": {
      "title": "Spatial area of the resource",
      "description": "This is a synthetic sortable that the server computes on
the fly based on the spatial extent of the resource.  Resources without spatial
extent have zero area.",
      "type": "number"
    },
    "relevance": {
      "title": "Relevance of the result",
      "description": "This is a synthetic sortable that the server computes on
the fly to assign a relevance score to each result with respect to the query
parameters specified in the original request.  How exactly the relevance score
is computed is up to the discretion of the server.",
      "type": "number",
      "minimum": 0,
      "maximum": 10
    },
    "extent": {
      "title": "Spatio-Temporal Extent",
      "description": "The spatio-temporal coverage of the resource.",
      "type": "object"
    }
  }
}
```

Listing 28 — List of sortables

```
.../collections/mycat/items?...&sortby=%2Dupdated,%2Bid&...
```

Listing 29 — sortby Example of descending sort by updated date and ascending sort of record identifier.

```
.../collections/mycat/items?...&sortby=%2Bextent&...
```

Listing 30 — sortby Example of an ascending sort by extent (i.e., smallest area first)

```
.../collections/mycat/items?...&sortby=%2Drelevance&...
```

Listing 31 — sortby Example of a descending sort by relevance (i.e., most relevant first)

7.7. Requirements Class "Filtering" (Common Component)

7.7.1. Overview

REQUIREMENTS CLASS 6	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/filtering`
TARGET TYPE	Web API
PREREQUISITE	OGC API — Features — Part 3: Filtering and the Common Query Language (CQL)
NORMATIVE STATEMENTS	Requirement 48: `/req/features-filter/filter-param` Requirement 49: `/req/record-filter/filter-lang-param` Requirement 50: `/req/record-filter/filter-crs-param` Requirement 51: `/req/record-filter/response`

This clause defines the binding to the filtering parameters defined in the OGC API — Features — Part 3: Filtering and the use of the Common Query Language (CQL2) as the query language.

7.7.2. Records filtering

7.7.2.1. Operation

As specified in the Records Access clause, records are accessed using the HTTP GET method via the `/collections/{collectionId}/items` path. The following additional requirements bind the parameters filter, filter-lang and filter-crs to the GET operation on this path.

REQUIREMENT 48	
IDENTIFIER	`/req/features-filter/filter-param`
INCLUDED IN	Requirements class 6: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/filtering`
A	The HTTP GET operation SHALL support the `filter` parameter as defined in Parameter filter.

REQUIREMENT 48	
B	For searchable catalogs, the path for which this parameter is supported SHALL be `/collections/{collectionId}/items`.
C	For local resources catalogs, the path for which this parameter is supported SHALL be whichever path is being extended for catalog-like search capabilities (e.g., `/collections`).

REQUIREMENT 49	
IDENTIFIER	`/req/record-filter/filter-lang-param`
INCLUDED IN	Requirements class 6: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/filtering`
A	The HTTP GET operation SHALL support the `filter-lang` parameter as defined in Parameter filter-lang.
B	For searchable catalogs, the path for which this parameter is supported SHALL be `/collections/{collectionId}/items`.
C	For local resources catalogs, the path for which this parameter is supported SHALL be whichever path is being extended for catalog-like search capabilities (e.g., `/collections`).

RECOMMENDATION 30	
IDENTIFIER	`/rec/record-filtering/cql2-text-encoding`
A	If a filter expression can be represented for its intended use as text, servers SHOULD consider supporting the CQL text encoding.

RECOMMENDATION 31	
IDENTIFIER	`/rec/record-filtering/cql2-JSON-encoding`
A	If a filter expression can be represented for its intended use as JSON, servers SHOULD consider supporting the CQL JSON encoding.

REQUIREMENT 50	
IDENTIFIER	`/req/record-filter/filter-crs-param`
INCLUDED IN	Requirements class 6: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/filtering`

REQUIREMENT 50	
CONDITION	Server implements OGC API — Features — Part 2: Coordinate Reference Systems by Reference
A	The HTTP GET operation SHALL support the `filter-crs` parameter as defined in Parameter filter-crs.
B	For searchable catalogs, the path for which this parameter is supported SHALL be `/collections/{collectionId}/items`.
C	For local resources catalogs, the path for which this parameter is supported SHALL be whichever path is being extended for catalog-like search capabilities (e.g., `/collections`).

7.7.2.2. Response

REQUIREMENT 51	
IDENTIFIER	`/req/record-filter/response`
INCLUDED IN	Requirements class 6: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/filtering`
A	A filter expression SHALL be evaluated for each record of a collection.
B	All other filtering parameters specified (i.e., zero or more of bbox, datetime, q, type, externalIds and any additional query parameters) SHALL be evaluated for each record of a collection.
C	If the filter expression AND all other specified filtering parameters (i.e., zero or more of bbox, datetime, q, type, externalIds and any additional query parameters) evaluate to `TRUE` then the record SHALL be included in the result set.
D	If the filter expression OR any other specified filtering parameter (i.e., zero or more of bbox, datetime, q, type, externalIds and any additional query parameters) evaluates to `FALSE` then the record SHALL be excluded from the result set.

7.7.3. Filtering on links

7.7.3.1. Overview

Links can be added to a catalog record in various ways. This clause describes the various linking patterns and describes how to use CQL to filter based on links.

7.7.3.2. Linking patterns in records

Links can be added to the following sections in a catalog record:

1. the links section,
2. the linkTemplates section,
3. the properties section.

The schema for links added to the links and linkTemplates sections is based on RFC 8288 and is described in the OGC API — Features — Part 1: Core Standard.

Links added directly to the properties section of a catalog record can be added using the following patterns:

1. The link relation is the key for an array of links with a schema similar to that used in the links section but omitting the `rel` key.
2. The link relation is the key for an array of `href` values.
3. The link relation is the key for an an `href` value.

The following examples illustrate each of these linking patterns:

```
{
   ...,
   "properties": {
      ...,
      "links":
      [
         ...,
         {
            "rel" : "alternate",
            "title" : "This records as XML.",
            "href" : "https://example.org/collections/mycat/items/rec01.xml"
         },
         {
            "rel": "next",
            "title": "The next record in this result set.",
            "href" : "https://example.org/collections/mycat/items/rec02"
         },
         ...
      ],
      ...
   }
}
```

Listing 32 — Adding links to the links section

```
{
```

```
  ...,
  "properties": {
    ...,
    "related": [
      {
        "title" : "A related record",
        "href" : "https://example.org/related-record-path/rec15"
      }
    ],
    "license": {
      "title" : "CC BY 2.0",
      "href" : "https://creativecommons.org/licenses/by/2.0/"
    },
    "enclosure": {
      "title": "GeoPackage of Feature Data",
       "href" : "https:/example.org/collections/MyCollection/items?f=application/
geopackage+vnd.sqlite3&bbox={bbox}",
       "varBase": "https://example.org/variables/"
    },
    ...
  }
}
```

Listing 33 — Adding links to the properties section

```
{
   ...,
   "properties": {
      ...,
      "related": ["https://example.org/related-record-path/rec15"],
      "license": "https://creativecommons.org/licenses/by/2.0/",
      ...
   }
}
```

Listing 34 — Adding links to the properties section

7.7.3.3. Examples

7.7.3.3.1. CQL filters on links

The following example illustrates how to filter links that may appear in a catalog record.

```
links[rel='license'].title LIKE 'CC%'
```

Listing 35 — Filtering links in the links section

```
properties.related[*].href LIKE '%/rec15' AND properties.license.title LIKE 'CC%'
```

Listing 36 — Filtering links in the properties section

```
properties.related LIKE '%/rec15' AND properties.license LIKE 'https://
creativecommons.org/licenses/%'
```

Listing 37 — Filtering links in the properties section

7.7.3.3.2. Filtering by keyword from a controlled vocabulary

The following example illustrates how to use OGC API — Feature — Part 3: Filtering and CQL2 to query by keywords from a specific controlled vocabulary.

The use of keywords from a controlled vocabulary is encoded in a record using the `themes` property. Consider a record that has the following `themes` property:

```
   "themes": [
     {
       "concepts": [
         "dobson",
         "brewer",
         "vassey",
         "pion",
         "microtops",
         "spectral",
         "hoelper",
         "saoz",
         "filter"
       ],
       "scheme": "https://geo.woudc.org/codelists.xml#WOUDC_InstrumentCode"
     },
     {
       "concepts": [
         "atmosphericComposition",
         "pollution",
         "observationPlatform",
         "rocketSounding"
       ],
       "scheme": "https://wis.wmo.int/2012/codelists/WMOCodeLists.xml#WMO_
CategoryCode"
     }
   ],
```

Listing 38

Using an implementation of the OGC API — Feature — Part 3: Filtering Standard, a server would need to expose two queryables, `scheme` and `concept` for the purpose of filtering using keywords from a specific controlled vocabulary. Using these queryables, the following filter could be expressed:

```
https://example.org/collections/myCatalog/items?filter-lang=cql2-text&filter=
scheme%3D%27https%3A%2F%2Fgeo.woudc.org%2Fcodelists.xml%23WOUDC_InstrumentCode%27
%20and%20concept%3D%27spectral%27%2C%0Ahoelper%27
```

Listing 39

An alternative approach that does rely on using OGC API — Feature — Part 3: Filtering and CQL2 would rely on the server defining a set of query parameters in the server's API description document, again named `scheme` and `concept`, that would allow the following filter to be expressed:

```
https://example.org/collections/myCatalog/items?scheme=https%3A%2F%2Fgeo.woudc.
org%2Fcodelists.xml%23WOUDC_InstrumentCode&concept=spectral,hoelper
```

Listing 40

7.8. Requirements Class "Autodiscovery" (Common Component)

7.8.1. Overview

REQUIREMENTS CLASS 7	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/autodiscovery`
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.11: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/autodiscovery`
PREREQUISITE	Record Collection
NORMATIVE STATEMENT	Requirement 52: `/req/autodiscovery/links`

The purpose of autodiscovery is, knowing the location of a web page, to find the addresses(s) of that page's associated catalog(s). For example, a client could retrieve the landing page of an OGC API deployment, find the location of the site's searchable catalog by locating the link relation type that has the value `http://www.opengis.net/def/rel/ogc/1.0/ogc-catalog` in the landing page and then, using that catalog, search for resources offered by the site.

7.8.2. Autodiscovery links

REQUIREMENT 52	
IDENTIFIER	`/req/autodiscovery/links`
INCLUDED IN	Requirements class 7: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/autodiscovery`

REQUIREMENT 52	
A	A web page SHALL include one or more links (relation: `http://www.opengis.net/def/rel/ogc/1.0/ogc-catalog`) that point to a crawlable catalog, searchable catalogs or local resource catalog associated with the page.
B	The value of the `href` member SHALL point to the access endpoint of the corresponding catalog.

NOTE:

- For a crawlable catalog the access endpoint is the URL of the catalog.
- For a searchable catalog the access endpoint is the URL of the `/collections/{catalogId}/items` endpoint.
- For a local resource catalog the access endpoint is the URL of the local resources endpoint (e.g., `/processes`).

7.9. Encodings

7.9.1. Overview

This Standard defines two requirements classes, HTML and JSON for encoding resources defined in this Standard.

HTML is the core language of the World Wide Web. An API that supports HTML supports browsing resources with a web browser and also enables search engines to crawl and index those resources.

JSON is a language-independent file format that uses human-readable text to store and transmit data objects consisting of key-value pairs and arrays. It is a commonly used data format in machine-to-machine communication including that between web applications and servers. It is well supported by numerous tools and software libraries.

"GeoJSON is a geospatial data interchange format based on JavaScript Object Notation (JSON). It defines several types of JSON objects and the way they are combined to represent data about geographic features, their properties, and their spatial extents. GeoJSON uses a geographic coordinate reference system, World Geodetic System 1984, and units of decimal degrees."
IETF RFC 7946

GeoJSON provides a simple way of representing records in JSON and is supported by numerous geospatial tools and software libraries. It is best used for content which has a spatial extent that can be used with the World Geodetic System 1984 Coordinate Reference System.

7.9.2. Requirements Class JSON (Common Component)

7.9.2.1. Overview

REQUIREMENTS CLASS 8	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/json`
TARGET TYPE	Document encoding
CONFORMANCE CLASS	Conformance class A.9: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/json`
PREREQUISITES	IETF RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format JSON Schema GeoJSON
NORMATIVE STATEMENTS	Requirement 58: `/req/json/catalog-content` Requirement 57: `/req/json/collection-response` Requirement 53: `/req/json/conformance` Requirement 55: `/req/json/record-content` Requirement 54: `/req/json/record-response`

REQUIREMENT 53	
IDENTIFIER	`/req/json/conformance`
INCLUDED IN	Requirements class 8: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/json`
A	The `conformsTo` array SHALL contain the value `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/json`.

NOTE: The location of the `conformsTo` array depends on the type of catalog being deployed. In the case of the searchable or local resources catalog, the `conformsTo` array can be found in the conformance page (path: `/conformance`) of the OGC API deployment or in the catalog. In the case of a crawlable catalog, the `conformsTo` array is found in the catalog.

7.9.2.2. Record encoding

GeoJSON is a commonly used format based on JSON that is simple to understand and well supported by tools and software libraries. Since most Web developers are comfortable with

using a JSON-based format, supporting GeoJSON is recommended. The caveat is, if record data can be represented in GeoJSON for the intended use.

The following requirements apply when:

1. The Records API endpoint advertises conformance to the JSON Conformance Class; and
2. The client negotiates that the content of the server's response be GeoJSON.

REQUIREMENT 54	
IDENTIFIER	`/req/json/record-response`
INCLUDED IN	Requirements class 8: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/json`
A	200-responses of the server SHALL support the following media type: • `application/geo+json` for resources that include record content.
B	The `geometry` member SHALL be present and SHALL store a geometry, for the purpose of discovery, of the resource being described by the record. This can be the actual geometry of the resource or some other representation of the space that the resource occupies.
C	The `extent` member SHALL include the bounding extent of the resource being described by the record. If the server chooses to use the bounding box of the resource as the `geometry` of the resource, then the `geometry` and `extent` values may be the same.

REQUIREMENT 55	
IDENTIFIER	`/req/json/record-content`
INCLUDED IN	Requirements class 8: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/json`
A	Every 200-response with the media type `application/geo+json` SHALL be • a GeoJSON FeatureCollection Object for records, and • a GeoJSON Feature Object for a single record.
B	The schema of records in all responses with the media type `application/geo+json` SHALL validate against the OpenAPI 3.0 schema document `recordGeoJSON.yaml`.
C	The links specified in the requirements /req/core/rc-links and /req/core/record-links SHALL be added in an extension property (foreign member) with the name `links`.
D	The response SHALL include, in the `links` section, a profile link (href: `http://www.opengis.net/def/profile/OGC/0/ogc-catalog`, relation: `profile`).

REQUIREMENT 56

IDENTIFIER	`/req/json/record-content-profile`
CONDITION	The negotiated profile is `http://www.opengis.net/def/profile/OGC/0/ogc-record`.
A	Every 200-response with the media type `application/geo+json` SHALL be • a GeoJSON FeatureCollection Object for records, and • a GeoJSON Feature Object for a single record.
B	The schema of records in all responses SHALL validate against the OpenAPI 3.0 schema document `recordGeoJSON.yaml`.
C	The links specified in the requirements /req/core/rc-links and /req/core/record-links SHALL be added in an extension property (foreign member) with the name `links`.
D	The response SHALL include, in the `links` section, a profile link (href: `http://www.opengis.net/def/profile/OGC/0/ogc-catalog`, relation: `profile`).

NOTE 1: It should be noted that the `recordGeoJSON.yaml` schema makes the `geometry` member mandatory because the GeoJSON RFC requires that the `geometry` member be present. If a resource described by a JSON-encoded record does not have any spatial characteristics, the values of the `geometry` member can be `null` as per the GeoJSON RFC.

RECOMMENDATION 32

IDENTIFIER	`/rec/json/header-profile-link`
A	A server SHOULD include a profile link (href: `http://www.opengis.net/def/profile/OGC/0/ogc-catalog`, relation: `profile`) in the response HTTP headers (see RFC 6906, The 'profile' Link Relation Type).

