DRAFT OGC API - Records - Part 1

Core

Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: 2020-01-13

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

Internal reference number of this OGC® document: 20-004

Version: 0.0.1

Category: OGC® Implementation Specification

Editors: Panagiotis (Peter) A. Vretanos, Tom Kralidis, Charles Heazel

DRAFT OGC API - Records - Part 1: Core

Warning

The OGC API - Records specification is still a work in progress. Please be aware that the specification document that is in this repository is not yet in a state that could be useful to anyone who has not been attending the bi-weekly standards working group (SWG) meetings and/or who is not actively working on the document. The SWG is working diligently to update the specification so please refer back to this readme periodically for any status updates.

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

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

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

License Agreement

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

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

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

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

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

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

Table of Contents

i. Abstract

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

This document defines the OGC API for Records. A Record is a way of making a resource discoverable. In this context, resources are things that would be useful to a user or developer, such as features, coverages, tiles / maps, assets, services or widgets. The OGC API - Records provides a way to query or browse a curated set of Records known as a catalogue.

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	Declaration of Conformance Classes
Record collections	`/collections`	GET	Collection Access
Record collection	`/collections/{catalogueId}`	GET	Collection Information
Records	`/collections/{catalogueId}/items`	GET	Records Access
Record	`/collections/{catalogueId}/items/{recordId}`	GET	Record Access

ii. Keywords

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

ogcdoc, OGC document, OGC API, catalogue, record, resource

iii. Preface

NOTE

Insert Preface Text here. Give OGC specific commentary: describe the technical content, reason for document, history of the document and precursors, and plans for future work. > 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.

iv. Submitting organizations

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

Organization name(s)

- CubeWerx Inc.

v. 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

Chapter 1. Scope

This document specifies the behaviour of an API that supports the ability to search collections of descriptive information, called records, about resources such as data collections, services, processes, styles, code lists and other related resources. Records represent resource characteristics that can be queried and presented for evaluation and further processing by both humans and software.

This specification defines a list of core queryables and a set of core query parameter organized into X conformance classes.

Chapter 2. Conformance

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

The one Standardization Target for this standard is Web APIs.

OGC API - Common provides a common foundation for OGC API standards. Therefore, this standard should be viewed as an extension to API - Common. Conformance to this standard requires demonstrated conformance to the applicable Conformance Classes of API - Common.

This standard identifies six (6) Conformance Classes. The Conformance Classes implemented by 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 Class.

The Requirements Classes for OGC API - Records are:

- **Core**
- **Sorting**
- **OpenSearch**
- **JSON-Record**
- **ATOM-Record**
- **HTML-Record**

The *Core* Requirements Class is the minimal useful service interface for the OGC Records API. The requirements specified in this Requirements Class are mandatory for all implementations of API - Records. The core defines a set of mandatory properties (core queryables) that every record must contain, a set of core query parameters that support bounding box spatial queries, temporal queries, full text searching, parameters for sorting responses.

The *Sorting* Requirements Class defined the requirements to support sorting of records in a query response.

The *OpenSearch* Requirements Class defines the requirements to support query by OpenSearch clients.

The *JSON* Requirements Class defines the requirements for a GeoJSON (see RFC 7946) representation of a standard catalogue record.

The *ATOM* Requirements Class defines the requirements for an ATOM representation of a standard catalogue record.

The *HTML* Requirements Class defines the requirements for an HTML representation of a standard catalogue record.