```
---
type: object
required:
  - id
  - type
  - geometry
  - properties
properties:
  id:
    type:
      - string
      - integer
    description:
      A unique identifier of the catalog record.
  type:
    type: string
    const: Feature
  time:
    oneOf:
```

```
      - type: null
      - $ref: 'time.yaml'
  geometry:
    oneOf:
      - type: null
      - $ref: 'https://schemas.opengis.net/ogcapi/features/part1/1.0/openapi/
schemas/geometryGeoJSON.yaml'
  conformsTo:
    type: array
    description:
      The extensions/conformance classes used in this record.
    items:
      type: string
  properties:
    oneOf:
      - type: null
      - allOf:
        - type: object
        - $ref: 'recordCommonProperties.yaml'
  links:
    type: array
    items:
      $ref: 'link.yaml'
  linkTemplates:
    type: array
    items:
      $ref: 'linkTemplate.yaml'
```

Listing 41 — Schema for a catalog record encoded as GeoJSON (recordGeoJSON.yaml)

```
---
type: object
properties:
  created:
    type: string
    description:
      The date this record was created in the server.
    format: date-time
  updated:
    type: string
    description:
      The most recent date on which the record was changed.
    format: date-time
  type:
    type: string
    description:
      The nature or genre of the resource. The value
      should be a code, convenient for filtering
      records. Where available, a link to the canonical
      URI of the record type resource will be added to
      the 'links' property.
  title:
    type: string
    description:
      A human-readable name given to the resource.
  description:
    type: string
    description:
      A free-text account of the resource.
  keywords:
    type: array
```

```
  description:
    The topic or topics of the resource. Typically
    represented using free-form keywords, tags, key
    phrases, or classification codes.
  items:
    type: string
themes:
  type: array
  description:
    A knowledge organization system used to classify
    the resource.
  minItems: 1
  items:
    $ref: 'theme.yaml'
language:
  description:
    The language used for textual values in this
    record representation.
  $ref: 'language.yaml'
languages:
  type: array
  description:
    This list of languages in which this record is
    available.
  items:
    $ref: 'language.yaml'
resourceLanguages:
  type: array
  description:
    The list of languages in which the resource
    described by this record is available.
  items:
    $ref: 'language.yaml'
externalIds:
  type: array
  description:
    An identifier for the resource assigned by an
    external (to the catalog) entity.
  items:
    type: object
    properties:
      scheme:
        type: string
        description:
          A reference to an authority or identifier
          for a knowledge organization system from
          which the external identifier was obtained.
          It is recommended that the identifier be a
          resolvable URI.
      value:
        type: string
        description: The value of the identifier.
    required:
      - value
formats:
  type: array
  description:
    A list of available distributions of the resource.
  items:
    $ref: 'format.yaml'
contacts:
  type: array
  description:
```

```
      A list of contacts qualified by their role(s) in
      association to the record or the resource described
      by the record.
    items:
      $ref: 'contact.yaml'
  license:
    $ref: 'license.yaml'
  rights:
    type: string
    description:
      A statement that concerns all rights not addressed
      by the license such as a copyright statement.
```

Listing 42 — Common record properties schema (recordCommonProperties.yaml)

NOTE 2: Referenced schemas can be found at:

- time.yaml,
- geometryGeoJSON.yaml,
- language.yaml,
- theme.yaml,
- format.yaml,
- contact.yaml,
- license.yaml,
- linkBase.yaml.

PERMISSION 15	
IDENTIFIER	`/per/json/time`
A	In the GeoJSON encoding of a record, the `time` member is a top-level field. Since some popular Geo JSON tools only manipulate record properties encoded in the `properties` section of a GeoJSON record, implementations MAY duplicate the `time` member from the top level into the `properties` section to take advantage of these available tools.

NOTE 3: This standard does not define a normative schema for a record encoding that is not GeoJSON. However, this informative schema is offered as an example of what a plain JSON record schema might look like.

7.9.2.3. Catalog encoding

JSON is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays (or other serializable values). JSON is a commonly used data encoding with diverse uses in electronic data interchange, including that of web applications with servers. Therefore, the recommendation is

to use the JSON encoding if the catalog (i.e., a collection of record) data can be represented in JSON for the intended use.

The following requirements apply when:

1. The Records API endpoint advertises conformance to the JSON Conformance Class; and
2. The client negotiates that the content of the server's response be JSON.

REQUIREMENT 57	
IDENTIFIER	`/req/json/collection-response`
INCLUDED IN	Requirements class 8: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/json`
A	200-responses of the server SHALL support the following media type: • `application/ogc-catalog+json` for catalog or record collection resources.

REQUIREMENT 58	
IDENTIFIER	`/req/json/catalog-content`
INCLUDED IN	Requirements class 8: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/json`
A	The schema of all responses with the media type `application/ogc-catalog+json` SHALL validate against the OpenAPI 3.0 schema document `catalog.yaml`.

REQUIREMENT 59	
IDENTIFIER	`/req/json/catalog-content-profile`
CONDITION	The negotiated profile is `http://www.opengis.net/def/profile/OGC/0/ogc-catalog`.
A	The schema of all responses with the media type `application/json` SHALL validate against the OpenAPI 3.0 schema document `catalog.yaml`.
B	The response SHALL include, in the `links` section, a profile link (href: `http://www.opengis.net/def/profile/OGC/0/ogc-catalog`, relation: `profile`).

See also recommendation: /rec/json/header-profile-link.

```
---
allOf:
```

```
  - $ref: 'https://schemas.opengis.net/ogcapi/features/part1/1.0/openapi/schemas/
collection.yaml'
  - $ref: 'catalogCommonProperties.yaml'
  - type: object
    properties:
      itemType:
        description:
          If this catalog is a homogenous collection
          of records then itemType is a string of fixed
          value of record.
          If this catalog is a homogenous collection
          of other catalogs then itemType is a string of
          fixed value of catalog.
          If this catalog is a heterogenous collection
          of records and catalogs then itemType is a array
          indicated that item types of the members of this
          collections (i.e. record and/or catalog).
        oneOf:
          - type: string
            enum:
              - record
              - catalog
          - type: array
            items:
              type: string
              enum:
                - record
                - catalog
```

Listing 43 — Schema for a catalog encoded as JSON(catalog.yaml)

NOTE: Referenced schemas can be found at:

- collection.yaml,
- language.yaml,
- theme.yaml,
- contact.yaml,
- license.yaml.

7.9.3. Requirements Class HTML (Common Component)

REQUIREMENTS CLASS 9	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/html`
TARGET TYPE	Document encoding
CONFORMANCE CLASS	Conformance class A.8: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/html`

REQUIREMENTS CLASS 9	
PREREQUISITE	http://www.w3.org/TR/html5/
NORMATIVE STATEMENTS	Requirement 60: `/req/html/conformance` Requirement 61: `/req/html/definition` Requirement 62: `/req/html/content`

REQUIREMENT 60	
IDENTIFIER	`/req/html/conformance`
INCLUDED IN	Requirements class 9: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/html`
A	The `conformsTo` array SHALL contain the value `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/html`.

NOTE: The location of the `conformsTo` array depends on the type of catalog being deployed. In the case of the searchable or local resources catalog, the `conformsTo` array can be found in the conformance page (path: `/conformance`) of the OGC API deployment or in the catalog. In the case of a crawlable catalog, the `conformsTo` array is found in the catalog.

Catalog information that is only accessible in formats primarily intended to be read by machines, such as JSON or XML, have two issues:

- the data is not discoverable using the most common mechanism for discovering information, the Web search engines; and
- the data cannot be viewed directly in a browser (additional tools are required to view the data).

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

This is discussed in detail in Best Practice 2: Make your spatial data indexable by search engines SDWBP. The Records API Standard therefore recommends supporting HTML as an encoding.

REQUIREMENT 61	
IDENTIFIER	`/req/html/definition`
INCLUDED IN	Requirements class 9: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/html`

REQUIREMENT 61	
A	Every `200` response of an operation of the server SHALL support the media type `text/html`.

REQUIREMENT 62	
IDENTIFIER	`/req/html/content`
INCLUDED IN	Requirements class 9: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/html`
A	Every `200` response of the server with the media type `text/html` SHALL be a HTML 5 document that includes the following information in the HTML body: • All information identified in the schemas of the Response Object in the HTML `<body>`, and • All links in HTML `<a>` elements in the HTML `<body>`.

RECOMMENDATION 33	
IDENTIFIER	`/rec/html/schema-org`
A	A `200` response with the media type `text/html`, SHOULD include Schema.org annotations.

7.10. Requirements Class “OpenAPI 3.0” (Common Component)

7.10.1. Basic requirements

REQUIREMENTS CLASS 10	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/oas30`
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.10: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/oas30`
PREREQUISITE	OpenAPI Specification 3.0

REQUIREMENTS CLASS 10	
NORMATIVE STATEMENTS	Requirement 63: `/req/oas30/oas-common` Requirement 64: `/req/oas30/conformance`

Servers conforming to this requirements class define their API using an OpenAPI Document.

REQUIREMENT 63	
IDENTIFIER	`/req/oas30/oas-common`
INCLUDED IN	Requirements class 10: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/oas30`
A	The API implementation SHALL demonstrate conformance to the OpenAPI 3.0 requirements class of the OGC API — Feature — Part 1: Core Standard.

REQUIREMENT 64	
IDENTIFIER	`/req/oas30/conformance`
INCLUDED IN	Requirements class 10: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/oas30`
A	The `conformsTo` array found in the conformance path (path: `/conformance`) SHALL contain the value `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/oas30`.

8

CATALOG DEPLOYMENTS

8 CATALOG DEPLOYMENTS

8.1. Overview

This clause uses the common components defined in this Standard to define various catalog deployments.

8.2. Requirements Class “Crawlable Catalog” (Deployment)

8.2.1. Overview

REQUIREMENTS CLASS 11	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/crawlable-catalog`
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.12: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/crawlable-catalog`
PREREQUISITES	Record Core Record Collection
NORMATIVE STATEMENTS	Requirement 65: `/req/crawlable-catalog/record` Requirement 66: `/req/crawlable-catalog/record-location` Requirement 67: `/req/crawlable-catalog/catalog` Requirement 68: `/req/crawlable-catalog/location` Requirement 69: `/req/crawlable-catalog/conformance`

The `Crawlable Catalog` Requirements Class defines the requirements for a catalog composed of static, web-accessible files.

The goal of a crawlable catalog is to expose metadata about resources online with as low an entry barrier as possible, thus making those resources more easily discoverable.

A crawlable catalog is a deployment pattern that uses the collection and record common components defined in this Standard. It is simply a set of files deployed on the web, usually

encoded as HTML or JSON, that conform to the collection and record schemas defined in this Standard. The files contain embedded links to enable navigation from one to another. Any HTTP server or cloud storage service, for example, could be used to expose the files of a crawlable catalog.

A crawlable catalog is not searchable and does not respond to query requests. As the name implies, it can only be crawled (or harvested) by following the embedded links, typically by search engines and other active catalogs or browsed using a web browser. However, due to its simplicity, a crawlable catalog is easy to deploy and maintain, and is very reliable.

NOTE: It should be pointed out that a searchable catalog can also behave as a crawlable catalog as long as the requirements in this clause are satisfied by the searchable catalog implementation.

8.2.2. Making a resource discoverable

REQUIREMENT 65	
IDENTIFIER	`/req/crawlable-catalog/record`
INCLUDED IN	Requirements class 11: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/crawlable-catalog`
A	A record SHALL be created for each resource to be made discoverable.

Attention is drawn to requirement /per/record-core/additional-properties-record that allows the information content of the record to be enriched as required to describe the resource being made discoverable.

REQUIREMENT 66	
IDENTIFIER	`/req/crawlable-catalog/record-location`
INCLUDED IN	Requirements class 11: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/crawlable-catalog`
A	The record file SHALL be placed in a web-accessible location.

NOTE: This Standard is primarily concerned with making resources discoverable on the open Web that sits on top of the public Internet. However, the same requirement applies in more restricted contexts such as a company Intranet or within classified networks. In other words, the record should be placed in a location that is accessible to the intended audience.

RECOMMENDATION 34	
IDENTIFIER	`/rec/crawlable-catalog/record-file-name`
A	The name of the record file SHOULD be `{id}.json` where `{id}` is the identifier of the record.

RECOMMENDATION 35	
IDENTIFIER	`/rec/crawlable-catalog/record-file-location`
A	The record file SHOULD be located close to the resource(s) that the record describes.

For example, the record file could be placed in the same web-accessible directory or object store as the resource(s) that the record describes.

8.2.3. Grouping a collection of records into a catalog

REQUIREMENT 67	
IDENTIFIER	`/req/crawlable-catalog/catalog`
INCLUDED IN	Requirements class 11: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/crawlable-catalog`
A	A collection of related records, called a *catalog*, SHALL be described by a Record Collection object.

REQUIREMENT 68	
IDENTIFIER	`/req/crawlable-catalog/location`
INCLUDED IN	Requirements class 11: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/crawlable-catalog`
A	The catalog SHALL be placed in a web-accessible location.

NOTE: This Standard is primarily concerned with making resources discoverable on the open Web that sits on top of the public Internet. However, the same requirement applies in more restricted contexts such as a company Intranet or within classified networks. In other words, the catalog should be placed in a location that is accessible to the intended audience.

REQUIREMENT 69	
IDENTIFIER	/req/crawlable-catalog/conformance
INCLUDED IN	Requirements class 11: http://www.opengis.net/spec/ogcapi-records-1/1.0/req/crawlable-catalog
A	The conformsTo array found in the catalog SHALL contain the value http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/crawlable-catalog.

8.2.4. Organizing crawlable catalogs

In order to provide flexibility with regard to how crawlable catalogs can be organized (e.g., series, hierarchy, etc.) a catalog can also provide information about and access to related catalogs. As such, the Record Collection object can be used as a common entry point for crawling to multiple related catalogs, sub-catalogs, records and other resources.

The relationships among catalogs and records are expressed as links in the links section of the catalog and the links section of the record.

See “Other catalogs” and Records.

8.3. Requirements Class “Searchable Catalog” (Deployment)

REQUIREMENTS CLASS 12	
IDENTIFIER	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.17: http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog
PREREQUISITE	Records API
NORMATIVE STATEMENTS	Requirement 70: /req/searchable-catalog/common Requirement 71: /req/searchable-catalog/conformance Requirement 72: /req/searchable-catalog/mandatory-catalog-properties Requirement 73: /req/searchable-catalog/mandatory-record-properties

8.3.1. Overview

The Searchable Catalog Requirements Class defines the requirements for a catalog that is searchable through an API. A searchable catalog is an implementation of the OGC API — Features — Part 1: Core Standard that uses a defined information model. In the case of the OGC API — Records Standard, that information model is the Record.

REQUIREMENT 70	
IDENTIFIER	`/req/searchable-catalog/common`
INCLUDED IN	Requirements class 12: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog`
STATEMENT	Implementations of this requirements class SHALL implement the following common components:
A	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/records-api
B	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core
C	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection
D	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters

PERMISSION 16	
IDENTIFIER	`/per/searchable-catalog/additional-conformance`
STATEMENT	Implementations of this conformance class MAY, additionally, implement the following common components:
A	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/oas30
B	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/json
C	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/html
D	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/autodiscover

REQUIREMENT 71

IDENTIFIER	`/req/searchable-catalog/conformance`
INCLUDED IN	Requirements class 12: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog`
A	A server's conformance declaration SHALL be retrieved from the `/conformance` endpoint as defined in the Declaration of conformance classes clause of the OGC API — Features — Part 1: Core Standard.
B	An implementation of a searchable catalog SHALL declare the following conformance classes in the conformance declaration: • http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/core • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog
C	A server that supports JSON responses SHALL declare the following conformance classes in the conformance declaration: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/json
D	A server that supports HTML responses SHALL declare the following conformance classes in the conformance declaration: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/html
E	A server that supports a server description document using OpenAPI 3.0 SHALL declare the following conformance classes in the conformance declaration: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/oas30
F	A server that supports autodiscovery SHALL declare the following conformance classes in the conformance declaration: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/autodiscovery

REQUIREMENT 72

IDENTIFIER	`/req/searchable-catalog/mandatory-catalog-properties`
INCLUDED IN	Requirements class 12: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog`
A	The information model describing a searchable catalog SHALL contain all the mandatory properties listed in Table 11.

PERMISSION 17

IDENTIFIER	`/per/searchable-catalog/additional-catalog-properties`
A	A catalog object MAY include zero or more of the optional properties listed in Table 11.
B	A catalog object MAY include any number of additional properties not listed in Table 11. The meaning of these additional properties is not specified in this document.

REQUIREMENT 73	
IDENTIFIER	/req/searchable-catalog/mandatory-record-properties
INCLUDED IN	Requirements class 12: http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog
A	Each record accessibles via a searchable catalog SHALL include all the mandatory properties listed in Table 8 and Table 9.

PERMISSION 18	
IDENTIFIER	/per/searchable-catalog/additional-record-properties
A	Each record accessible via a searchable catalog MAY include zero or more of the optional properties listed in Table 8 and Table 9.
B	Each record accessible via a searchable catalog MAY include any number of additional properties not listed in Table 8 and Table 9. The meaning of these additional properties is not specified in this document.

8.3.2. Requirements Class "Searchable Catalog — Filtering" (Deployment)

REQUIREMENTS CLASS 13	
IDENTIFIER	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog/filtering
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.18: http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog/filtering
PREREQUISITES	Searchable Catalog Filtering
NORMATIVE STATEMENTS	Requirement 74: /req/searchable-catalog/filtering Requirement 75: /req/searchable-catalog/filtering-conformance Requirement 76: /req/searchable-catalog/mandatory-queryables

This requirements class specifies requirements and recommendations for supporting advanced filtering at searchable catalog endpoints. Advanced filter expressions can only reference properties from an advertised set of queryables.

REQUIREMENT 74

IDENTIFIER	`/req/searchable-catalog/filtering`
INCLUDED IN	Requirements class 13: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog/filtering`
STATEMENT	Implementations that offer enhanced filtering SHALL implement the following requirements class at the searchable catalog enpoint of `/collections/{collectionId}/items` where the `itemType` property of the collection with id `{collectionId}` (JSONPath: `$.collections[*].itemType`) is a string with the value `record`.
A	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/filtering

REQUIREMENT 75

IDENTIFIER	`/req/searchable-catalog/filtering-conformance`
INCLUDED IN	Requirements class 13: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog/filtering`
A	A server's conformance declaration SHALL be retrieved from the `/conformance` endpoint as defined in the Declaration of conformance classes clause of the OGC API — Features — Part 1: Core Standard.
B	An implementation of this requirements class SHALL declare the following conformance classes in the conformance declaration: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog-filtering

REQUIREMENT 76

IDENTIFIER	`/req/searchable-catalog/mandatory-queryables`
INCLUDED IN	Requirements class 13: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog/filtering`
A	The list of queryables SHALL include all the mandatory properties listed in Table 8 and Table 9.

PERMISSION 19

IDENTIFIER	`/per/searchable-catalog/filtering/additional-queryables`
STATEMENT	The list of queryables MAY also include:

PERMISSION 19	
A	Zero or more of the optional properties listed in Table 8 and Table 9,
B	Any number of additional properties that are part of the record but are not listed in Table 8 and Table 9.

8.3.3. Requirements Class "Searchable Catalog — Sorting" (Deployment)

REQUIREMENTS CLASS 14	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog/sorting`
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.19: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog/sorting`
PREREQUISITES	Searchable Catalog Sorting
NORMATIVE STATEMENTS	Requirement 77: `/req/searchable-catalog/sorting` Requirement 78: `/req/searchable-catalog/sorting-conformance`

The Searchable Catalog — Sorting requirements class specifies requirements for supporting sorting of responses at searchable catalog endpoints.

REQUIREMENT 77	
IDENTIFIER	`/req/searchable-catalog/sorting`
INCLUDED IN	Requirements class 14: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog/sorting`
STATEMENT	Implementations of sorting of record responses SHALL implement the following requirements class at the searchable catalog endpoint of `/collections/{collectionId}/items` where the `itemType` property of the collection with id `{collectionId}` (JSONPath: `$.collections[*].itemType`) is a string with the value `record`.
A	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/sorting

REQUIREMENT 78	
IDENTIFIER	/req/searchable-catalog/sorting-conformance
INCLUDED IN	Requirements class 14: http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog/sorting
A	A server's conformance declaration SHALL be retrieved from the /conformance endpoint as defined in the Declaration of conformance classes clause of the OGC API — Features — Part 1: Core Standard.
B	An implementation of this requirements class SHALL declare the following conformance classes in the conformance declaration: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog-sorting

8.4. Requirements Class "Local Resources Catalog" (Deployment)

8.4.1. Overview

REQUIREMENTS CLASS 15	
IDENTIFIER	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.13: http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog
PREREQUISITES	Record Core Record Collection http://www.opengis.net/spec/ogcapi-records-1/1.0/req/autodiscovery
NORMATIVE STATEMENTS	Requirement 79: /req/local-resources-catalog/common Requirement 80: /req/local-resources-catalog/conformance Requirement 81: /req/local-resources-catalog/mandatory-catalog-properties Requirement 82: /req/local-resources-catalog/mandatory-record-properties Requirement 83: /req/local-resources-catalog/discovery

The set of OGC API Standards define a number of resource endpoints from which summary information about resources accessible via the server's API can be retrieved. These endpoints

are referred to as *local resources catalogs* in this Standard because they provide metadata about resources offered locally by a deployment of some or all of the set of OGC API Standards.

NOTE:

- ***local_resources_endpoint***
  Refers to the path from which a catalog of local resources can be retrieved.

- ***local_resources_catalog***
  Refers to an instance of the resource that can be retrieved from the *local resources endpoint* (i.e., the catalog).

- ***local_resources_item***
  Each local resources catalog includes an array member. The items of that array are objects (i.e., records) that contain summary metadata about resources retrievable via the server's API. These objects are referred to as ***local resources items***.

The `/collections` endpoint is an example of a *local resources endpoint*. Doing a GET operation at the local resources endpoint (i.e., `GET /collections`) returns a document that is an example of a *local resources catalog*. This document contains an array member named `collections` and the items of that array are the *local resources items*. Each of these *local resources items* provides metadata about some resource (e.g., a feature collection) that is accessible via the server's API.

Other *local resources endpoints* available in the OGC API resource tree include the `/processes` endpoint which provides summary metadata about processes and the `/collections/{collectionId}/scenes` endpoint which provides metadata about the source images of a coverage.

In order to enable catalog-like searching at these local resources endpoints, this requirements class:

1. extends the information content of the local resources catalog to satisfy the requirements of the Record Collection common component;

2. extends the information content of the local resources i.e., to satisfy the requirements of the Record Core common component; and

3. optionally extends the local resources endpoint with a set of query parameters that support a basic capability.

The Local Resources Catalog requirements class does not define the specific endpoints in the OGC API resource tree that should be enabled with catalog-like query capabilities. It only defines how those endpoints should be enhanced should the requirement for catalog-like searching be identified by the responsible OGC Standard. It is anticipated that the responsible OGC standard will define a requirements class or extension to enable local resources catalog capabilities at that endpoint.

8.4.2. Common components

REQUIREMENT 79	
IDENTIFIER	`/req/local-resources-catalog/common`
INCLUDED IN	Requirements class 15: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog`
STATEMENT	Implementations of this requirements class SHALL implement the following common components:
A	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core
B	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection
C	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/autodiscovery

8.4.3. Declaring conformance

REQUIREMENT 80	
IDENTIFIER	`/req/local-resources-catalog/conformance`
INCLUDED IN	Requirements class 15: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog`
A	A server's conformance declaration SHALL be retrieved from the `/conformance` endpoint as defined in the Declaration of conformance classes clause of the OGC API — Features — Part 1: Core Standard.
B	An implementation of a local resources catalog SHALL declare the following conformance classes in the conformance declaration: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog
C	A server that supports JSON responses SHALL declare the following conformance classes in the conformance declaration: • http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/json
D	A server that supports HTML responses SHALL declare the following conformance classes in the conformance declaration: • http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/html

8.4.4. Extending the local resources catalog

As an example of a local resources endpoint, consider the `/collections` endpoint. The resource, called *collections*, that can be retrieved from this endpoint is defined in OGC API — Features. The following table cross-walks the information content of the collections object with the catalog.

Table 14 — Crosswalk of collections object and catalog

COLLECTIONS PROPERTY	REQUIREMENT	CATALOG PROPERTY	REQUIREMENT
—	—	id	**required**
—	—	title	optional
—	—	description	optional
—	—	links	optional
—	—	extent	optional
—	—	itemType	optional
—	—	crs	optional
—	—	created	optional
—	—	updated	optional
—	—	type	**required**
—	—	keywords	optional
—	—	themes	optional
—	—	language	optional
—	—	languages	optional
—	—	resourceLanguages	optional
—	—	externalIds	optional
—	—	formats	optional
—	—	contacts	optional
—	—	license	optional
—	—	rights	optional
—	—	conformsTo	optional

COLLECTIONS PROPERTY	REQUIREMENT	CATALOG PROPERTY	REQUIREMENT
—	—	recordsArrayName	optional
collections	**required**	records	optional
links	**required**	links	**required**
linkTemplates	optional	linkTemplates	optional

The information content of the *collections* object, a local resources catalog, is sparse and so the following requirements and recommendations are made:

REQUIREMENT 81	
IDENTIFIER	`/req/local-resources-catalog/mandatory-catalog-properties`
INCLUDED IN	Requirements class 15: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog`
A	The information model of the local resources catalog SHALL contain all the mandatory properties listed in Table 11.

PERMISSION 20	
IDENTIFIER	`/per/local-resources-catalog/additional-catalog-properties`
A	Table 15 contains a list of recommended additional properties that MAY be added to the information model of a local resources catalog.

Table 15 — Additional properties for a local resources object

PROPERTY	DESCRIPTION
conformsTo	The extensions/conformance classes used in this local resources object.
created	Date of creation of this local resources object.
updated	The most recent date on which this local resources object was changed.
type	The nature or genre of the resource described by this local resources object.

PROPERTY	DESCRIPTION
title	A human readable name assigned to the resource described by this local resources object.
description	A free-text account about the resource described by this local resources object.
keywords	The topic or topics associated with the resource described by this local resources object. Typically represented using free-form keywords, tags, key phrases, or classification codes.
themes	A knowledge organization system used to classify the resource described by this local resources object.
language	The language used for textual values for this local resources object.
languages	This list of other languages in which this local resources object can be retrieved.
resourceLanguages	The list of languages in which the resource described by this local resources object is available.
formats	A list of available distributions of the resource described by this local resources object.
contacts	A list of contacts qualified by their role(s).
license	A legal document under which the resource described by this local resources object is made available.
rights	A statement that concerns all rights not addresses by the license such as a copyright statement.

PERMISSION 21	
IDENTIFIER	`/per/local-resources-catalog/other-catalog-properties`
A	A local resources catalog MAY contain any number of other additional properties not listed in Table 11 and Table 15. The meaning of these other additional properties is not specified in this document.

In practice the responsible standard would define how the two sets of properties are merged into the local resources catalog. In this example it is straight forward since the collections object (path: `/collections`) is sparse and most catalog properties can be added without issue. It should further be noted that for the collections object (path: `/collections`) the name of the array containing the local resources items (i.e., the records) is **"collections"** and so the value of the `recordsArrayName` property would be set to **"collections"**.

8.4.5. Extending the local resources item

As an example of a local resources item, consider the *collection* object defined in defined in OGC API — Features. The following table cross-walks the information content of the collection object with the properties defined for the Record Core common component:

Table 16 — Crosswalk of collection and record properties

COLLECTION PROPERTY	REQUIREMENT	RECORD PROPERTY	REQUIREMENT
id	**required**	id	**required**
—	—	created	optional
—	—	updated	optional
—	—	conformsTo	optional
—	—	type	optional
itemType	optional	—	—
title	optional	title	optional
description	optional	description	optional
extent (space)	optional	geometry	optional
extent (time)	optional	time	optional
crs	optional	—	—
—	—	keywords	optional
—	—	themes	optional
—	—	language	optional
—	—	languages	optional
—	—	resourceLanguages	optional
—	—	externalIds	optional

COLLECTION PROPERTY	REQUIREMENT	RECORD PROPERTY	REQUIREMENT
—	—	formats	optional
—	—	contacts	optional
—	—	license	optional
—	—	rights	optional
links	**required**	links	optional
linkTemplates	optional	linkTemplates	optional

Many of the properties of the collection, a local resources item, are also properties of a record but the searchability of the collection can be improved by enhancing its information content. Thus, the following requirements and recommendations are made:

REQUIREMENT 82	
IDENTIFIER	`/req/local-resources-catalog/mandatory-record-properties`
INCLUDED IN	Requirements class 15: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog`
A	Each record accessible via the local resource catalog SHALL contain all the mandatory properties listed in Table 8 and Table 9.

PERMISSION 22	
IDENTIFIER	`/per/local-resources-catalog/additional-record-properties`
A	Table 15 table contains a list of recommended additional properties that MAY be added to the information model of a local resources i.e., to enhance its searchability.

PERMISSION 23	
IDENTIFIER	`/per/local-resources-catalog/other-record-properties`
A	Each local resources i.e., MAY contain any number of other additional properties not listed in Table 8, Table 9 and Table 15. The meaning of these other additional properties is not specified in this document.

In practice the responsible standard would define how the two sets of properties are merged into the local resources item. In this case it is straight forward since the collection object (path: `/collections/{collectionID}`) and the record share a large number of properties. Where there is a difference, `extent` versus `geometry` and `time`, can be handled by including both sets of properties in the local resources i.e., and ensuring that their values are consistent.

8.4.6. Examples

8.4.6.1. Collections example

The following JSON Schema fragments illustrate how to extend the https://docs.ogc.org/is/17-069r4/17-069r4.html#_collections_ [collections object] (a local resources catalog) and the collection object (a local resources item) to include properties required and recommended by this requirements class.

```
---
allOf:
  - $ref: 'https://schemas.opengis.net/ogcapi/features/part1/1.0/openapi/schemas/collections.yaml'
  - $ref: '../../../openapi/schemas/catalogCommonProperties.yaml'
  - type: object
    properties:
      itemType:
        type: string
        enum:
          - record
      recordsArrayName:
        type: string
        enum:
          - collections
```

Listing 44

```
---
allOf:
  - $ref: 'https://schemas.opengis.net/ogcapi/features/part1/1.0/openapi/schemas/collection.yaml'
  - $ref: '../../../openapi/schemas/recordCommonProperties.yaml'
```

Listing 45

NOTE: Referenced schemas can be found at:

- collections.yaml,
- collection.yaml,
- catalog.yaml,
- recordCommonProperties.yaml,
- linkBase.yaml,
- linkTemplate.yaml.

8.4.6.2. Processes example

The following JSON Schema fragments illustrate how to extend the processes object (a local resources catalog) and the process summary (a local resources item) to include properties required and recommended by this requirements class.

```
---
allOf:
  - $ref: 'https://schemas.opengis.net/ogcapi/processes/part1/1.0/openapi/
schemas/processList.yaml'
  - $ref: '../../../openapi/schemas/catalogCommonProperties.yaml'
  - type: object
    properties:
      itemType:
        type: string
        enum:
          - record
      recordsArrayName:
        type: string
        enum:
          - processes
```

Listing 46

```
---
allOf:
  - $ref: 'https://schemas.opengis.net/ogcapi/processes/part1/1.0/openapi/
schemas/processSummary.yaml'
  - $ref: '../../../openapi/schemas/recordCommonProperties.yaml'
```

Listing 47

NOTE 1: Referenced schemas can be found at:

- processList.yaml,
- processSummary.yaml,
- catalog.yaml,
- recordCommonProperties.yaml,
- linkBase.yaml,
- linkTemplate.yaml.

NOTE 2: T.D.B. Should add an instance example of a collection object.

8.4.7. Discovery of local resources catalog endpoints

Discovery of which local resources catalog endpoints in an OGC API deployment are instrumented for catalog-like searching is accomplished using Autodiscovery.

REQUIREMENT 83	
IDENTIFIER	/req/local-resources-catalog/discovery
INCLUDED IN	Requirements class 15: http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog
A	The links section of a server's landing page SHALL include one link (relation: http://www.opengis.net/def/rel/ogc/1.0/ogc-catalog) for each API endpoint (e.g., /collections) that has been instrumented to be a local resources catalog.

8.4.8. Requirements Class "Local Resources Catalog — Query Parameters" (Deployment)

REQUIREMENTS CLASS 16	
IDENTIFIER	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog/query-parameters
TARGET TYPE	Web API
PREREQUISITES	Local Resources Catalog Record Core Query Parameters
NORMATIVE STATEMENTS	Requirement 84: /req/local-resource-catalog/query-parameters/conformance Requirement 85: /req/local-resources-catalog/query-parameters Requirement 86: /req/local-resources-catalog/response

REQUIREMENT 84	
IDENTIFIER	/req/local-resource-catalog/query-parameters/conformance
INCLUDED IN	Requirements class 16: http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog/query-parameters
A	A server's conformance declaration SHALL be retrieved from the /conformance endpoint as defined in the Declaration of conformance classes clause of the OGC API — Features — Part 1: Core Standard.
B	An implementation of this requirements class SHALL declare the following conformance classes in the conformance declaration: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog-query-parameters

8.4.8.1. Operation

REQUIREMENT 85	
IDENTIFIER	`/req/local-resources-catalog/query-parameters`
INCLUDED IN	Requirements class 16: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog/query-parameters`
A	The server SHALL, at the local resources endpoint, implement one or more of the Record Core Query Parameters as required to satisfy the server's search requirements.
B	The set of Record Core Query Parameters implemented at the local resources endpoint SHALL be defined in the server's API definition document.

8.4.8.2. Response

REQUIREMENT 86	
IDENTIFIER	`/req/local-resources-catalog/response`
INCLUDED IN	Requirements class 16: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog/query-parameters`
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The response SHALL conform to the requirements of the local resources endpoint being extended.
C	The response SHALL only include local resources objects selected by the request (e.g., only collections that satisfy the query predicates are retrieved from the `/collections` endpoint).

8.4.9. Requirements Class "Local Resources Catalog — Filtering" (Deployment)

8.4.9.1. Overview

REQUIREMENTS CLASS 17	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog/filtering`

REQUIREMENTS CLASS 17	
TARGET TYPE	Web API
PREREQUISITES	Local Resources Catalog Filtering
NORMATIVE STATEMENTS	Requirement 87: `/req/local-resource-catalog/filtering-conformance` Requirement 88: `/req/local-resources-catalog/filtering` Requirement 89: `/req/features-filter/filtering/queryables-link`

The Local Resources Catalog — Filtering requirements class specifies requirements and recommendations for supporting advanced filtering at *local resource catalog* endpoints.

8.4.9.2. Conformance

REQUIREMENT 87	
IDENTIFIER	`/req/local-resource-catalog/filtering-conformance`
INCLUDED IN	Requirements class 17: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog/filtering`
A	A server's conformance declaration SHALL be retrieved from the `/conformance` endpoint as defined in the Declaration of conformance classes clause of the OGC API — Features — Part 1: Core Standard.
B	An implementation of this requirements class SHALL declare the following conformance classes in the conformance declaration: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog-filtering

8.4.9.3. Filtering

REQUIREMENT 88	
IDENTIFIER	`/req/local-resources-catalog/filtering`
INCLUDED IN	Requirements class 17: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog/filtering`
STATEMENT	Implementations that offer enhanced filtering SHALL, at the local resources catalog endpoint (e.g., `/collections`), implement the following requirements class.
A	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/filtering

REQUIREMENT 89	
IDENTIFIER	/req/features-filter/filtering/queryables-link
INCLUDED IN	Requirements class 17: http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog/filtering
A	Implementations of this requirements class SHALL, in the links section of the local resources endpoint, include a link (relation: http://www.opengis.net/def/rel/ogc/1.0/queryables).
B	This link SHALL point to a resource for discovering the list of record properties (and their types) that may be used to construct a filter expressions.

8.4.10. Requirements Class "Local Resources Catalog — Sorting" (Deployment)

REQUIREMENTS CLASS 18	
IDENTIFIER	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog/sorting
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.16: http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog/sorting
PREREQUISITES	Local Resources Catalog Sorting
NORMATIVE STATEMENTS	Requirement 90: /req/local-resources-catalog/sorting-conformance Requirement 91: /req/local-resources-catalog/sorting

The Local Resources Catalog — Sorting requirements class specifies requirements and recommendations for supporting sorting of responses at *local resource catalog* endpoints.

8.4.10.1. Conformance

REQUIREMENT 90	
IDENTIFIER	/req/local-resources-catalog/sorting-conformance
INCLUDED IN	Requirements class 18: http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog/sorting

REQUIREMENT 90	
A	A server's conformance declaration SHALL be retrieved from the `/conformance` endpoint as defined in the Declaration of conformance classes clause of the OGC API — Features — Part 1: Core Standard.
B	An implementation of this requirements class SHALL declare the following conformance classes in the conformance declaration: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog-sorting

8.4.10.2. Sorting

REQUIREMENT 91	
IDENTIFIER	`/req/local-resources-catalog/sorting`
INCLUDED IN	Requirements class 18: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog/sorting`
STATEMENT	Implementations of sorting of record responses SHALL, at the local resources endpoint (e.g., `/collections`), implement the following requirements class.
A	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/sorting

See also: /req/sorting/get-sortables-op, /req/sorting/get-sortables-success.

8.4.10.3. Examples

The following example illustrates a catalog filter expression being applied to the local resources endpoint /collections. In this case, the client is seeking all collection objects whose extent intersects a specified bounding box and that include the keyword “windmills”.