Chapter 3. References

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

- Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: IETF RFC 2616, **HTTP/1.1**, RFC 2616
- Rescorla, E.: IETF RFC 2818, **HTTP Over TLS**, RFC 2818
- Klyne, G., Newman, C.: IETF RFC 3339, **Date and Time on the Internet: Timestamps**, RFC 3339
- Berners-Lee, T., Fielding, R., Masinter, L.: IETF RFC 3986, **Uniform Resource Identifier (URI): Generic Syntax**, RFC 3986
- Duerst, M., Suignard, M.: IETF RFC 3987, **Internationalized Resource Identifiers (IRIs)**, RFC 3987
- Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., Orchard, D.: IETF RFC 6570, **URI Template**, RFC 6570
- IETF RFC 7946: **The GeoJSON Format**, eoJSON
- Nottingham, M.: IETF RFC 8288, **Web Linking**, RFC 8288
- OGC 19-072: **OGC API (OAPI) Common Specification**, (Draft) API Common
- Open API Initiative: **OpenAPI Specification 3.0.2**, OpenAPI
- **Schema.org**: Schema.org
- W3C: **HTML5**, W3C Recommendation, HTML5
- W3C, **RDF 1.1 Semantics**, February 2014, https://www.w3.org/TR/rdf11-mt/
- W3C, **Extensible Markup Language (XML) 1.0 (Fifth Edition)**, November 2008, https://www.w3.org/TR/xml/
- OGC: OGC 07-036, Geography Markup Language (GML) Encoding Standard, version 3.2.1, 2007
- OGC: OGC 10-129r1, OGC® Geography Markup Language (GML) – Extended schemas and encoding rules (GML 3.3), version 3.3, 2012
- W3C: W3C Recommendation, XML Path Language (XPath), version 2, 2007
- W3C: W3C Recommendation, XML Linking Language (XLink), version 1, 2001
- W3C: W3C Working Draft, The app: URI scheme, 2013
- ISO/IEC: ISO/IEC 19757-3:2006 Information technology – Document Schema Definition Languages (DSDL) – Part 3: Rule-based validation – Schematron, 2006
- IETF: RFC 2183, 1997
- IETF: RFC 2387, 1998
- IETF: RFC 2392, 1998 [18] IETF: RFC 3986, 2005 [19] IETF: RFC 7159, The JavaScript Object Notation (JSON) Data Interchange Format https://www.ietf.org/rfc/rfc7159.txt, 2014

- W3C: W3C JSON-LD 1.0, A JSON-based Serialization for Linked Data. http://www.w3.org/TR/json-ld/, 2014
- W3C: W3C JSON-LD 1.0 Processing Algorithms and API. http://www.w3.org/TR/json-ld-api, 2014
- W3C: W3C RDF 1.1 Concepts and Abstract Syntax. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/, 2014

Chapter 4. Terms and Definitions

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

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

4.1. Conformance Module; Conformance Test Module

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

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

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

4.3. dataset

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

4.4. Distribution

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

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

4.5. Executable Test Suite (ETS)

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

4.6. 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)

4.7. 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)

4.8. Requirements Class

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

4.9. Requirements Module

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

4.10. Standardization Target

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

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

Chapter 5. Conventions

The following conventions will be used in this document. Examples of conventions are symbols, abbreviations, use of XML 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. 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. Implementors are free to use any format they desire as long as there is a Conformance Class for that format and the API advertises its support for that Conformance Class.

5.3. Schema

JSON Schema is 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 JSON schema is required. Implementations should use a schema language and encoding appropriate for the format of the resource.

5.4. UML Notation

Diagrams using the Unified Modeling Language (UML) adhere to the following conventions:

- UML elements having a package name of “GML” are those defined in the UML model of GML 3.2.1
- UML elements having a package name of “SWE Common” are those defined in the UML model of SWE Common 2.0
- UML elements not qualified with a package name, or with “CIS”, are those defined in this standard.

Further, in any class where an attribute name or association role name is identical to a name in some superclass the local definition overrides the superclass definition.

5.5. Namespace Prefix Conventions

UML diagrams and XML code fragments adhere to the namespace conventions shown in [namespace-mapping-conventions]. The namespace prefixes used in this document are not normative and are merely chosen for convenience. The namespaces to which the prefixes correspond are normative, however.

Table 2. Namespace mapping conventions

UML prefix	GML prefix	Namespace URL	Description
GML	gml	http://www.opengis.net/gml/3.2	GML 3.2.1
GML33	gml33	http://www.opengis.net/gml/3.3	GML 3.3

Chapter 6. Overview

6.1. General

The OGC API family of standards enable access to resources using the HTTP protocol and its' associated operations (GET, PUT, POST, etc.). OGC API - Common defines a set of features which are applicable to all OGC APIs. Other OGC standards extend API - Common with features specific to a resource type. This OGC API - Records standard defines an API with two goals:

1. Provide modern API patterns and encodings to facilitiate further lowering the barrier to finding the existence of spatial resources on the Web.
2. Provide functionality comparable to that of the OGC Catalogue Service (CSW) standard.