```
        "keywords": [
          "Netherlands",
          "INSPIRE",
          "Windmills",
          "Heritage",
          "Holland",
          "RD"
        ],
        "links": [
          {
            "type": "text/html",
            "rel": "canonical",
            "title": "information",
            "href": "https://www.nationaalgeoregister.nl/geone
                    twork/srv/dut/catalog.search#/metadata/29
                    1afe4b-4f4b-497c-8026-fb437c4e9c7e",
            "hreflang": "nl-NL"
          },
          {
            "type": "application/json",
            "rel": "root",
            "title": "Landing page of this server as JSON",
            "href": "https://demo.pygeoapi.io/master?f=json"
          },
          {
            "type": "text/html",
            "rel": "root",
            "title": "Landing page of this server as HTML",
            "href": "https://demo.pygeoapi.io/master?f=html"
          },
          {
            "type": "application/json",
            "rel": "self",
            "title": "This document as JSON",
            "href": "https://demo.pygeoapi.io/master/collectio
                    ns/dutch_windmills?f=json"
          },
          {
            "type": "application/ld+json",
            "rel": "alternate",
            "title": "This document as RDF (JSON-LD)",
            "href": "https://demo.pygeoapi.io/master/collectio
                    ns/dutch_windmills?f=jsonld"
          },
          {
            "type": "text/html",
            "rel": "alternate",
            "title": "This document as HTML",
            "href": "https://demo.pygeoapi.io/master/collectio
                    ns/dutch_windmills?f=html"
          },
          {
            "type": "application/json",
            "rel": "http://www.opengis.net/def/rel/ogc/1.0/sch
                   ema",
            "title": "Schema of collection in JSON",
            "href": "https://demo.pygeoapi.io/master/collectio
                    ns/dutch_windmills/schema?f=json"
          },
          {
            "type": "text/html",
            "rel": "http://www.opengis.net/def/rel/ogc/1.0/sch
                   ema",
```

```
      "title": "Schema of collection in HTML",
      "href": "https://demo.pygeoapi.io/master/collectio
               ns/dutch_windmills/schema?f=html"
    },
    {
      "type": "application/schema+json",
      "rel": "http://www.opengis.net/def/rel/ogc/1.0/que
             ryables",
      "title": "Queryables for this collection as JSON",
      "href": "https://demo.pygeoapi.io/master/collectio
               ns/dutch_windmills/queryables?f=json"
    },
    {
      "type": "text/html",
      "rel": "http://www.opengis.net/def/rel/ogc/1.0/que
             ryables",
      "title": "Queryables for this collection as HTML",
      "href": "https://demo.pygeoapi.io/master/collectio
               ns/dutch_windmills/queryables?f=html"
    },
    {
      "type": "application/geo+json",
      "rel": "items",
      "title": "Items as GeoJSON",
      "href": "https://demo.pygeoapi.io/master/collectio
               ns/dutch_windmills/items?f=json"
    },
    {
      "type": "application/ld+json",
      "rel": "items",
      "title": "Items as RDF (GeoJSON-LD)",
      "href": "https://demo.pygeoapi.io/master/collectio
               ns/dutch_windmills/items?f=jsonld"
    },
    {
      "type": "text/html",
      "rel": "items",
      "title": "Items as HTML",
      "href": "https://demo.pygeoapi.io/master/collectio
               ns/dutch_windmills/items?f=html"
    }
  ],
  "extent": {
    "spatial": {
      "bbox": [
        [
          3.37,
          50.75,
          7.21,
          53.47
        ]
      ],
      "crs": "http://www.opengis.net/def/crs/OGC/1.3/CRS
              84"
    },
    "temporal": {
      "interval": [
        [
          null,
          null
        ]
      ]
    }
```

```
        },
        "itemType": "feature",
        "crs": [
          "http://www.opengis.net/def/crs/OGC/1.3/CRS84",
          "http://www.opengis.net/def/crs/EPSG/0/4326",
          "http://www.opengis.net/def/crs/EPSG/0/3857",
          "http://www.opengis.net/def/crs/EPSG/0/4258",
          "http://www.opengis.net/def/crs/EPSG/0/28992"
        ],
        "storageCRS": "http://www.opengis.net/def/crs/EPSG/0/2
                       8992"
      }
    ],
    "links": [
      {
        "type": "application/json",
        "rel": "self",
        "title": "This document as JSON",
        "href": "https://demo.pygeoapi.io/master/collections?f
                 =json"
      },
      {
        "type": "application/ld+json",
        "rel": "alternate",
        "title": "This document as RDF (JSON-LD)",
        "href": "https://demo.pygeoapi.io/master/collections?f
                 =jsonld"
      },
      {
        "type": "text/html",
        "rel": "alternate",
        "title": "This document as HTML",
        "href": "https://demo.pygeoapi.io/master/collections?f
                 =html"
      }
    ]
  }
<----------------------------------------------------------------
```


Listing 48

9

MEDIA TYPES

9 MEDIA TYPES

9.1. Overview

The OGC API — Records Standard does not mandate any particular encoding for a record or a catalog. However, it does provide requirements classes for encodings that are commonly used in implementations of OGC API Standards. These encodings include JSON and HTML. This clause indicates the media that to be used for these encodings of the resources defined in this Standard.

9.2. Media types

A description of the MIME types is mandatory for any OGC API Standard that specifies requirements for data encoding(s). Table 17 provides a list of suitable MIME types for records and catalogs.

Table 17 — API — Records MIME Types

ENCODING		MEDIA TYPE	PROFILE IDENTIFIER
Record	HTML	`text/html`	n/a
	JSON	`application/json`	`ogc-record`
	GeoJSON	`application/geo+json`	`ogc-record`
Catalog	HTML	`text/html`	n/a
	JSON	`application/json`	`ogc-catalog`
		`application/ogc-catalog+json`	n/a

A JSON-encoded record can be identified using the media type `application/json` or `application/geo+json`. Neither media type, however, specifically identifies the content as a record as defined in this Standard. For this reason, this Standard defines the profile identifier `http://www.opengis.net/def/profile/OGC/0/ogc-record` to more specifically identify JSON-

encoded record content using the profiling mechanism defined in RFC 6909, The 'profile' Link Relation Type.

A JSON-encoded catalog resource can be specifically identified using the media type `application/ogc-catalog+json` but there might be situations where use of the generic media type, `application/json`, is more appropriate. For these siturations this standard defines the profile identifier `http://www.opengis.net/def/profile/OGC/0/ogc-catalog` to more specifically identify JSON-encoded Catalog content using the profiling mechanism defined in RFC 6909, The 'profile' Link Relation Type.

NOTE 1: The tokens `ogc-record` and `ogc-catalog` in Table 17 are the profile identifiers extracted from the complete profile URIs of:

- `http://www.opengis.net/def/profile/OGC/0/ogc-record`
- `http://www.opengis.net/def/profile/OGC/0/ogc-catalog`.

NOTE 2: It should be noted that the use of a generic media type without profile links may require that a resource be completely retrieved and inspected to determine conclusively that it is a record or catalog resource. On the other hand, using resource-specific media types or generic media types with profile links indicates the specific type of resource being dealt with without the need to first retrieve and inspect the complete resource in order to make the type determination.

9.3. Examples

9.3.1. Catalog example

The following example illustrates two ways that a server could respond to a request for a JSON-encoded catalog.

This first response example uses the specific media type, `application/ogc-catalog+json`, to indicate that the response content is a JSON-encoded catalog.

```
      |   Link: <http://www.opengis.net/def/profile/OGC/0/ogc-catalog>; rel=
"profile"|
      |
    |
      |   {
    |
      |     "id":"dutch_windmills",
    |
      |     "type": "Collection",
    |
      |     "title":"Windmills within The Netherlands",
    |
      |     "description":"Locations of windmills within the Netherlands",
    |
      |     "itemType":"record",
    |
      |     "keywords":[
    |
      |       "Netherlands",
    |
      |       "INSPIRE",
    |
      |       "Windmills",
    |
      |       "Heritage",
    |
      |       "Holland",
    |
      |       "RD"
    |
      |     ],
    |
      |     "links":[
    |
      |       .
    |
      |       .
    |
      |       .
    |
      |       {
    |
      |         "type":"application/geo+json",
    |
      |         "rel":"items",
    |
      |         "title":"Items as GeoJSON",
    |
      |         "href":"https://demo.pygeoapi.io/master/collections/dutch_
windmills/items?f=json"
      |       },
    |
      |       {
    |
      |         "type":"application/ld+json",
    |
      |         "rel":"items",
    |
      |         "title":"Items as RDF (GeoJSON-LD)",
    |
      |         "href":"https://demo.pygeoapi.io/master/collections/dutch_
windmills/items?f=jsonld"
```

```
    },
    {
       "href": "http://www.opengis.net/def/profile/OGC/0/ogc-catalog",
       "rel": "profile",
       "title": "This is a Catalog"
    },
    .
    .
    .
  ],
  "extent":{
    "spatial":{
      "bbox":[
        [
          3.37,
          50.75,
          7.21,
          53.47
        ]
      ],
      "crs":"http://www.opengis.net/def/crs/OGC/1.3/CRS84"
    },
    "temporal":{
      "interval":[
        [
          null,
          null
        ]
      ]
    }
  },
```

```
     |     "crs":[
   |
     |       "http://www.opengis.net/def/crs/OGC/1.3/CRS84",
   |
     |       "http://www.opengis.net/def/crs/EPSG/0/4326",
   |
     |       "http://www.opengis.net/def/crs/EPSG/0/3857",
   |
     |       "http://www.opengis.net/def/crs/EPSG/0/4258",
   |
     |       "http://www.opengis.net/def/crs/EPSG/0/28992"
   |
     |     ],
   |
     |     "storageCRS":"http://www.opengis.net/def/crs/EPSG/0/28992"
   |
     | }
   |
     |<-------------------------------------------------------------------------
---|
```

Listing 49

This second response example uses a generic media type, `application/json`, with a profile link (identifier: `ogc-catalog`) to indicate that the response content is a JSON-encoded catalog. As per RFC 6909 and recommendation /rec/query-param-profile/link-header the response includes a profile link in the response HTTP headers.

```
   Client
 Server
     |
   |
     |   GET /collections/dutch_windmills?profile=ogc-catalog   HTTP/1.1
   |
     |   Accept: application/json
   |
     |-------------------------------------------------------------------------
-->|
     |
   |
     |   HTTP/1.1 200 OK
   |
     |   Content-Type: application/json
   |
     |   Link: <http://www.opengis.net/def/profile/OGC/0/ogc-catalog>; rel=
"profile"|
     |
   |
     |   {
   |
     |     "id":"dutch_windmills",
   |
     |     "type": "Collection",
   |
     |     "title":"Windmills within The Netherlands",
   |
     |     "description":"Locations of windmills within the Netherlands",
   |
     |     "itemType":"record",
   |
     |     "keywords":[
   |
```

```
      |       "Netherlands",
      |       "INSPIRE",
      |       "Windmills",
      |       "Heritage",
      |       "Holland",
      |       "RD"
      |     ],
      |     "links":[
      |       .
      |       .
      |       .
      |       {
      |         "type":"application/geo+json",
      |         "rel":"items",
      |         "title":"Items as GeoJSON",
      |         "href":"https://demo.pygeoapi.io/master/collections/dutch_
windmills/items?f=json"
      |       },
      |       {
      |         "type":"application/ld+json",
      |         "rel":"items",
      |         "title":"Items as RDF (GeoJSON-LD)",
      |         "href":"https://demo.pygeoapi.io/master/collections/dutch_
windmills/items?f=jsonld"
      |       },
      |       {
      |          "href": "http://www.opengis.net/def/profile/OGC/0/ogc-catalog",
      |          "rel": "profile",
      |          "title": "This is a Catalog"
      |       },
      |       .
      |       .
      |       .
      |
```

```
    ],
    "extent":{
      "spatial":{
        "bbox":[
          [
            3.37,
            50.75,
            7.21,
            53.47
          ]
        ],
        "crs":"http://www.opengis.net/def/crs/OGC/1.3/CRS84"
      },
      "temporal":{
        "interval":[
          [
            null,
            null
          ]
        ]
      }
    },
    "crs":[
      "http://www.opengis.net/def/crs/OGC/1.3/CRS84",
      "http://www.opengis.net/def/crs/EPSG/0/4326",
      "http://www.opengis.net/def/crs/EPSG/0/3857",
      "http://www.opengis.net/def/crs/EPSG/0/4258",
      "http://www.opengis.net/def/crs/EPSG/0/28992"
    ],
    "storageCRS":"http://www.opengis.net/def/crs/EPSG/0/28992"
}
```

```
     |<-------------------------------------------------------------------------
---|
```

Listing 50

In both cases the server follows good practice and includes the profile link in the `links` section of the catalog for those situations where the response document is dissociated from the server that generated it.

9.3.2. Record example

The following example illustrates how a server might respond to a request for a GeoJSON-encoded record.

As per RFC 6906, recommendation /rec/query-param-profile/link-header and recommendation /rec/json/header-profile-link the response includes a profile link in the response HTTP headers. The same link is also included in the `links` section of the record for those situations where the response document is dissociated from the server that generated it.

```
   Client
 Server
     |
   |
     |   GET /collections/my_catalog/items/r01?profile=ogc-record   HTTP/1.1
   |
     |   Accept: application/geo+json
   |
     |-------------------------------------------------------------------------
-->|
     |
   |
     |   HTTP/1.1 200 OK
   |
     |   Content-Type: application/geo+json
   |
     |   Link: <http://www.opengis.net/def/profile/OGC/0/ogc-record>; rel=
"profile" |
     |
   |
     |   {
   |
     |
   |
     |     ... JSON-encoded records content ...
   |
     |
   |
     |     "links": [
   |
     |       .
   |
     |       .
   |
     |       .
   |
     |       {
   |
     |          "href": "http://www.opengis.net/def/profile/OGC/0/ogc-record",
   |
```

```
    |          "rel": "profile",
    |
    |          "title": "This is a OARec Record"
    |
    |       },
    |
    |       .
    |
    |       .
    |
    |       .
    |
    |     ]
    |
    |   }
    |
    |<-------------------------------------------------------------------------
---|
```

Listing 51

9.4. Default Encodings

The canonical method of selecting a desired response encoding is to use HTTP content negotiation. However, content negotiation is not required by the HTTP standard. Therefore, default encodings must be established.

REQUIREMENT 92	
IDENTIFIER	`/req/record-core/default-mediatype`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core`
A	If the JSON conformance class is advertised, then the default media type for record content SHALL be GeoJSON (i.e., `application/geo+json; application=ogc-record`).
B	If the JSON conformance class is not advertised, then the default media type for record content SHALL be HTML (i.e `text/html`).

REQUIREMENT 93	
IDENTIFIER	`/req/record-collection/default-mediatype`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
A	If the JSON conformance class is advertised, then the default media type for catalog content SHALL be JSON (i.e., `application/ogc-catalog+json`).

REQUIREMENT 93	
B	If the JSON conformance class is not advertised, then the default media type for catalog content SHALL be HTML (i.e., `text.html`).

10

REQUIREMENTS CLASS "PROFILE QUERY PARAMETER"

10 REQUIREMENTS CLASS "PROFILE QUERY PARAMETER"

The Requirements Class "Profile query parameter" specifies additional provisions for properties that reference another resource.

REQUIREMENTS CLASS 19	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/req/query-param-profile`
TARGET TYPE	Web API
CONFORMANCE CLASS	Conformance class A.5: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/query-param-profile`
NORMATIVE STATEMENTS	Requirement 94: `/req/query-param-profile/definition` Requirement 95: `/req/query-param-profile/response`

Some properties may be represented in multiple representations in the same format, depending on the intended use of the data. One example are references to another web resource (see the [rc_profile-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 94	
IDENTIFIER	`/req/query-param-profile/definition`
INCLUDED IN	Requirements class 19: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/query-param-profile`
A	A Records API implementation SHALL support the Profile (`profile`) parameter for the GET operation.

B	The following OpenAPI 3.0 schema fragment SHALL define the characteristics of the `profile` parameter:
	`name: profile`
	`in: query`

REQUIREMENT 94