Resources exposed through an OGC API may be accessed through a Universal Resource Identifier (URI). URIs are composed of three sections:

- Service Offering: The service endpoint (subsequently referred to as Base URI or {root})
- Access Paths: Unique paths to Resources
- Query: Parameters to adjust the representation of a Resource or Resources like encoding format or subsetting

Some resources are also accessible through links on previously accessed resources. Unique relation types are used for each resource.

The following table, Record API Paths, summarizes the access paths and relation types defined in this standard.

Table 3. Record API Paths

Path Template	Relation	Resource
Common		
{root}/	none	Landing page
{root}/api	`service-desc` or `service-doc`	API Description (optional)
{root}/conformance	`conformance`	Conformance Classes
{root}/collections	`data`	Metadata describing the spatial collections available from this API.
{root}/collections/{collectionid}		Metadata describing the collection which has the unique identifier `{collectionid}`
Records		

Path Template	Relation	Resource
{root}/collections/{collectionid}/items	`items`	Search results based on querying the service for records satisfying 0..n query parameters.
{root}/collections/{collectionid}/items/{recordid}	`item`	Record of metadata which has the unique identifier `{recordid}`.

Where:

- {root} = Base URI for the API server
- {collectionid} = an identifier for a specific collection
- {recordid} = an identifier for a specific record within a collection

6.2. Record Schema

6.3. API Behavior Model

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

NOTE Replace the following with a discussion of CSW and API - Record

OGC Catalogue Service standard version 3 provides an abstract core model of metadata (data about data) describing a number of different information types (data, services, styles, processes, etc.) on which the classic operations GetCapabilities, DescribeRecord, GetRecords, and GetRecordById can be explained naturally. This model consists of a 1..n catalogue collections residing in a CSW backend repository. It holds service metadata describing service qualities (identification, contact, operations, filtering capabilities, etc.). At its heart, a catalogue may provide discovery services to any number of metadata repositories. The core catalogue 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 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.

6.4. Search

The Records API offers various levels of search capability of escalating complexity and capability. At the core is the ability to search the catalogue by specifying:

- a bounding box
- a time instant or time period

- keywords
- equality predicates based on a subset of core queryables (e.g. by resource type, by external identifier)

This specification also includes a conformance class that allows a catalogue to be searched using 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.

Finally extensions, such as the OGC API - Features - Part 3: Filter and the Common Query Language (CQL), may be used with the Records API to support complex search capabilities using a rich set of logically connected search predicates where the user has full control over to query expression.

6.5. Dependencies

The OGC API - Records standard is an extension of the OGC API - Common standard. Therefore, an implementation of OGC API - Records must first satisfy the appropriate Requirements Classes from API - Common. The following table, Mapping API - Records Sections to API - Common Requirements Classes, identifies the OGC API - Common Requirements Classes which are applicable to each section of this Standard. Instructions on when and how to apply these Requirements Classes are provided in each section.

Table 4. Mapping API - Records Sections to API - Common Requirements Classes

API - Record Section	API - Common Requirements Class
API Landing Page	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core
API Definition	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core
Declaration of Conformance Classes	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core
Collections	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/collections
OpenAPI 3.0	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/oas30
JSON	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/geojson
HTML	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/html

Chapter 7. Requirements Class "Core"

7.1. Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-records-1/1.0/req/core	
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core

The `Core` Requirements Class defines the requirements for locating, understanding, and accessing Record resources. The `Core` Requirements Class is presented in five sections:

1. API Platform: a set of common capabilities
2. Collection Access: operations for accessing information about collections of records
3. Records Access: operations for accessing record resources
4. Record Access: operations for accessing a single record
5. General: general principles for use with this standard

7.2. Dependencies

The OGC API - Records standard is an extension of the OGC API - Common standard. Therefore, an implementation of API - Records must first satisfy the appropriate Requirements Classes from API - Common.

Requirement 1	**/req/core/api-common**
The API implementation SHALL demonstrate conformance with the following Requirements Classes of the OGC API - Common version 1.0 Standard.	
A	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core
B	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/collections

7.3. Platform

API - Common defines a set of common capabilities which are applicable to any OGC Web API. Those capabilities provide the platform upon which resource-specific APIs can be built. This section describes those capabilities and any modifications needed to better support Record resources.

7.3.1. API landing page

The landing page provides links to start exploration of the resources offered by an API. Its most important component is a list of links. OGC API - Common already requires some common links.

Those links are sufficient for this standard.

Table 5. Dependencies

http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core

7.3.1.1. Operation

The `Landing Page` operation is defined in the `Core` conformance class of API - Common. No modifications are needed to support `Record` resources. The `Core` conformance class specifies only one way of performing this operation:

1. Issue a `GET` request on the `{root}/` path

Support for `GET` on the `{root}/` path is required by API - Common.

7.3.1.2. Response

A successful response to the `Landing Page` operation is defined in API - Common. The schema for this resource is provided in Landing Page Response Schema.

Landing Page Response Schema

```
type: object
required:
  - links
properties:
  title:
    description: The title of the API
    type: string
  description:
    description: A textual description of the API
    type: string
  links:
    description: Links to the resources exposed through this API.
    type: array
    items:
      $ref: link.yaml
```

The following JSON fragment is an example of a response to an OGC API - Records Landing Page operation.

Landing Page Example


```
{
  "links": [
    { "href": "http://data.example.org/",
      "rel": "self",
      "type": "application/json",
      "title": "This Document" },
    { "href": "http://data.example.org/api",
      "rel": "service-desc",
      "type": "application/openapi+json;version=3.0",
      "title": "The API Definition" },
    { "href": "http://data.example.org/api",
      "rel": "service-doc",
      "type": "text/html",
      "title": "The API Document" },
    { "href": "http://data.example.org/conformance",
      "rel": "conformance",
      "type": "application/json",
      "title": "OGC Conformance Classes Implemented by this API" },
    { "href": "http://data.example.org/collections",
      "rel": "data",
      "type": "application/json",
      "title": "Metadata About the Resource Collections" }
  ]
}
```


7.3.1.3. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP status codes.

7.3.2. API definition

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

Table 6. Dependencies

http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core

7.3.2.1. Operation

This operation is defined in the `Core` conformance class of API - Common. No modifications are needed to support `Records` resources. The `Core` conformance class describes two ways of performing this operation:

1. Issue a `GET` request on the `{root}/api` path
2. Follow the `service-desc` or `service-doc` link on the landing page

Only the link is required by API - Common.

7.3.2.2. Response

A successful response to the API Definition request is a resource which documents the design of the API. API - Common leaves the selection of format for the API Definition response to the API implementor. However, the options are limited to those which have been defined in the API - Common standard. At this time OpenAPI 3.0 is the only option provided.

7.3.2.3. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP status codes.

7.3.3. Declaration of conformance classes

To support "generic" clients that want to access multiple OGC API standards and extensions - and not "just" a specific API / server, the API has to declare the conformance classes it claims to have implemented.

Table 7. Dependencies

http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core

7.3.3.1. Operation

This operation is defined in the `Core` conformance class of API - Common. No modifications are needed to support `Records` resources. The `Core` conformance class describes two ways of performing this operation:

1. Issue a `GET` request on the `{root}/conformance` path
2. Follow the `conformance` link on the landing page

Both techniques are required by API - Common.

7.3.3.2. Response

A successful response to the Conformance operation is a list of URLs. Each URL identifies an OGC Conformance Class for which this API claims conformance. The schema for this resource is defined in API - Common and provided for reference in Conformance Response Schema.

Requirement 2	**/req/core/conformance**
The list of Conformance Classes advertised by the API SHALL include:	
A	http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/core
B	http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/collections

C	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/core

Permission 1	**/per/core/additional-conformance**
The list of Conformance Classes advertised by the API may additionally include:	
A	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/opensearch
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/atom
D	http://www.opengis.net/spec/ogcapi-records-1/1.0/req/html
E	http://www.opengis.net/spec/ogcapi-records/1.0/req/oas30

Conformance Response Schema

```
type: object
required:
  - conformsTo
properties:
  conformsTo:
    type: array
    items:
      type: string
      example: "http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/core"
```

The following JSON fragment is an example of a response to an OGC API - Records conformance operation.