```
required: false
schema:
  type: array
  items:
    type: string
explode: false
style: form
```

C

Each i.e., SHALL be one of the following:

- A HTTP(S) URI of a profile, e.g., in the OGC Profile Register (http://www.opengis.net/def/profile);
- A Safe CURIE of a profile in the OGC Profile Register (e.g., `[ogc-profile:my-profile]`);
- A profile identifier (the `profileId` value in the URI template `http://www.opengis.net/def/profile/OGC/0/{profileId}`)

PERMISSION 24

IDENTIFIER	`/per/query-param-profile/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 25

IDENTIFIER	`/per/query-param-profile/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 36

IDENTIFIER	`/rec/query-param-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 95

IDENTIFIER	`/req/query-param-profile/response`
INCLUDED IN	Requirements class 19: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/query-param-profile`
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 37

IDENTIFIER	`/rec/query-param-profile/link-header`
A	The response SHOULD include links to the selected profile(s) in the HTTP response headers.

A

ANNEX A (NORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE (NORMATIVE)

ANNEX A (NORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE (NORMATIVE)

A.1. Common components

CONFORMANCE CLASS A.1	
IDENTIFIER	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core
REQUIREMENTS CLASS	Requirements class 1: http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.1: /conf/record-core/mandatory-properties-record Abstract test A.2: /conf/record-core/links Abstract test A.3: /conf/record-core/templated-link Abstract test A.4: /conf/record-core/templated-link-header Abstract test A.5: /conf/record-core/license Abstract test A.6: /conf/record-core/contact Abstract test A.7: /conf/record-core/time-instant-interval Abstract test A.8: /conf/record-core/time-instant Abstract test A.9: /conf/record-core/time-interval Abstract test A.10: /conf/record-core/time-zone Abstract test A.11: /conf/record-core/default-mediatype

A.1.1. Mandatory properties

ABSTRACT TEST A.1	
IDENTIFIER	/conf/record-core/mandatory-properties-record

ABSTRACT TEST A.1	
REQUIREMENT	Requirement 1: `/req/record-core/mandatory-properties-record`
TEST PURPOSE	Validate that a catalog record includes all mandatory properties.
TEST METHOD	1. Construct a path for a given catalog record. 2. Issue a HTTP GET request on that path. 3. Validate that the retrieved content includes all the mandatory properties listed in Table 8. 4. Validate that the `id` property is not NULL or an empty string.

A.1.2. Links

ABSTRACT TEST A.2	
IDENTIFIER	`/conf/record-core/links`
REQUIREMENT	Requirement 8: `/req/record-core/links`
TEST PURPOSE	Validate the behavior of link relations for link objects.
TEST METHOD	1. Construct a path for a given record. 2. Issue a HTTP GET request on that path. 3. Parse all link objects. 4. Check if the record includes links with a `rel` property value of `collection`. 5. If it does, ensure that only one such link exists in the record. 6. Issue a HTTP GET request on that path. 7. Validate that the retrieved content is a valid catalog object.

A.1.3. Link templates

ABSTRACT TEST A.3	
IDENTIFIER	`/conf/record-core/templated-link`
REQUIREMENT	Requirement 9: `/req/record-core/templated-link`

ABSTRACT TEST A.3	
TEST PURPOSE	Validate the behavior of templated links.
TEST METHOD	1. Construct a path for a given record. 2. Issue a HTTP GET request on that path. 3. Parse all link template objects. 4. Verify that each link template includes a `uriTemplate` property. 5. The link template may optionally include a `varBase` property. 6. The link template may optionally include a `variables` property.

ABSTRACT TEST A.4	
IDENTIFIER	`/conf/record-core/templated-link-header`
REQUIREMENT	Requirement 10: `/req/record-core/templated-link-header`
TEST PURPOSE	Validate the behavior of templated link header.
TEST METHOD	1. Construct a path for a given record with a templated link. 2. Check that a templated link appears in a HTTP header and is encoded according to https://ietf-wg-httpapi.github.io/link-template/draft-ietf-httpapi-link-template.html.

A.1.4. License

ABSTRACT TEST A.5	
IDENTIFIER	`/conf/record-core/license`
REQUIREMENT	Requirement 7: `/req/record-core/license`
TEST PURPOSE	Validate the behavior of license object.
TEST METHOD	1. Construct a path for a given record. 2. Issue a HTTP GET request on that path. 3. Parse all link objects. 4. Parse the license object if it exists. 5. Check if the license includes a SPDX license identifier or the value `other`. 6. If the value `other` is used, check that one link has a `rel` property with value of `license`.

A.1.5. Contacts

ABSTRACT TEST A.6	
IDENTIFIER	`/conf/record-core/contact`
REQUIREMENT	Requirement 6: `/req/record-core/contact`
TEST PURPOSE	Validate the behavior of contacts objects.
TEST METHOD	1. Construct a path for a given record. 2. Issue a HTTP GET request on that path. 3. Parse the `contacts` member if it exists. 4. If there is a contacts `logo` member, check that the link has a `rel` property with value of `icon`. Check that the media type is an `image` media type. 5. If there is a contacts `links` member, check that the link has a `rel` property with value of `about`. Check that the media type indicates the content type of the link (e.g., `text/html` for a company's web page versus `text/vcard` for a virtual contact file).

A.1.6. Time

ABSTRACT TEST A.7	
IDENTIFIER	`/conf/record-core/time-instant-interval`
REQUIREMENT	Requirement 4: `/req/record-core/time-instant-interval`
TEST PURPOSE	Validate the behavior of time instance and intervals in time object.
TEST METHOD	1. Construct a path for a given record that includes the `time` member. 2. Issue a HTTP GET request on that path. 3. Parse time object. 4. If the time object includes both a `date` and a `timestamp` member, check that the `full-date` parts are identical. 5. If the time object includes both a `timestamp` and an `interval` member with start/end dates, check that the interval contains the date of the timestamp. 6. If the time object includes both a `timestamp` and an `interval` member with start/end timestamps check that the interval contains the timestamp.

ABSTRACT TEST A.7

7. If the time object includes both a `date` and an `interval` member with start/end dates, check that the interval contains the date.
8. If the time object includes both a `date` and an `interval` member with start/end timestamps, check that the interval includes timestamps on the date.

ABSTRACT TEST A.8

IDENTIFIER	`/conf/record-core/time-instant`
REQUIREMENT	Requirement 2: `/req/record-core/time-instant`
TEST PURPOSE	Validate the behavior of time instances for time objects.
TEST METHOD	1. Construct a path for a given record that includes the `time` member. 2. Issue a HTTP GET request on that path. 3. Parse time object. 4. If the time object includes a `date` member, check that the value conforms to RFC 3339 (Date and Time on the Internet: Timestamps) and matches the production rule `full-date`. 5. If the time object includes a `timestamp` member, check that the value conforms to RFC 3339 (Date and Time on the Internet: Timestamps) and matches the production rule `date-time`.

ABSTRACT TEST A.9

IDENTIFIER	`/conf/record-core/time-interval`
REQUIREMENT	Requirement 3: `/req/record-core/time-interval`
TEST PURPOSE	Validate the behavior of time interval for time objects.
TEST METHOD	1. Construct a path for a given record that includes the `time` member. 2. Issue a HTTP GET request on that path. 3. Parse time object. 4. If the time object includes an `interval` member, check that each array i.e., is a string that is a double-dot (`..`) or conforms to RFC 3339 (Date and Time on the Internet: Timestamps) and matches one of the following production rules: `full-date` (a date) or `date-time` (a timestamp) 5. If the start is a date, check that the end is a date, or `..` 6. If the start is a timestamp, check that the end is a timestamp or `..`

ABSTRACT TEST A.10	
IDENTIFIER	`/conf/record-core/time-zone`
REQUIREMENT	Requirement 5: `/req/record-core/time-zone`
TEST PURPOSE	Validate the behavior of time zone for time member.
TEST METHOD	1. Construct a path for a given record that includes the `time` member. 2. Issue a HTTP GET request on that path. 3. Parse time member. 4. Check that timestamps in the time member use UTC “Z” as the time zone.

A.1.7. Default media type

ABSTRACT TEST A.11	
IDENTIFIER	`/conf/record-core/default-mediatype`
REQUIREMENT	Requirement 92: `/req/record-core/default-mediatype`
TEST PURPOSE	Check the default encoding for a catalog record.
TEST METHOD	1. Construct a path for a given catalog record. 2. Issue a HTTP GET request on that path. 3. Check if the JSON conformance class is advertised for the catalog. 4. If yes, then check that the media type for content is `application/geo+json; application=ogc-record`. 5. If no, then check that the media type for content is HTML.

CONFORMANCE CLASS A.2	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-collection`
REQUIREMENTS CLASS	Requirements class 2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.12: `/conf/record-collection/mandatory-properties-collection` Abstract test A.13: `/conf/record-collection/itemType`

CONFORMANCE CLASS A.2
Abstract test A.14: `/conf/record-collection/contact` Abstract test A.15: `/conf/record-collection/license` Abstract test A.16: `/conf/record-collection/links` Abstract test A.17: `/conf/record-collection/links-catalog-hierarchy` Abstract test A.18: `/conf/record-collection/links-catalog-related` Abstract test A.19: `/conf/record-collection/links-catalog-series` Abstract test A.20: `/conf/record-collection/links-record` Abstract test A.21: `/conf/record-collection/links-records` Abstract test A.22: `/conf/record-collection/records-array-name` Abstract test A.23: `/conf/record-collection/records` Abstract test A.24: `/conf/record-collection/default-mediatype`

A.1.8. Catalog properties

ABSTRACT TEST A.12	
IDENTIFIER	`/conf/record-collection/mandatory-properties-collection`
REQUIREMENT	Requirement 11: `/req/record-collection/mandatory-properties-collection`
TEST PURPOSE	Validate that a catalog object includes all mandatory properties.
TEST METHOD	1. Construct a path for a catalog object. 2. Issue a HTTP GET request on that path. 3. Validate that the retrieved content includes all the mandatory properties listed in Table 11.

ABSTRACT TEST A.13	
IDENTIFIER	`/conf/record-collection/itemType`
REQUIREMENT	Requirement 12: `/req/record-collection/itemType`
TEST PURPOSE	Validate the behavior of `itemType` member.
TEST METHOD	1. Construct a path for a catalog object. 2. Issue a HTTP GET request on that path. 3. Parse a `itemType` member if it exists. 4. If the value of the `itemType` member is `record` verify that the catalog contains or references only record objects.

ABSTRACT TEST A.13	
	5. If the value of the `itemType` member is `catalog` verify that the catalog only references other catalog objects. 6. If the value of the `itemType` member is an array containing the values `record` and `catalog` verify that the catalog only references other record and catalog objects.

ABSTRACT TEST A.14	
IDENTIFIER	`/conf/record-collection/contact`
REQUIREMENT	Requirement 13: `/req/record-collection/contact`
TEST PURPOSE	Validate the behavior of `contacts` objects.
TEST METHOD	1. Construct a path for a catalog object. 2. Issue a HTTP GET request on that path. 3. Parse a `contacts` member if it exists. 4. If there is a contacts `logo` member, check that the link has a `rel` property with value of `icon`. Check that the media type is an `image` media type. 5. If there is a contacts `links` member, check that the link has a `rel` property with value of `about`. Check that the media type indicates the content type of the link (e.g., `text/html` for a company's web page versus `text/vcard` for a virtual contact file).

ABSTRACT TEST A.15	
IDENTIFIER	`/conf/record-collection/license`
REQUIREMENT	Requirement 14: `/req/record-collection/license`
TEST PURPOSE	Validate the behavior of the `license` member.
TEST METHOD	1. Construct a path for a catalog object. 2. Issue a HTTP GET request on that path 3. Parse the `license` member if it exists. 4. Check if the license includes a SPDX license identifier or the value `other`. 5. If the value `other` is used, check that one link has a `rel` property with value of `license`.

A.1.9. Catalog links

ABSTRACT TEST A.16

IDENTIFIER	`/conf/record-collection/links`
REQUIREMENT	Requirement 22: `/req/record-collection/links`
TEST PURPOSE	Validate the behavior of “self” and “alternate” links.
TEST METHOD	1. Construct a path for catalog object. 2. Issue a HTTP GET request on that path. 3. Parse all link objects. 4. Check that one link has a `rel` property with value of `self`. 5. Issue a HTTP GET request on the resulting path. 6. Validate that the content between both requests is equal. 7. Check if the catalog has alternative representations and check that it has links with `rel` property with value of `alternate`. 8. Issue a HTTP GET request on each resulting path. 9. Validate that the retrieved content satisfies the asserted media type of the alternate representation.

ABSTRACT TEST A.17

IDENTIFIER	`/conf/record-collection/links-catalog-hierarchy`
REQUIREMENT	Requirement 21: `/req/record-collection/links-catalog-hierarchy`
TEST PURPOSE	Validate the behavior of link relations to sub-ordinate catalogs.
TEST METHOD	1. Construct a path for catalog object. 2. Issue a HTTP GET request on that path. 3. Parse all link objects. 4. Check if the catalog has links with `rel` property with value of `child`. 5. Issue a HTTP GET request on each resulting path. 6. Validate that the retrieved content is a catalog object.

ABSTRACT TEST A.18

IDENTIFIER	`/conf/record-collection/links-catalog-related`

ABSTRACT TEST A.18

REQUIREMENT	Requirement 19: `/req/record-collection/links-catalog-related`
TEST PURPOSE	Validate the behavior of link to related catalogs.
TEST METHOD	1. Construct a path for catalog object. 2. Issue a HTTP GET request on that path. 3. Parse all link objects. 4. Check if the catalog has links with `rel` property with value of `related`. 5. Issue a HTTP GET request on each resulting path. 6. If the media type of the retrived content indicates a catalog, validate that the retrieved content is a valid catalog object.

ABSTRACT TEST A.19

IDENTIFIER	`/conf/record-collection/links-catalog-series`
REQUIREMENT	Requirement 20: `/req/record-collection/links-catalog-series`
TEST PURPOSE	Validate the behavior of a series of catalogs.
TEST METHOD	1. Construct a path for catalog object. 2. Issue a HTTP GET request on that path. 3. Parse all link objects. 4. Check if the catalog has links with `rel` property with value of `next`. 5. Issue a HTTP GET request on each resulting path. 6. If the media type of the retrived content indicates a catalog, validate that the retrieved content is a valid catalog object.

ABSTRACT TEST A.20

IDENTIFIER	`/conf/record-collection/links-record`
REQUIREMENT	Requirement 15: `/req/record-collection/links-record`
TEST PURPOSE	Validate the behavior of explicit links to records.
TEST METHOD	1. Construct a path for catalog object. 2. Issue a HTTP GET request on that path.

ABSTRACT TEST A.20	
	3. Parse all link objects. 4. Check if the catalog has links with `rel` property with value of `item`. 5. Issue a HTTP GET request on each resulting path. 6. If the media type of the retrived content indicates a record, validate that the retrieved content is a valid record object.

ABSTRACT TEST A.21	
IDENTIFIER	`/conf/record-collection/links-records`
REQUIREMENT	Requirement 16: `/req/record-collection/links-records`
TEST PURPOSE	Validate the behavior of a link to a record retrieval endpoint.
TEST METHOD	1. Construct a path for catalog object. 2. Issue a HTTP GET request on that path. 3. Parse all link objects. 4. Check if the catalog has links with `rel` property with value of `items`. 5. Issue a HTTP GET request on each path. 6. Verify that the retrived content is a valid record collection.

A.1.10. Catalog records

ABSTRACT TEST A.22	
IDENTIFIER	`/conf/record-collection/records-array-name`
REQUIREMENT	Requirement 18: `/req/record-collection/records-array-name`
TEST PURPOSE	Validate that the named of the inline records array is correctly advertised.
TEST METHOD	1. Construct a path for a given catalog. 2. Issue a HTTP GET request on that path. 3. If the retrieved content contains a `recordsArrayName` then locate a member whose name is the value of the `recordsArrayName` member. 4. Verify that this member exists in the catalog object and that it is an array.

ABSTRACT TEST A.23	
IDENTIFIER	`/conf/record-collection/records`
REQUIREMENT	Requirement 17: `/req/record-collection/records`
TEST PURPOSE	Validate that inline records are correctly represented.
TEST METHOD	1. Construct a path for a given catalog. 2. Issue a HTTP GET request on that path. 3. If the retrieved content contains a `recordsArrayName` then locate a member whose name is the value of the `recordsArrayName` member. Otherwise locate a member named `records`. 4. If this member exists, verify that it is an array. 5. If this member exists, verify that each member of this array is a valid record.

A.1.11. Media type

ABSTRACT TEST A.24	
IDENTIFIER	`/conf/record-collection/default-mediatype`
REQUIREMENT	Requirement 93: `/req/record-collection/default-mediatype`
TEST PURPOSE	Check the default encoding for a catalog.
TEST METHOD	1. Construct a path for a given catalog. 2. Issue a HTTP GET request on that path. 3. Check if the JSON conformance class is advertised for the catalog. 4. If yes, then check that the media type for content is `application/ogc-catalog+json`. 5. If no, then check that the media type for content is HTML.

CONFORMANCE CLASS A.3	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/records-api`
REQUIREMENTS CLASS	Requirements class 4: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/records-api`
TARGET TYPE	Web API

CONFORMANCE CLASS A.3	
CONFORMANCE TESTS	Abstract test A.25: `/conf/records-api/features-api` Abstract test A.26: `/conf/records-api/catalog-response` Abstract test A.27: `/conf/records-api/catalogs-response` Abstract test A.28: `/conf/records-api/record-op` Abstract test A.29: `/conf/records-api/record-response` Abstract test A.30: `/conf/records-api/records-op` Abstract test A.31: `/conf/records-api/resource-name-mapping` Abstract test A.32: `/conf/records-api/mandatory-params`

A.1.12. Records API

ABSTRACT TEST A.25	
IDENTIFIER	`/conf/records-api/features-api`
REQUIREMENT	Requirement 34: `/req/records-api/features-api`
TEST PURPOSE	Validate compliance to Features Core.
TEST METHOD	1. Demonstrate compliance with Conformance Class Core of OGC API — Features — Part 1: Core.

ABSTRACT TEST A.26	
IDENTIFIER	`/conf/records-api/catalog-response`
REQUIREMENT	Requirement 36: `/req/records-api/catalog-response`
TEST PURPOSE	Validate collection API endpoint.
TEST METHOD	1. Construct a path to retrieve a catalog description (path: `/collections/{collectionId}`). 2. Issue a HTTP GET request on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Verify that the collection validates against the schema as defined in collection.yaml. 5. Verify that the `itemType` property is a string with the value `record`.

ABSTRACT TEST A.27

IDENTIFIER	`/conf/records-api/catalogs-response`
REQUIREMENT	Requirement 37: `/req/records-api/catalogs-response`
TEST PURPOSE	Validate collections API endpoint.
TEST METHOD	1. Construct a path to retrieve a list of collections (path: `/collections`). 2. Issue a HTTP GET request on that path. 3. Check that the value of the returned HTTP status header is 200. 4. For each collection where the `itemType` property is a string with the value `record` use the test /conf/records-api/catalog-response to validate the schema of each catalog description.

ABSTRACT TEST A.28

IDENTIFIER	`/conf/records-api/record-op`
REQUIREMENT	Requirement 41: `/req/records-api/record-op`
TEST PURPOSE	Validate that a record can be retrieved from the expected location.
TEST METHOD	1. For a sufficiently large subset of all records in a catalog (path: `/collections/{collectionId}`), issue an HTTP GET request to the URL `/collections/{collectionId}/items/{recordId}` where `{collectionId}` is the id property for a collection, where the the `itemType` property is a string with the value `records`, and `{recordId}` is the id property of the record. 2. Check that the value of the returned HTTP status header is 200. 3. Validate the contents of the returned records using the test /req/records-api/record-response.

ABSTRACT TEST A.29

IDENTIFIER	`/conf/records-api/record-response`
REQUIREMENT	Requirement 42: `/req/records-api/record-response`
TEST PURPOSE	Validate a record response from the API.
TEST METHOD	1. Construct a path (`/collections/{catalogId}/items/{recordId}`) to retrieve records from a catalog. 2. Issue an HTTP GET request on that path and negotiate a GeoJSON (accept: `application/geo+json application=ogc-record`) response.

ABSTRACT TEST A.29

3. Check that the value of the returned HTTP status header is 200.
4. Validate that the value of the `Content-Type` HTTP header is `application/geo+json application=ogc-record`.
5. Validate the content of the response using the schema recordGeoJSON.yaml.
6. Verify that all links in the `links` section include a `rel` parameter.
7. Check that the `links` section of the record contains the following links:
   - a link to this response document (relation: `self`),
   - a typed link to the response document in every other media type supported by the service (relation: `alternate`), and
   - a link to the catalog that contains this record (relation: `collection`).

ABSTRACT TEST A.30

IDENTIFIER `/conf/records-api/records-op`

REQUIREMENT Requirement 38: `/req/records-api/records-op`

TEST PURPOSE Validate that records can be retrieved from a catalog using query parameters.

TEST METHOD

1. Construct a path to retrieve records from a catalog (path: `/collections/{collectionId}/items`).
2. The parameter `collectionId` is each `id` property in the collections response where the the `itemsType` property is a string with the value `record`.
3. Demonstrate compliance with test /conf/core/fc-op.
4. In addition to the parameters tested in /conf/core/fc-op, also test the following parameters:

```
External Ids:
```

Figure A.1

- Parameter /conf/record-core-query-parameters/externalIds-definition
- Response /conf/record-core-query-parameters/externalIds-response

```
Ids:
```

Figure A.2

- Parameter /conf/record-core-query-parameters/ids-definition
- Response /conf/record-core-query-parameters/ids-response