Conformance Information Example

```
{
  "conformsTo": [
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/core",
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/collections",
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/oas3",
    "http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/core",
    "http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/search",
    "http://www.opengis.net/spec/ogcapi-records-1/1.0/req/opensearch",
    "http://www.opengis.net/spec/ogcapi-records-1/1.0/req/json",
    "http://www.opengis.net/spec/ogcapi-records-1/1.0/req/atom",
    "http://www.opengis.net/spec/ogcapi-records-1/1.0/req/html"
  ]
}
```

7.3.3.3. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP status codes.

7.4. Collection Access

API - Common starts with the assumption that spatial resources are organized into collections. An API will expose one or more collections. The API - Common Collections Conformance Class defines how to organize and provide access to a collection of collections.

This standard extends the API - Common `Collections` conformance class to support collections of records, then extends that class to support `Record` unique capabilities.

7.4.1. Record Collections

The `Collections` operation returns a set of metadata which describes the collections available from this API.

Table 8. Dependencies

http://www.opengis.net/spec/ogcapi_common-1/1.0/req/collections

7.4.1.1. Operation

This operation is defined in the `Collections` conformance class of API - Common. No modifications are needed to support `Record` resources. The `Collections` conformance class describes two ways of performing this operation:

1. Issue a `GET` request on `{root}/collections` path
2. Follow the `data` link on the landing page

Support for both the `{root}/collections` path and the `data` link is required by API - Common.

7.4.1.2. Response

A successful response to the `Collections Operation` is a document which includes summary metadata for each collection accessible though the API.

In a typical deployment the collections response will list collections of all offered resource types. The collections labeled with an `itemType` of `record` are catalogues (i.e. collections of records).

Collections Response Schema

```
type: object
required:
  - links
  - collections
properties:
  links:
    type: array
    items:
      $ref: link.yaml
  collections:
    type: array
    items:
      $ref: collectionInfo.yaml
```

The following JSON fragment is an example of a response to an OGC API - Records Collections operation.

Collections Example

```
{
  "links": [
    {
      "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/ogcapi/catalogues/collections?f=app
lication%2Fjson",
      "rel": "self",
      "type": "application/json",
      "title": "this document"
    },
    {
      "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/ogcapi/catalogues/collections?f=tex
t%2Fxml",
      "rel": "alternate",
      "type": "text/xml",
      "title": "this document as XML"
    },
    {
      "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/ogcapi/catalogues/collections?f=tex
t%2Fhtml",
      "rel": "alternate",
      "type": "text/html",
      "title": "this document as HTML"
    }
  ],
  "collections": [
    .
```

```
    .
    .
    {
      "id": "radarsat2cat",
      "itemType": "record",
      "title": "CubeWerx RADARSAT-2 Catalogue",
      "description": "A sample catalogue of RADARSAT-2 products stored in S3 on AWS.",
      "links": [
        {
          "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/ogcapi/catalogues/collections/s1tepcat",
          "rel": "collection",
          "title": "Root URL for this record collection.  At this endpoint you can
retrieve a description of this catalogue as well as hypermedia controls that allow you
to query the catalogue."
        }
      ]
    },
    .
    .
    .
    {
      "id": "sentinel1cat",
      "itemType": "record",
      "title": "CubeWerx Sentinel-1 Catalogue",
      "description": "A sample catalogue of Sentinel-1 products stored in S3 on AWS.",
      "links": [
        {
          "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/ogcapi/catalogues/collections/sentinel1cat",
          "rel": "collection",
          "title": "Root URL for this record collection.  At this endpoint you can
retrieve a description of this catalogue as well as hypermedia controls that allow you
to query the catalogue."
        }
      ]
    },
    .
    .
    .
  ]
}
```


7.4.1.3. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP status codes.

7.4.2. Collection Information

Collection Information is the set of metadata which describes a single collection. An abbreviated copy of this information is returned for each Collection in the `/collections` response.

Table 9. Dependencies

http://www.opengis.net/spec/ogcapi_common-1/1.0/req/collections

7.4.2.1. Operation

This operation is defined in the `Collections` conformance class of API - Common. No modifications are required to support `Records` resources.