```
Keyword search:
```

Figure A.3

- Parameter /conf/record-core-query-parameters/q-definition

ABSTRACT TEST A.30

- Response /conf/record-core-query-parameters/q-response

  Type:

  Figure A.4

- Parameter /conf/record-core-query-parameters/type-definition
- Response /conf/record-core-query-parameters/type-response

5. Execute requests with combinations of these query parameters and verify that only records that match all selection criteria are returned.

ABSTRACT TEST A.31

IDENTIFIER	`/conf/records-api/resource-name-mapping`
REQUIREMENT	Requirement 35: `/req/records-api/resource-name-mapping`
TEST PURPOSE	Validate the Records API implementation using Features API Standard with naming mapping.
TEST METHOD	1. Construct an API endpoint 2. Check if the API conforms with OGC API — Features — Part 1: Core Standard by replacing each reference to `features` or `feature` by the terms `records` or `record` when interpreting requirements from the OGC API — Features — Part 1: Core Standard. 3. Check the implementation of the Records API Standard with the Abstract Test Suite from the OGC API — Features — Part 1: Core Standard.

ABSTRACT TEST A.32

IDENTIFIER	`/conf/records-api/mandatory-params`
REQUIREMENT	Requirement 39: `/req/records-api/mandatory-params`
TEST PURPOSE	Validate each record containing the mandatory properties.
TEST METHOD	1. Construct a path to retrieve the server's API description. 2. Issue an HTTP GET operation on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Inspect the API description document and verify that the following parameters, listed in Table 12, are defined for the `/collections/{collectionId}/items` endpoint for all collections where the value of the `itemType` (JSONPath `$.collections[*].itemType`) property is `record`: • bbox • datetime

ABSTRACT TEST A.32	
	• req_record-core-query-parameters_externalIds-definition • ids • limit • q • type

CONFORMANCE CLASS A.4	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core-query-parameters`
REQUIREMENTS CLASS	Requirements class 3: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.33: `/conf/record-core-query-parameters/bbox` Abstract test A.34: `/conf/record-core-query-parameters/datetime` Abstract test A.35: `/conf/record-core-query-parameters/externalIds-definition` Abstract test A.36: `/conf/record-core-query-parameters/externalIds-response` Abstract test A.37: `/conf/record-core-query-parameters/ids-definition` Abstract test A.38: `/conf/record-core-query-parameters/ids-response` Abstract test A.39: `/conf/record-core-query-parameters/limit` Abstract test A.40: `/conf/record-core-query-parameters/q-definition` Abstract test A.41: `/conf/record-core-query-parameters/q-response` Abstract test A.42: `/conf/record-core-query-parameters/type-definition` Abstract test A.43: `/conf/record-core-query-parameters/type-response`

A.1.13. Query Parameters

ABSTRACT TEST A.33	
IDENTIFIER	`/conf/record-core-query-parameters/bbox`
REQUIREMENT	Requirement 23: `/req/record-core-query-parameters/bbox`
TEST PURPOSE	Validate `bbox` query parameter.
TEST METHOD	1. Construct a path for a searchable endpoint and include a `bbox` query parameter.

ABSTRACT TEST A.33

2. Validate that the `bbox` query parameter is constructed correctly using the test /conf/core/fc-bbox-definition.
3. Issue an HTTP GET request on that path.
4. Check that the value of the returned HTTP status header is 200.
5. Validate that the `bbox` query parameter is processed correctly using the test from /conf/core/fc-bbox-response.
6. All references to the term "features" must be replaced by the term "records" or "local resources" as the context may indicate

ABSTRACT TEST A.34

IDENTIFIER	`/conf/record-core-query-parameters/datetime`
REQUIREMENT	Requirement 24: `/req/record-core-query-parameters/datetime`
TEST PURPOSE	Validate `datetime` query parameter.
TEST METHOD	1. Construct a path for a searchable endpoint and include a `datetime` query parameter. 2. Validate that the `datetime` query parameter is constructed correctly using the test from /conf/core/fc-time-definition. 3. Issue an HTTP GET request on that path. 4. Check that the value of the returned HTTP status header is 200. 5. Validate that the `datetime` query parameter is processed correctly using the test /conf/core/fc-time-response. 6. All references to the term "features" must be replaced by the term "records" or "local resources" as the context may indicate

ABSTRACT TEST A.35

IDENTIFIER	`/conf/record-core-query-parameters/externalIds-definition`
REQUIREMENT	Requirement 32: `/req/record-core-query-parameters/externalIds-definition`
TEST PURPOSE	Validate `externalIds` query parameter.

TEST METHOD	1. Construct a path for a searchable endpoint and include the `externalIds` query parameter.
	2. Validate that the `externalIds` query parameter is constructed correctly using an OpenAPI Specification 3.0 fragment:
	`name: externalIds`
	`in: query`

ABSTRACT TEST A.35

```
required: false
schema:
  type: array
  items:
    type: string
explode: false
```

1. Issue an HTTP GET request on that path.
2. Check that the value of the returned HTTP status header is 200.

ABSTRACT TEST A.36

IDENTIFIER	`/conf/record-core-query-parameters/externalIds-response`
REQUIREMENT	Requirement 33: `/req/record-core-query-parameters/externalIds-response`
TEST PURPOSE	Validate `externalIds` query parameter.
TEST METHOD	1. Construct a path for a searchable endpoint as per test /conf/record-core-query-parameters/externalIds-definition. 2. Issue an HTTP GET request on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Check that only records that have an external identifier, as indicated by the values of the `externalIds` core queryable, equal to one of the listed values specified using the `externalIds` parameter are in the result set.

ABSTRACT TEST A.37

IDENTIFIER	`/conf/record-core-query-parameters/ids-definition`
REQUIREMENT	Requirement 30: `/req/record-core-query-parameters/ids-definition`
TEST PURPOSE	Validate `ids` query parameter.

TEST METHOD	1. Construct a path for a searchable endpoint and include the `ids` query parameter.
	2. Validate that the `ids` query parameter is constructed correctly using an OpenAPI Specification 3.0 fragment:
	
	`name: ids`
	`in: query`
	`required: false`
	`schema:`
	`  type: array`
	`  items:`
	`  type: string`
	`explode: false`

ABSTRACT TEST A.37

	1. Issue an HTTP GET request on that path. 2. Check that the value of the returned HTTP status header is 200.

ABSTRACT TEST A.38

IDENTIFIER	`/conf/record-core-query-parameters/ids-response`
REQUIREMENT	Requirement 31: `/req/record-core-query-parameters/ids-response`
TEST PURPOSE	Validate `ids` query parameter.
TEST METHOD	1. Construct a path for a searchable endpoint as per test /conf/record-core-query-parameters/ids-definition. 2. Issue an HTTP GET request on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Check that only records whose identifier match one of the identifiers specified using the `ids` query parameter are in the results set.

ABSTRACT TEST A.39

IDENTIFIER	`/conf/record-core-query-parameters/limit`
REQUIREMENT	Requirement 25: `/req/record-core-query-parameters/limit`
TEST PURPOSE	Validate `limit` query parameter.
TEST METHOD	1. Construct a path for a searchable endpoint and include a `limit` query parameter. 2. Validate that the limit query parameters are constructed correctly using the test from /conf/core/fc-limit-definition. 3. Issue an HTTP GET request on that path. 4. Check that the value of the returned HTTP status header is 200. 5. Validate that the server responds correctly using the test /conf/core/fc-limit-response. 6. All references to the term “features” must be replaced by the term “records” or “local resources” as the context may indicate

ABSTRACT TEST A.40

IDENTIFIER	`/conf/record-core-query-parameters/q-definition`

ABSTRACT TEST A.40

REQUIREMENT	Requirement 26: `/req/record-core-query-parameters/q-definition`
TEST PURPOSE	Validate q query parameter.

TEST METHOD	1. Construct a path for a searchable endpoint and include the q query parameter.
	2. Validate that the q parameter is constructed correctly using an OpenAPI Specification 3.0 fragment:
	`name: q`
	`in: query`
	`required: false`
	`schema:`
	`  type: array`
	`  items:`
	`  type: string`
	`explode: false`
	1. Issue an HTTP GET request on that path.
	2. Check that the value of the returned HTTP status header is 200.

ABSTRACT TEST A.41

IDENTIFIER	`/conf/record-core-query-parameters/q-response`
REQUIREMENT	Requirement 27: `/req/record-core-query-parameters/q-response`
TEST PURPOSE	Validate q query parameter.
TEST METHOD	1. Construct a search path with a single search term as per test /conf/record-core-query-parameters/q-definition. 2. Issue an HTTP GET request on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Check that only records that contain that search term in one or more of the searched text fields are in the result set. 5. Construct a search path with multiple search terms that are comma separated (logical OR) as per test /conf/record-core-query-parameters/q-definition. 6. Issue an HTTP GET request on that path. 7. Check that the value of the returned HTTP status header is 200. 8. Check that only records that contain one or more the specified search terms in one or more of the searched text fields are in the result set. 9. Construct a search path with multiple search terms that are white space separated as per test /conf/record-core-query-parameters/q-definition. 10. Issue an HTTP GET request on that path.

ABSTRACT TEST A.41

11. Check that the value of the returned HTTP status header is 200.

12. Check that only records that contain all the search terms specified, in the order specified and separated by any number of white spaces in one or more of the searched text fields are in the result set.

ABSTRACT TEST A.42

IDENTIFIER `/conf/record-core-query-parameters/type-definition`

REQUIREMENT Requirement 28: `/req/record-core-query-parameters/type-definition`

TEST PURPOSE Validate `type` query parameter.

TEST METHOD

1. Construct a path for a searchable endpoint and include the `type` query parameter.

2. Validate that the `type` query parameter is constructed correctly using an OpenAPI Specification 3.0 fragment:

```
name: type
in: query
required: false
schema:
  type: array
  items:
    type: string
    maxLength: 64
explode: false
```

1. Issue an HTTP GET request on the path

2. Check that the value of the returned HTTP status header is 200.

ABSTRACT TEST A.43

IDENTIFIER `/conf/record-core-query-parameters/type-response`

REQUIREMENT Requirement 29: `/req/record-core-query-parameters/type-response`

TEST PURPOSE Validate `type` query parameter.

TEST METHOD

1. Construct a path for a searchable endpoint as per test /conf/record-core-query-parameters/type-definition.

2. Issue an HTTP GET request on that path.

3. Check that the value of the returned HTTP status header is 200.

4. Check that only records whose `type`, as indicated by the value of the `type` core queryable, is equal to one of the listed values specified using the `type` parameter are in the result set.

CONFORMANCE CLASS A.5	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/query-param-profile`
REQUIREMENTS CLASS	Requirements class 19: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/query-param-profile`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.44: `/conf/query-param-profile/definition` Abstract test A.45: `/conf/query-param-profile/response`

A.1.14. Query Parameter Profile

ABSTRACT TEST A.44	
IDENTIFIER	`/conf/query-param-profile/definition`
REQUIREMENT	Requirement 94: `/req/query-param-profile/definition`
TEST PURPOSE	Validate `profile` query parameter.

TEST METHOD	1. Construct a path for a searchable endpoint and include the `profile` query parameter.
	2. Validate that the `profile` query parameter is constructed correctly using an OpenAPI Specification 3.0 fragment:
	`name: profile`
	`in: query`
	`required: false`
	`schema:`
	`  type: array`
	`  items:`
	`  type: string`
	`explode: false`
	`style: form`
	1. Issue an HTTP GET request on that path.
	2. Check that the value of the returned HTTP status header is 200.

ABSTRACT TEST A.45	
IDENTIFIER	`/conf/query-param-profile/response`
REQUIREMENT	Requirement 95: `/req/query-param-profile/response`

ABSTRACT TEST A.45	
TEST PURPOSE	Validate `profile` query parameter.
TEST METHOD	1. Construct a path for a searchable endpoint as per test /conf/query-param-profile/ definition. 2. Issue an HTTP GET request on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Check that that the content of response adheres to the requirements of the specified profile(s).

CONFORMANCE CLASS A.6	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/cql-filter`
REQUIREMENTS CLASS	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/cql-filter
PREREQUISITES	OGC API — Features — Part 3: Filtering Common Query Language (CQL2)
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.46: `/conf/record-filter/filter-param` Abstract test A.47: `/conf/record-filter/filter-lang-param` Abstract test A.48: `/conf/record-filter/filter-crs-param` Abstract test A.49: `/conf/record-filter/response`

A.1.15. Filter Parameters

ABSTRACT TEST A.46	
IDENTIFIER	`/conf/record-filter/filter-param`
REQUIREMENT	/req/record-filter/filter-param
TEST PURPOSE	Validate `filter` query parameter.
TEST METHOD	1. Construct a path for a searchable endpoint and include a `filter` query parameter. That may optionally include a `filter-crs` and `filter-lang` parameter. 2. Validate that the `filter` query parameter is constructed correctly as per /req/filter/filter-param. 3. Issue an HTTP GET request on that path.

ABSTRACT TEST A.46	
	4. Check that the value of the returned HTTP status header is 200.

ABSTRACT TEST A.47	
IDENTIFIER	`/conf/record-filter/filter-lang-param`
REQUIREMENT	Requirement 49: `/req/record-filter/filter-lang-param`
TEST PURPOSE	Validate `filter-lang` query parameter.
TEST METHOD	1. Construct a path for a searchable endpoint and include a `filter-lang` query parameter. 2. Validate that the `filter-lang` query parameter is constructed correctly as per /req/filter/filter-lang-param. 3. Issue an HTTP GET request on that path. 4. Check that the value of the returned HTTP status header is 200.

ABSTRACT TEST A.48	
IDENTIFIER	`/conf/record-filter/filter-crs-param`
REQUIREMENT	Requirement 50: `/req/record-filter/filter-crs-param`
TEST PURPOSE	Validate `filter-crs` query parameter.
TEST METHOD	1. Construct a path for a searchable endpoint and include a `filter-crs` query parameter. 2. Validate that the `filter-crs` query parameter is constructed correctly as per /req/filter/filter-crs-param. 3. Issue an HTTP GET request on that path. 4. Check that the value of the returned HTTP status header is 200.

A.1.16. Filter Response

ABSTRACT TEST A.49	
IDENTIFIER	`/conf/record-filter/response`
REQUIREMENT	Requirement 51: `/req/record-filter/response`

ABSTRACT TEST A.49	
TEST PURPOSE	Validate filter response.
TEST METHOD	1. Construct a path for a searchable endpoint and include the `filter` query parameter. The path may optionally include the `filter-lang` and `filter-crs` query parmeters. 2. Issue an HTTP GET on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Validate that the filter query parameters are processed correctly as per /req/filter/response. 5. All references to the term “features” must be replaced by the term “records” or “local resources” as the context may indicate

CONFORMANCE CLASS A.7	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/sorting`
REQUIREMENTS CLASS	Requirements class 5: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/sorting`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.50: `/conf/sorting/conformance` Abstract test A.51: `/conf/sorting/defaultSortOrder-definition` Abstract test A.52: `/conf/sorting/get-sortables-op` Abstract test A.53: `/conf/sorting/get-sortables-success` Abstract test A.54: `/conf/sorting/sortby-definition` Abstract test A.55: `/conf/sorting/sortby-response`

A.1.17. Conformance

ABSTRACT TEST A.50	
IDENTIFIER	`/conf/sorting/conformance`
REQUIREMENT	/req/sorting/conformance
TEST PURPOSE	Validate conformance identification.
TEST METHOD	1. Construct a path for a conformance page. 2. Issue an HTTP GET request on that path. 3. Check that the `conformsTo` array contains the value `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/sorting`.

A.1.18. Default sort order definition

ABSTRACT TEST A.51	
IDENTIFIER	`/conf/sorting/defaultSortOrder-definition`
REQUIREMENT	Requirement 47: `/req/sorting/defaultSortOrder-definition`
TEST PURPOSE	Validate the behavior of sort order definitions.
TEST METHOD	Sequence 1: 1. Construct a path for a collection page. 2. Issue an HTTP GET request on the path. 3. Check that the `defaultSortOrder` attribute exists. 4. Construct a path for collection sortables. 5. Issue an HTTP GET request on the path 6. For each sortable found in the `defaultSortOrder` array, check that it is present in the results of the sortables result. Sequence 2: 1. Construct a path for a collection page. 2. Issue an HTTP GET request on the path. 3. Check that the `defaultSortOrder` attribute exists. 4. Using one sortable, construct a path for an `.../items` query with `sortby` using explicit ascending order (+). 5. Issue an HTTP GET request on the path. 6. Parse the response to capture the sortable property for each record returned. 7. Verify that sorting the list of sortable properties in ascending order is consistent with the response of the server. 8. Using one sortable, construct a path for an `.../items` query with `sortby` using descending order (-). 9. Issue an HTTP GET request on the path. 10. Parse the response to capture the sortable property for each record returned. 11. Verify that reverse sorting the list of sortable properties in descending order is consistent with the response of the server.

ABSTRACT TEST A.51	
	Sequence 3: 1. Construct a path for a collection page. 2. Issue an HTTP GET request on the path. 3. Check that the `defaultSortOrder` attribute exists. 4. Using one sortable, construct a path for an `.../items` query with `sortby` using explicit ascending order (+). 5. Issue an HTTP GET request on the path. 6. Using the same sortable, construct a path for an `.../items` query with `sortby` using implicit ascending order (i.e., no +). 7. Issue an HTTP GET request on the path. 8. Verify that the records returned in both responses are identical.

A.1.19. Sortables endpoint

ABSTRACT TEST A.52	
IDENTIFIER	`/conf/sorting/get-sortables-op`
REQUIREMENT	Requirement 45: `/req/sorting/get-sortables-op`
TEST PURPOSE	Validate the behavior of `sortables` endpoint.
TEST METHOD	1. Construct a path for a collection page. 2. Issue an HTTP GET request on the path. 3. Parse all link objects from collection objects whose `itemType` value is `record`. 4. Check that at least one link contains a rel of http://www.opengis.net/def/rel/ogc/1.0/sortables with a media type of `application/schema+json`. 5. Use the `href` attribute to construct a path for collection sortables page. 6. Issue an HTTP GET request on the path 7. For each sortable found in the `defaultSortOrder` array, check that it is present in the results of the sortables result.

ABSTRACT TEST A.53	
IDENTIFIER	`/conf/sorting/get-sortables-success`

ABSTRACT TEST A.53	
REQUIREMENT	Requirement 46: `/req/sorting/get-sortables-success`
TEST PURPOSE	Validate the behavior of `sortables` endpoint.
TEST METHOD	1. Construct a path for a collection sortables page. 2. Issue an HTTP GET request on the path. 3. Check that the value of the returned HTTP status header is 200. 4. If the HTTP Content Type is `application/schema+json`: 5. Check that the `$schema` attribute exists with a value of `http://json-schema.org/draft-07/schema#` or `https://json-schema.org/draft/2019-09/schema`. 6. Check that the `$id` attribute exists with a value of the name of the collection. 7. Check that the `type` attribute exists with a value of `object`. 8. Construct a path for a collections `.../items` query. 9. Issue an HTTP GET request on the path. 10. Check that each sortable is available in the record's `properties` object.

A.1.20. Query parameter definition

ABSTRACT TEST A.54	
IDENTIFIER	`/conf/sorting/sortby-definition`
REQUIREMENT	/req/sorting/sortby-definition
TEST PURPOSE	Validate the availability of the `sortby` query parameter.
TEST METHOD	1. Construct a path for a landing page. 2. Issue an HTTP GET request on the path. 3. Parse all link objects 4. Find the link object `href` attribute whose link object's `rel` attribute value is `service-desc` and `type` attribute value is `application/vnd.oai.openapi+json;version=3.0`. 5. Issue an HTTP GET request on the `href`. 6. Parse the resulting OpenAPI document and detect a `path` object whose endpoint value ends with +…/`items`. 7. Check that the `get.parameters` object contains an object defining a `sortby` parameter.

A.1.21. Query parameter behavior

ABSTRACT TEST A.55	
IDENTIFIER	`/conf/sorting/sortby-response`
REQUIREMENT	Requirement 44: `/req/sorting/sortby-response`
TEST PURPOSE	Validate the availability of the `sortby` query parameter.
TEST METHOD	Sequence 1: 1. Construct a path for a collection page. 2. Issue an HTTP GET request on the path. 3. Check that the `defaultSortOrder` attribute exists. 4. Construct a path for collection sortables. 5. Issue an HTTP GET request on the path 6. For each sortable found in the `defaultSortOrder` array, check that it is present in the results of the sortables result. Sequence 2: 1. Construct a path for a collection page. 2. Issue an HTTP GET request on the path. 3. Check that the `defaultSortOrder` attribute exists. 4. Using one sortable, construct a path for an `.../items` query with `sortby` using explicit ascending order (+). 5. Issue an HTTP GET request on the path. 6. Parse the response to capture the sortable property for each record returned. 7. Verify that sorting the list of sortable properties in ascending order is consistent with the response of the server. 8. Using one sortable, construct a path for an `.../items` query with `sortby` using descending order (–). 9. Issue an HTTP GET request on the path. 10. Parse the response to capture the sortable property for each record returned. 11. Verify that reverse sorting the list of sortable properties in descending order is consistent with the response of the server.

ABSTRACT TEST A.55	
	Sequence 3: 1. Construct a path for a collection page. 2. Check that the specific set of keys that may be used for sorting are specified by the `/collections/{collectionId}/sortables` resource

CONFORMANCE CLASS A.8	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/html`
REQUIREMENTS CLASS	Requirements class 9: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/html`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.56: `/conf/html/conformance` Abstract test A.57: `/conf/html/content` Abstract test A.58: `/conf/html/definition`

A.1.22. Conformance

ABSTRACT TEST A.56	
IDENTIFIER	`/conf/html/conformance`
REQUIREMENT	Requirement 60: `/req/html/conformance`
TEST PURPOSE	Validate html identification.
TEST METHOD	1. Construct a path for a conformance page. 2. Issue an HTTP GET request on that path. 3. Check that the `conformsTo` array contains the value `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/html`.

A.1.23. API

ABSTRACT TEST A.57	
IDENTIFIER	`/conf/html/content`

ABSTRACT TEST A.57

REQUIREMENT	Requirement 62: `/req/html/content`
TEST PURPOSE	Validate the HTML response content.
TEST METHOD	1. Construct a path for a `collections/items` page. 2. Issue an HTTP GET request on the path 3. Check that the value of the returned HTTP status header is 200. 4. Check that the HTTP Content-Type header value is `text/html`. 5. Check that the response validates as an HTML 5 document 6. Using the `id` of a given record, construct a path for a `collections/items/<featureId>` page. 7. Issue an HTTP GET request on the path 8. Check that the value of the returned HTTP status header is 200. 9. Check that the HTTP Content-Type header value is `text/html`. 10. Check that the response validates as an HTML 5 document 11. Check that all information identified in the schemas of the Response Object exists in the HTML `<body>` and all links in HTML `<a>` elements in the HTML `<body>`

ABSTRACT TEST A.58

IDENTIFIER	`/conf/html/definition`
REQUIREMENT	Requirement 61: `/req/html/definition`
TEST PURPOSE	Validate the content type of HTML response.
TEST METHOD	1. Construct a path for each server operation. 2. Issue an HTTP GET request on the path 3. Check that the value of the returned HTTP status header is `200`. 4. Check that the HTTP Content-Type header value is `text/html`.

CONFORMANCE CLASS A.9

IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/json`
REQUIREMENTS CLASS	Requirements class 8: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/json`

CONFORMANCE CLASS A.9	
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.59: `/conf/json/conformance` Abstract test A.60: `/conf/json/catalog-content` Abstract test A.62: `/conf/json/catalog-response` Abstract test A.63: `/conf/json/record-content` Abstract test A.65: `/conf/json/record-response`

A.1.24. Conformance

ABSTRACT TEST A.59	
IDENTIFIER	`/conf/json/conformance`
REQUIREMENT	Requirement 53: `/req/json/conformance`
TEST PURPOSE	Validate conformance identification.
TEST METHOD	1. Construct a path for a conformance page. 2. Issue an HTTP GET request on that path. 3. Check that the `conformsTo` array contains the value `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/json`.

A.1.25. Catalog

ABSTRACT TEST A.60	
IDENTIFIER	`/conf/json/catalog-content`
REQUIREMENT	Requirement 58: `/req/json/catalog-content`
TEST PURPOSE	Validate the JSON representation of a catalog.
TEST METHOD	1. Construct a path for a `catalog` page. 2. Negotiate for a JSON-encoded catalog response. 3. Issue an HTTP GET request on the path 4. Check that the media type of the response is `application/ogc-catalog+json`

ABSTRACT TEST A.60

	5. Check that the response validates against the schema as defined in https://schemas.opengis.net/ogcapi/records/part1/1.0/openapi/schemas/catalog.yaml. 6. Check that the `links` section includes a link (href: `http://www.opengis.net/def/profile/OGC/0/ogc-catalog`, relation: `profile`).

ABSTRACT TEST A.61

IDENTIFIER	`/conf/json/catalog-content-profile`
REQUIREMENT	Requirement 59: `/req/json/catalog-content-profile`
TEST PURPOSE	Validate the JSON representation of a catalog.
TEST METHOD	1. Construct a path for a `catalog` page. 2. Negoation for a JSON-encoded catalog response. 3. Negotiate for the profile `http://www.opengis.net/def/profile/OGC/0/ogc-catalog`. 4. Issue an HTTP GET request on the path 5. Check that the media type of the response is `application/json`. 6. Check that the response validates against the schema as defined in https://schemas.opengis.net/ogcapi/records/part1/1.0/openapi/schemas/catalog.yaml. 7. Check that the `links` section includes a link (href: `http://www.opengis.net/def/profile/OGC/0/ogc-catalog`, relation: `profile`).

ABSTRACT TEST A.62

IDENTIFIER	`/conf/json/catalog-response`
REQUIREMENT	/req/json/catalog-response
TEST PURPOSE	Validate the JSON response of a catalog.
TEST METHOD	1. Construct a path for a `catalog` page. 2. Issue an HTTP GET request on the path 3. Check that the value of the returned HTTP status header is `200`. 4. Check that the HTTP Content-Type header value is `application/ogc-catalog+json`.

A.1.26. Record

ABSTRACT TEST A.63

IDENTIFIER	`/conf/json/record-content`
REQUIREMENT	Requirement 55: `/req/json/record-content`
TEST PURPOSE	Validate the behavior of link relations for record geojson.
TEST METHOD	1. Construct a path for a `collections/items` page. 2. Negotiate for a JSON-encoded record response. 3. Issue an HTTP GET request on the path 4. Check that the value of the returned HTTP status header is 200. 5. Check that the HTTP Content-Type header value is `application/geo+json`. 6. Check that the response validates against as defined in https://schemas.opengis.net/ogcapi/records/part1/1.0/openapi/schemas/featureCollectionGeoJSON.yaml 7. Using the `id` of a given record, construct a path for a `collections/items/<featureId>` page. 8. Issue an HTTP GET request on the path 9. Check that the value of the returned HTTP status header is 200. 10. Check that the HTTP Content-Type header value is `application/geo+json`. 11. Check that the response contains a `links` array. 12. Check that the response validates against as defined in https://schemas.opengis.net/ogcapi/records/part1/1.0/openapi/schemas/recordGeoJSON.yaml. 13. Check that the `links` section include a link (href: `http://www.opengis.net/def/profile/OGC/0/ogc-record`, relation: `profile`).

ABSTRACT TEST A.64

IDENTIFIER	`/conf/json/record-content-profile`
REQUIREMENT	Requirement 56: `/req/json/record-content-profile`
TEST PURPOSE	Validate the behavior of link relations for record geojson.
TEST METHOD	1. Construct a path for a `collections/items` page. 2. Negoation for a JSON-encoded record response.

ABSTRACT TEST A.64

3. Issue an HTTP GET request on the path
4. Check that the value of the returned HTTP status header is 200.
5. Check that the HTTP Content-Type header value is `application/geo+json`.
6. Check that the response validates against as defined in https://schemas.opengis.net/ogcapi/records/part1/1.0/openapi/schemas/featureCollectionGeoJSON.yaml
7. Using the `id` of a given record, construct a path for a `collections/items/<featureId>` page.
8. Issue an HTTP GET request on the path
9. Check that the value of the returned HTTP status header is 200.
10. Check that the HTTP Content-Type header value is `application/geo+json`.
11. Check that the response contains a `links` array.
12. Check that the response validates against as defined in https://schemas.opengis.net/ogcapi/records/part1/1.0/openapi/schemas/recordGeoJSON.yaml.
13. Check that the `links` section includes a link (href: `http://www.opengis.net/def/profile/OGC/0/ogc-record`, relation: `profile`).

ABSTRACT TEST A.65

IDENTIFIER	`/conf/json/record-response`
REQUIREMENT	Requirement 54: `/req/json/record-response`
TEST PURPOSE	Validate the JSON response of a record.
TEST METHOD	1. Construct a path for one or more `records`. 2. Issue an HTTP GET request on the path 3. Check that the value of the returned HTTP status header is 200. 4. Check that the HTTP Content-Type header value is `application/geo+json`. 5. Check that the response validates against as defined in https://schemas.opengis.net/ogcapi/records/part1/1.0/openapi/schemas/recordGeoJSON.yaml.

CONFORMANCE CLASS A.10

IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/oas30`
REQUIREMENTS CLASS	Requirements class 10: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/oas30`

CONFORMANCE CLASS A.10	
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.66: `/conf/oas30/conformance` Abstract test A.67: `/conf/oas30/oas-common`

A.1.27. Conformance

ABSTRACT TEST A.66	
IDENTIFIER	`/conf/oas30/conformance`
REQUIREMENT	Requirement 64: `/req/oas30/conformance`
TEST PURPOSE	Validate conformance identification.
TEST METHOD	1. Construct a path for a conformance page. 2. Issue an HTTP GET request on that path. 3. Check that the `conformsTo` array contains the value `http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/oas30`.

A.1.28. API description document

ABSTRACT TEST A.67	
IDENTIFIER	`/conf/oas30/oas-common`
REQUIREMENT	Requirement 63: `/req/oas30/oas-common`
TEST PURPOSE	Validate oas30 service description document.
TEST METHOD	1. Verify that the server passes abstract tests /conf/oas30/completeness, /conf/oas30/exceptions-codes, /conf/oas30/oas-definition-1, /conf/oas30/oas-definition-2, /conf/oas30/oas-impl, /conf/oas30/security from OGC API — Features — Part 1: Core/

CONFORMANCE CLASS A.11	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/autodiscovery`

CONFORMANCE CLASS A.11	
REQUIREMENTS CLASS	Requirements class 7: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/autodiscovery`
TARGET TYPE	Web API
CONFORMANCE TEST	Abstract test A.68: `/conf/autodiscovery/links`

A.1.29. Links

ABSTRACT TEST A.68	
IDENTIFIER	`/conf/autodiscovery/links`
REQUIREMENT	Requirement 52: `/req/autodiscovery/links`
TEST PURPOSE	Validate the behavior of link relations for searchable catalog autodiscovery.
TEST METHOD	1. Construct a path for the landing page of an OGC API deployment. 2. Issue an HTTP GET request on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Parse all link objects. 5. Check that at least one link contains a `rel` of `http://www.opengis.net/def/rel/ogc/1.0/ogc-catalog`. 6. Check that the `type` attribute exists with a valid media type. 7. Check that the `href` attribute exists. 8. Issue an HTTP GET request on the value of the `href` attribute. 9. Validate that the link resolves with the associated media type identified in the `type` attribute.

A.2. Crawlable catalog

CONFORMANCE CLASS A.12	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/crawlable-catalog`

CONFORMANCE CLASS A.12	
REQUIREMENTS CLASS	Requirements class 11: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/crawlable-catalog`
PREREQUISITES	Conformance class A.2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-collection` Conformance class A.1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.69: `/conf/crawlable-catalog/conformance` Abstract test A.70: `/conf/crawlable-catalog/record` Abstract test A.71: `/conf/crawlable-catalog/record-location` Abstract test A.72: `/conf/crawlable-catalog/catalog` Abstract test A.73: `/conf/crawlable-catalog/location`

A.2.1. Conformance

ABSTRACT TEST A.69	
IDENTIFIER	`/conf/crawlable-catalog/conformance`
REQUIREMENT	Validate conformance identification.
TEST PURPOSE	/req/crawlable-catalog/conformance
TEST METHOD	1. Obtain the path to a catalog 2. Issue an HTTP GET request on the path 3. Check that the `conformsTo` array contains the value `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/crawlable-catalog`.

A.2.2. Record

ABSTRACT TEST A.70	
IDENTIFIER	`/conf/crawlable-catalog/record`
REQUIREMENT	Requirement 65: `/req/crawlable-catalog/record`
TEST PURPOSE	Verify that a record is retrieveble from the Web

ABSTRACT TEST A.70	
TEST METHOD	1. Determine the URL of a record 2. Resolve the URL and retrieve the record 3. Check that the record validates against the schema defined in https://schemas.opengis.net/ogcapi/records/part1/1.0/openapi/schemas/recordGeoJSON.yaml 4. Locate each link (relation: `describes`) in the links or linkTemplates section of the record and verify that each link target is retrieveable

ABSTRACT TEST A.71	
IDENTIFIER	`/conf/crawlable-catalog/record-location`
REQUIREMENT	Requirement 66: `/req/crawlable-catalog/record-location`
TEST PURPOSE	Validate a crawlable catalog.
TEST METHOD	1. Determine the URL of a record 2. Verify that the URL can be resolved and retrieve the record

A.2.3. Catalog

ABSTRACT TEST A.72	
IDENTIFIER	`/conf/crawlable-catalog/catalog`
REQUIREMENT	Requirement 67: `/req/crawlable-catalog/catalog`
TEST PURPOSE	Validate a crawlable catalog object
TEST METHOD	1. Obtain a catalog object 2. Check that the catalog object validates against the schema defined in https://schemas.opengis.net/ogcapi/records/part1/1.0/openapi/schemas/catalog.yaml

ABSTRACT TEST A.73	
IDENTIFIER	`/conf/crawlable-catalog/location`
REQUIREMENT	Requirement 68: `/req/crawlable-catalog/location`

ABSTRACT TEST A.73	
TEST PURPOSE	Verify that a catalog is retrievable from the Web
TEST METHOD	1. Determine the URL of the crawlable catalog. 2. Verify that the URL can be resolved and retrieve the catalog object.

A.3. Local Resources Catalog

CONFORMANCE CLASS A.13	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog`
REQUIREMENTS CLASS	Requirements class 15: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog`
PREREQUISITES	Conformance class A.1: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core` Conformance class A.2: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-collection` Conformance class A.11: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/autodiscovery`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.74: `/conf/local-resources-catalog/conformance` Abstract test A.75: `/conf/local-resources-catalog/building-blocks` Abstract test A.76: `/conf/local-resources-catalog/discovery` Abstract test A.77: `/conf/local-resources-catalog/mandatory-catalog-properties` Abstract test A.78: `/conf/local-resources-catalog/mandatory-record-properties`

A.3.1. Conformance

ABSTRACT TEST A.74	
IDENTIFIER	`/conf/local-resources-catalog/conformance`
REQUIREMENT	Requirement 80: `/req/local-resources-catalog/conformance`

ABSTRACT TEST A.74	
TEST PURPOSE	Validate conformance identification.
TEST METHOD	1. Construct a path for a conformance page. 2. Issue an HTTP GET request on that path. 3. Check that the `conformsTo` array contains the value `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog`. 4. If the server supports JSON responses, check that the `conformsTo` array contains the value `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/json`. 5. If the server supports HTML responses, check that the `conformsTo` array contains the value `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/html`.

A.3.2. Local resources catalog

ABSTRACT TEST A.75	
IDENTIFIER	`/conf/local-resources-catalog/building-blocks`
REQUIREMENT	/req/local-resource-catalog/common
TEST PURPOSE	Validate conformance identification.
TEST METHOD	1. Demonstrate conformance to the following tests: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-collection • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/autodiscovery

A.3.3. Discovery

ABSTRACT TEST A.76	
IDENTIFIER	`/conf/local-resources-catalog/discovery`
REQUIREMENT	Requirement 83: `/req/local-resources-catalog/discovery`
TEST PURPOSE	Validate discovery identification.
TEST METHOD	1. Construct a path to retrieve a server's landing page. 2. Issue an HTTP GET request on that path.

ABSTRACT TEST A.76	
	3. Check that the value of the returned HTTP status header is 200. 4. Parse the links section of the landing page and identify all links with relation `http://www.opengis.net/def/rel/ogc/1.0/ogc-catalog`. 5. Construct a path to retrieve the server's API description document. 6. Issue an HTTP GET request on that path. 7. Check that the value of the returned HTTP status header is 200. 8. Inspect the API description document an verify that all the endpoints identified by links with relation `http://www.opengis.net/def/rel/ogc/1.0/ogc-catalog` are also defined as endpoints in the server's API description document.

A.3.4. Mandatory catalog properties

ABSTRACT TEST A.77	
IDENTIFIER	`/conf/local-resources-catalog/mandatory-catalog-properties`
REQUIREMENT	Requirement 81: `/req/local-resources-catalog/mandatory-catalog-properties`
TEST PURPOSE	Verify that all mandatory catalog properties are present.
TEST METHOD	1. Construct a path for a local resources catalog page. 2. Issue an HTTP GET request on the path 3. Check that the value of the returned HTTP status header is 200. 4. Check that the response conforms to the requirements of the local resource being extended. 5. Check that the response includes all the mandatory properties listed in the Table 11.

A.3.5. Mandatory record properties

ABSTRACT TEST A.78	
IDENTIFIER	`/conf/local-resources-catalog/mandatory-record-properties`
REQUIREMENT	Requirement 82: `/req/local-resources-catalog/mandatory-record-properties`
TEST PURPOSE	Validate conformance identification.

ABSTRACT TEST A.78	
TEST METHOD	1. Construct a path for a record from local resources catalog. 2. Issue an HTTP GET request on the path 3. Check that the value of the returned HTTP status header is 200. 4. Check that the response conforms to the requirements of the local resource object (e.g., /collection/{collectionId}) extended. 5. Check that the response includes all the mandatory properties listed in Table 8 and Table 9

CONFORMANCE CLASS A.14	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resource-catalog/query-parameters`
REQUIREMENTS CLASS	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resource-catalog/query-parameters
PREREQUISITES	Conformance class A.13: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog` Conformance class A.4: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core-query-parameters`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.79: `/conf/local-resources-catalog/query-parameters/conformance` Abstract test A.80: `/conf/local-resources-catalog/query-parameters/query-parameters` Abstract test A.81: `/conf/local-resources-catalog/query-parameters/query-parameters-response`

A.3.6. Conformance

ABSTRACT TEST A.79	
IDENTIFIER	`/conf/local-resources-catalog/query-parameters/conformance`
REQUIREMENT	/req/local-resources-catalog/query-parameters/conformance
TEST PURPOSE	Validate conformance identification.
TEST METHOD	1. Construct a path for a conformance page. 2. Issue an HTTP GET request on that path.

ABSTRACT TEST A.79	
	3. Check that the `conformsTo` array contains the value `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog-query-parameters`.

A.3.7. Query Parameters

ABSTRACT TEST A.80	
IDENTIFIER	`/conf/local-resources-catalog/query-parameters/query-parameters`
REQUIREMENT	/req/local-resources-catalog/query-parameters/query-parameters
TEST PURPOSE	Determine the set of supported query parameters.
TEST METHOD	1. Fetch the server's API document (e.g., path: `/api`). 2. Inspect the API document and verify that one or more of the bbox, datetime, q, type, externalId and any additional query parameters are supported.

A.3.8. Response

ABSTRACT TEST A.81	
IDENTIFIER	`/conf/local-resources-catalog/query-parameters/query-parameters-response`
REQUIREMENT	/req/local-resources-catalog/query-parameters/query-parameters-response
TEST PURPOSE	Validate query parameters response.
TEST METHOD	1. Validate via the server's API document which query parameters are supported (see test / conf/local-resources-catalog/query-parameters/query-parameters). 2. Construct a path for records from the local resource catalog and include one or more of the supported query parameters. 3. Issue an HTTP GET request on that path. 4. Check that all records in the response satisfy the query predicates specified using the query parameters.

CONFORMANCE CLASS A.15	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog-filtering`
REQUIREMENTS CLASS	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog-filtering
PREREQUISITES	Conformance class A.6: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/cql-filter` Conformance class A.13: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.82: `/conf/local-resources-catalog/filtering-conformance` Abstract test A.83: `/conf/local-resources-catalog/building-blocks/filtering` Abstract test A.84: `/conf/local-resources-catalog/filtering/queryables-link`

A.3.9. Conformance

ABSTRACT TEST A.82	
IDENTIFIER	`/conf/local-resources-catalog/filtering-conformance`
REQUIREMENT	/req/local-resources-catalog/filtering-conformance
TEST PURPOSE	Validate conformance identification.
TEST METHOD	1. Construct a path for a conformance page. 2. Issue an HTTP GET request on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Check that the `conformsTo` array contains the values: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog-filtering

A.3.10. Filtering

ABSTRACT TEST A.83	
IDENTIFIER	`/conf/local-resources-catalog/building-blocks/filtering`

ABSTRACT TEST A.83	
REQUIREMENT	Requirement 88: `/req/local-resources-catalog/filtering`
TEST PURPOSE	Validate local-resources catalog CQL filtering.
TEST METHOD	1. Demonstrate conformance to http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/cql-filter at the local resources catalog endpoint (e.g., `/collections`).

A.3.11. Queryables link

ABSTRACT TEST A.84	
IDENTIFIER	`/conf/local-resources-catalog/filtering/queryables-link`
REQUIREMENT	/req/local-resources-catalog/filtering/queryables-link
TEST PURPOSE	Validate queryables link for local resources catalog.
TEST METHOD	1. Construct a path to retrieve a local resources catalog. 2. Issue an HTTP GET request and retrieve the catalog object. 3. Check that the value of the returned HTTP status header is 200. 4. Parse the `links` section of the response. 5. Verify that a link with relation `http://www.opengis.net/def/rel/ogc/1.0/queryables` is present. 6. Issue an HTTP GET request on the `href` of that link. 7. Check that the value of the returned HTTP status header is 200.

CONFORMANCE CLASS A.16	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog/sorting`
REQUIREMENTS CLASS	Requirements class 18: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/local-resources-catalog/sorting`
PREREQUISITES	Conformance class A.13: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog` Conformance class A.7: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/sorting`
TARGET TYPE	Web API

CONFORMANCE CLASS A.16	
CONFORMANCE TESTS	Abstract test A.85: `/conf/local-resources-catalog/sorting-conformance` Abstract test A.86: `/conf/local-resources-catalog/building-blocks/sorting`

A.3.12. Conformance

ABSTRACT TEST A.85	
IDENTIFIER	`/conf/local-resources-catalog/sorting-conformance`
REQUIREMENT	Requirement 90: `/req/local-resources-catalog/sorting-conformance`
TEST PURPOSE	Validate conformance identification.
TEST METHOD	1. Construct a path for a conformance page. 2. Issue an HTTP GET request on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Check that the `conformsTo` array contains the values: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/local-resources-catalog-sorting

A.3.13. Sorting

ABSTRACT TEST A.86	
IDENTIFIER	`/conf/local-resources-catalog/building-blocks/sorting`
REQUIREMENT	Requirement 91: `/req/local-resources-catalog/sorting`
TEST PURPOSE	Validate sorting.
TEST METHOD	1. Demonstrate conformance to http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/sorting at the local resources catalog endpoint (e.g., `/collections`) using the test http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/sorting.

A.4. Searchable Catalog

CONFORMANCE CLASS A.17	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog`
REQUIREMENTS CLASS	Requirements class 12: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.87: `/conf/searchable-catalog/conformance` Abstract test A.88: `/conf/searchable-catalog/building-blocks` Abstract test A.89: `/conf/searchable-catalog/mandatory-catalog-properties` Abstract test A.90: `/conf/searchable-catalog/mandatory-record-properties`

A.4.1. Conformance

ABSTRACT TEST A.87	
IDENTIFIER	`/conf/searchable-catalog/conformance`
REQUIREMENT	Requirement 71: `/req/searchable-catalog/conformance`
TEST PURPOSE	Validate conformance identification.
TEST METHOD	1. Construct a path for a conformance page. 2. Issue an HTTP GET request on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Check that the `conformsTo` array contains the values: • http://www.opengis.net/spec/ogcapi-features-1/1.0/conf/core • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog 5. If the server supports JSON responses check that the `conformsTo` array contains the values: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/json 6. If the server supports HTML responses check that the `conformsTo` array contains the values: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/html 7. If the server supports an OpenAPI 3.0 service description document check that the `conformsTo` array contains the values: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/oas30

A.4.2. Searchable catalog

ABSTRACT TEST A.88	
IDENTIFIER	`/conf/searchable-catalog/building-blocks`
REQUIREMENT	Requirement 70: `/req/searchable-catalog/common`
TEST PURPOSE	Validate conformance identification.
TEST METHOD	1. Demonstrate conformance to http://www.opengis.net/spec/ogcapi-records-1/1.0/req/records-api using test http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/records-api. 2. Demonstrate conformance to http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core using test http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core. 3. Demonstrate conformance to http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-collection using test http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-collection. 4. Demonstrate conformance to http://www.opengis.net/spec/ogcapi-records-1/1.0/req/record-core-query-parameters using test http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/record-core-query-parameters.

A.4.3. Mandatory catalog properties

ABSTRACT TEST A.89	
IDENTIFIER	`/conf/searchable-catalog/mandatory-catalog-properties`
REQUIREMENT	Requirement 72: `/req/searchable-catalog/mandatory-catalog-properties`
TEST PURPOSE	Verify that all mandatory catalog properties are present.
TEST METHOD	1. Construct a path for retrieving a catalog description document (path: `/collections/{collectionId}`). 2. The parameter `collectionId` is each `id` property in the collections response (JSONPath: `$.collections[*].id`) where the `itemType` (JSONPath `$.collections[*].itemType`) property is a string and its value is `record`. 3. Issue a HTTP GET request on that path. 4. Parse each retrieved catalog description. 5. Check that each record includes all the mandatory properties listed in Table 11.

A.4.4. Mandatory record properties

ABSTRACT TEST A.90	
IDENTIFIER	`/conf/searchable-catalog/mandatory-record-properties`
REQUIREMENT	Requirement 73: `/req/searchable-catalog/mandatory-record-properties`
TEST PURPOSE	Validate each record contains the mandatory properties.
TEST METHOD	1. Construct a path for retrieving records from a searchable catalog endpoint (path: `/collections/{collectionId}/items`). 2. The parameter `collectionId` is each `id` property in the collections response (JSONPath: `$.collections[*].id`) where the `itemType` (JSONPath `$.collections[*].itemType`) property is a string and its value is `record`. 3. Issue a HTTP GET request on that path. 4. Parse each retrieved record. 5. Check that each record includes all the mandatory properties listed in Table 8 and Table 9.

CONFORMANCE CLASS A.18	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog/filtering`
REQUIREMENTS CLASS	Requirements class 13: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog/filtering`
PREREQUISITES	Conformance class A.17: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog` Conformance class A.6: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/cql-filter`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.91: `/conf/searchable-catalog/filtering-conformance` Abstract test A.92: `/conf/searchable-catalog/building-blocks/filtering` Abstract test A.93: `/conf/searchable-catalog/mandatory-queryables`

A.4.5. Conformance

ABSTRACT TEST A.91	
IDENTIFIER	`/conf/searchable-catalog/filtering-conformance`
REQUIREMENT	Requirement 75: `/req/searchable-catalog/filtering-conformance`
TEST PURPOSE	Validate conformance identification.
TEST METHOD	1. Construct a path for a conformance page. 2. Issue an HTTP GET request on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Check that the `conformsTo` array contains the values: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog-filtering

A.4.6. Filtering

ABSTRACT TEST A.92	
IDENTIFIER	`/conf/searchable-catalog/building-blocks/filtering`
REQUIREMENT	Requirement 74: `/req/searchable-catalog/filtering`
TEST PURPOSE	Validate enhanced filtering.
TEST METHOD	1. Demonstrate conformance to http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/cql-filter at the searchable catalog endpoint of `/collections/{collectionId}/items` where the `itemType` property of the collection with id `{collectionId}` (JSONPath: `$.collections[*].itemType`) is a string with the value `record`.

A.4.7. Mandatory queryables

ABSTRACT TEST A.93	
IDENTIFIER	`/conf/searchable-catalog/mandatory-queryables`
REQUIREMENT	Requirement 76: `/req/searchable-catalog/mandatory-queryables`
TEST PURPOSE	Validate existence of mandatory queryables.
TEST METHOD	1. Construct a path for a collection queryables page.

ABSTRACT TEST A.93	
	2. Issue an HTTP GET request on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Inspect the response and verify that the list of queryables includes all the mandatory properties listed in Table 8 and Table 9.

CONFORMANCE CLASS A.19	
IDENTIFIER	`http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog/sorting`
REQUIREMENTS CLASS	Requirements class 14: `http://www.opengis.net/spec/ogcapi-records-1/1.0/req/searchable-catalog/sorting`
PREREQUISITES	Conformance class A.17: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog` Conformance class A.7: `http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/sorting`
TARGET TYPE	Web API
CONFORMANCE TESTS	Abstract test A.94: `/conf/searchable-catalog/sorting-conformance` Abstract test A.95: `/conf/searchable-catalog/building-blocks/sorting`

A.4.8. Conformance

ABSTRACT TEST A.94	
IDENTIFIER	`/conf/searchable-catalog/sorting-conformance`
REQUIREMENT	Requirement 78: `/req/searchable-catalog/sorting-conformance`
TEST PURPOSE	Validate conformance identification.
TEST METHOD	1. Construct a path for a conformance page. 2. Issue an HTTP GET request on that path. 3. Check that the value of the returned HTTP status header is 200. 4. Check that the `conformsTo` array contains the values: • http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/searchable-catalog-sorting

A.4.9. Sorting

ABSTRACT TEST A.95	
IDENTIFIER	`/conf/searchable-catalog/building-blocks/sorting`
REQUIREMENT	Requirement 77: `/req/searchable-catalog/sorting`
TEST PURPOSE	Validate sorting.
TEST METHOD	1. Demonstrate conformance to http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/sorting at the searchable catalog endpoint of `/collections/{collectionId}/items` where the `itemType` property of the collection with id `{collectionId}` (JSONPath: `$.collections[*].itemType`) is a string with the value `record`.

B

ANNEX B (INFORMATIVE) IMPLEMENTATION SUMMARIES (INFORMATIVE)

ANNEX B (INFORMATIVE) IMPLEMENTATION SUMMARIES (INFORMATIVE)

B.1. Overview

This annex presents implementation summaries for the crawlable catalog searchable catalog and local resources catalog deployment patterns. This information is meant to be a short guide for developers.

B.2. Crawlable catalog

To implement a catalog where records are stored as individual files in one or more web-accessible locations and can be crawled by simply following hypermedia controls.

- For each discoverable resource create a file that contains a summary description of the resource using the record schema defined in Table 8 and Table 9.
- Extend the record schema to enrich its content as necessary.
- To make the resource discoverable, place each record file in a web-accessible location, preferably co-located (e.g., in the same directory or object store location) with the resource that the record is describing.
- Optionally create a file containing a description of a *collection* of related records.
- Also place the collection file in a web-accessible location.

B.3. Searchable catalog

In order to implement a catalog where records are stored in some backend database and are accessed through an access/search API:

- implement the OGC API — Features — Core: Part 1 Standard;
- implement the parameters defined in Table 12;
- implement the record schema defined in Table 8 and Table 9; and
- extend the record schema to enrich its content as necessary.

B.4. Local resources catalog

In order to implement a local resource catalog at an OGC API resources endpoint such as `/collections` for example:

- implement the endpoint as per the relevant OGC API Standard;
- enhance the information content of the resource using the mandatory and optional parameters defined in Table 9;
- implement the query parameters defined in Table 12; and
- optionally implement filtering, sorting.

C

ANNEX C (INFORMATIVE) COMMON RESOURCE TYPES (INFORMATIVE)

ANNEX C
(INFORMATIVE)
COMMON RESOURCE TYPES (INFORMATIVE)

C.1. Overview

This annex describes common resource types when qualifying resources via a record's `properties.type` property, in the absence of a formal controlled vocabulary. Please note that these codelists and identifiers are NOT normative. For information communities wishing to formally identify and qualify, it is suggested that a specification extension be developed to meet those requirements.

C.1.1. Data

- http://www.opengis.net/def/rel/ogc/1.0/dataset

C.1.2. Services

C.1.2.1. OGC Web Services

- http://www.opengis.net/def/serviceType/ogc/csw
- http://www.opengis.net/def/serviceType/ogc/wms
- http://www.opengis.net/def/serviceType/ogc/wfs
- http://www.opengis.net/def/serviceType/ogc/wcs
- http://www.opengis.net/def/serviceType/ogc/wps
- http://www.opengis.net/def/serviceType/ogc/sos

C.1.2.2. OGC API

- http://www.opengis.net/def/interface/ogcapi-records
- http://www.opengis.net/def/interface/ogcapi-features
- http://www.opengis.net/def/interface/ogcapi-coverages
- http://www.opengis.net/def/interface/ogcapi-maps
- http://www.opengis.net/def/interface/ogcapi-tiles
- http://www.opengis.net/def/interface/ogcapi-styles
- http://www.opengis.net/def/interface/ogcapi-processes
- http://www.opengis.net/def/interface/ogcapi-environmental-data-retrieval

C.1.3. Additional Resource Type Vocabularies

- DCAT 2 type/genre:[1]
- DCAT 3 type/genre:[2]
- DMCI Type Vocabulary[3]

[1] https://www.w3.org/TR/vocab-dcat-2/#Property:resource_type

[2] https://www.w3.org/TR/vocab-dcat-3/#Property:resource_type

[3] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#section-7

D

ANNEX D (INFORMATIVE) COMMON CONTACT ROLES (INFORMATIVE)

ANNEX D (INFORMATIVE) COMMON CONTACT ROLES (INFORMATIVE)

D.1. Overview

This annex describes common contact types when qualifying contact roles via a record contact's `roles` property, in the absence of a formal controlled vocabulary. Please note that these codelists and identifiers are NOT normative. For information communities wishing to formally identify and qualify, it is suggested that a specification extension be developed to meet those requirements.

- ISO 19115 contact roles (`CI_RoleCode`)[4]
- ISO 19115-3 contact roles (`CI_RoleCode`)[5]
- STAC i.e., Provider roles[6]
- DCAT 2 https://www.w3.org/TR/vocab-dcat-2/#Class:Role
- DCAT 3 https://www.w3.org/TR/vocab-dcat-3/#Class:Role

[4]https://www.isotc211.org/2005/resources/Codelist/gmxCodelists.xml#CI_RoleCode

[5]https://schemas.isotc211.org/schemas/19115/resources/Codelist/cat/codelists.xml#CI_RoleCode

[6]https://github.com/radiantearth/stac-spec/blob/master/collection-spec/collection-spec.md#provider-object

E

ANNEX E (INFORMATIVE) CROSSWALK BETWEEN STAC AND OGC API – RECORDS – PART 1 CORE (INFORMATIVE)

ANNEX E (INFORMATIVE) CROSSWALK BETWEEN STAC AND OGC API — RECORDS — PART 1 CORE (INFORMATIVE)

The document "Crosswalk between STAC and OGC API — Records" presents a comparison between this standard and the Spatial Temporal Asset Catalog, V 1.0.0.

F

ANNEX F (INFORMATIVE) ABBREVIATED TERMS

ANNEX F (INFORMATIVE) ABBREVIATED TERMS

API	Application Programming Interface
CORS	Cross-Origin Resource Sharing
CRS	Coordinate Reference System
CSW	Catalog Service for the Web
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
OGC	Open Geospatial Consortium
RFC	Request for Comment
TRS	Temporal Coordinate Reference System
URI	Uniform Resource Identifier
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WPS	Web Processing Service
YAML	YAML Ain't Markup Language or Yet Another Markup Language

G

ANNEX G (INFORMATIVE) REVISION HISTORY

ANNEX G (INFORMATIVE) REVISION HISTORY

Table G.1

DATE	RELEASE	CONTRIBUTORS	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2024-12-19	1.0.0	P. Vretanos, T. Kralidis, A. Tzotsos	all	Review for 1.0.0.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Internet Assigned Numbers Authority (IANA). Link Relation Types [online, viewed 2024-12-09], https://www.iana.org/assignments/link-relations/link-relations.xml

[2] W3C: Data Catalog Vocabulary — Version 3, W3C Recommendation 22 August 2024, https://www.w3.org/TR/vocab-dcat/

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

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

[5] Linux Foundation: SPDX License List, https://spdx.org/licenses/

[6] M. Pilgrim, M., Ringnalda, P.: Atom Feed Autodiscovery [online, viewed 2024-12-09], https://datatracker.ietf.org/doc/html/draft-ietf-atompub-autodiscovery

[7] RSS Advisory Board: RSS Autodiscovery [online, viewed 2024-12-09], https://www.rssboard.org/rss-autodiscovery

[8] W3C: Data Catalog Vocabulary — Version 2, W3C Recommendation 4 February 2020, https://www.w3.org/TR/vocab-dcat-2/