1. Issue a `GET` request on the `{root}/collections/{collectionid}` path

The {collectionid} parameter is the unique identifier for a single collection on the API. The list of valid values for {collectionid} is provided in the /collections response.

Support for the `/collections/{collectionid}` path is required by API - Common.

7.4.2.2. Response

A successful response to the Collection Operation is a set of metadata which describes the collection identified by the {collectionid} parameter.

Collection Information Response Schema

```
type: object
required:
  - id
  - links
properties:
  id:
    type: string
    example: address
  title:
    type: string
    example: address
  description:
    type: string
    example: An address.
  links:
    type: array
    items:
      $ref: link.yaml
    example:
      - href: http://data.example.com/buildings
        rel: item
      - href: http://example.com/concepts/buildings.html
        rel: describedBy
        type: text/html
  extent:
    $ref: extent.yaml
  itemType:
    description: indicator about the type of the items in the collection (the default
value is 'unknown').
    type: string
    default: unknown
  crs:
    description: the list of coordinate reference systems supported by the API; the
first item is the default coordinate reference system
    type: array
    items:
      type: string
    default:
      - http://www.opengis.net/def/crs/OGC/1.3/CRS84
    example:
      - http://www.opengis.net/def/crs/OGC/1.3/CRS84
      - http://www.opengis.net/def/crs/EPSG/0/4326
```

The following JSON fragment is an example of a response to an OGC API - Records Collection Information operation.

Collection Information Example


```
{
   "id": "sentinel1cat",
   "title": "Sentinel-1 Catalogue",
   "description": "A sample catalogue of Sentinel-1 products stored in S3 on AWS.",
   "links": [
      {
         "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/ogcapi/catalogues/collections/senti
nel1cat/items",
         "rel": "items",
         "title": "Catalogue records describing SENTINEL-1 products."
      }
   ]
}
```


7.4.2.3. Error situations

The requirements for handling unsuccessful requests are provided in HTTP Response. General guidance on HTTP status codes and how they should be handled is provided in HTTP status codes.

7.5. Records Access

7.5.1. Operation

Requirement 3	/req/core/rc-op
A	A Records API implementation SHALL comply with the API - Common requirement http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-op.
B	For API - Records, the API - Common /req/collections/rc-op requirement SHALL only apply to collections where the itemType is specified as records.

7.5.2. Parameters

7.5.2.1. Overview

The following tables list the set mandatory query parameters.

Table 10. Table of 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 data/time value then the record shall be presented in the response document.
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 records shall appear in the response set.
type	An equality predicate consistent of a comma-separated list of resource types. Only records of the listed type shall appear in the resource set.
externalid	An equality predicate consistent of a comma-separated list of external resource identifiers. Only records with the specified external identifiers shall appear in the response set.

The query parameters `bbox`, `datetime` and `limit` are inherited from API -Common. All requirements and recommendations in API - Common regarding these parameters also apply for API - Records.

The query parameters `q`, `type` and `externalid` are specific to API - Records. This clause provides a brief description of the query parameters inherited from API - Common and requirements for the API - Records-specific parameters.

7.5.2.2. Parameter bbox

The Bounding Box (bbox) parameter is defined in API - Common. The following requirement governs use of that parameter in a Records API.

Requirement 4	/req/core/param-bbox
A	A Records API SHALL support the Bounding Box (bbox) parameter for the operation.
B	Requests which include the Bounding Box parameter SHALL comply with API - Common requirement `http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-bbox-definition`.
C	Responses to Bounding Box requests SHALL comply with API - Common requirement `http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-bbox-response`.

7.5.2.3. Parameter datetime

The Date-Time (datetime) parameter is defined in API - Common. The following requirement

governs use of that parameter in a Records API.

Requirement 5	**/req/core/param-datetime**
A	A Records API SHALL support the Date-Time (datetime) parameter for the operation.
B	Requests which include the Date-Time parameter SHALL comply with API - Common requirement http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-time-definition.
C	Responses to Date-Time requests SHALL comply with API - Common requirement http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-time-response.

7.5.2.4. Parameter limit

The limit parameter is defined in API - Common. The following requirement governs use of that parameter in a Records API.

Requirement 6	**/req/core/param-limit**
A	A Records API SHALL support the Limit (limit) parameter for the operation.
B	Requests which include the Limit parameter SHALL comply with API - Common requirement http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-limit-definition.
C	Responses to Limit requests SHALL comply with API - Common requirements: • http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-limit-response • http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-numberReturned • http://www.opengis.net/spec/ogcapi_common-2/1.0/req/collections/rc-numberMatched

7.5.2.5. Parameter q

Requirement 7	**/req/core/param-q-definition**
A	A Records API SHALL support the Keyword 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`
	`required: false`
	`schema:`
	`  type: array`
	`  items:`
	`  type: string`
	`explode: false`

B	Keyword searches using the `q` parameter shall be case insensitive.

Requirement 8	**/req/core/param-q-response**
A	If the `q` parameter is provided by the client, only records whose text fields contains one or more of the specified search terms SHALL be in the result set.
B	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.

7.5.2.6. Parameter type

Requirement 9	**/req/core/param-type-definition**
A	A Records API SHALL support the search by Type (type) parameter for the operation.

A	The `type` parameter SHALL have the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: type`
	`in: query`
	`required: false`
	`schema:`
	`  type: array`
	`  items:`
	`  type: string`
	`explode: false`

Requirement 10	/req/core/param-type-response
A	If the `type` parameter is provided by the client, only records whose type, as indicated by the value of the `type` core queryable, is equal to one of the listed values SHALL be in the result set.

7.5.2.7. Parameter externalid

Requirement 11	/req/core/param-externalid-definition
A	A Records API SHALL support the search by External Identifier (externalid) parameter for the operation.

B	The `externalid` parameter SHALL have the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: externalid`
	`in: query`
	`required: false`
	`schema:`
	`  type: array`
	`  items:`
	`  type: string`
	`explode: false`

Requirement 12	/req/core/param-type-response
A	If the `externalid` parameter is provided by the client, only records whose external identifier, as indicated by the value of the `externalid` core queryable, is equal to one of the listed values SHALL be in the result set.

NOTE	I have chosen a small subset of core queryables (type, externalId) and mapped them as URL query parameters. We could generalize this and say that any core queryable can be included as a URL query parameter. Should we do this instead?

7.5.2.8. Combinations of Filtering Parameters

Any combination of query parameters from the Table of Query Parameters may be specified for filtering.

Any combination of `bbox`, `datetime`, `q` and `limit` parameters for filtering on Record properties is allowed. Note that the requirements on these parameters imply that only records matching all the predicates are in the result set; i.e., the logical operator between the predicates is 'AND' (exclusive).

7.5.3. Response

Requirement 13	/req/core/rc-response
A	Each record in the response SHALL contain the mandatory properties listed in the Table of Core Queryables related to the catalogue record and the Table of Core Queryables related to the resource.

Permission 2	/per/core/additional-properties
A	Each record in the response MAY contain zero or more of the optional properties listed in the Table of Core Queryables related to the catalogue record and the Table of Core Queryables related to the resource.
B	Each record in the response MAY contain any number of additional domain-specific properties not listed in the Table of Core Queryables related to the catalogue record and the Table of Core Queryables related to the resource. The meaning of these additional properties is not specified in this document.

Table 11. Table of Core Queryables related to the catalogue record

Queryables	Requirement	Description	GeoJSON key
recordId	M	A unique record identifier assigned by the server.	id
record-created	O	The date this record was created in the server.	created
record-updated	O	The most recent date on which the record was changed.	updated
links	O	A list of links for navigating the API (e.g. link to previous or next pages; links to alterntive representations, etc.)	links

Table 12. Table of Core Queryables related to the resource

Queryables	Requirement	Description	GeoJSON key
type	M	The nature or genre of the resource.	properties.type

Queryables	Requirement	Description	GeoJSON key
title	M	A human-readable name given to the resource.	properties.title
description	O	A free-text description of the resource.	properties.description
keywords	O	A list of keywords or tag associated with the resource.	properties.keyword
language	O	This refers to the natural language used for textual values (i.e. titles, descriptions, etc) of a resource.	properties.language
externalid	O	An identifier for the resource assigned by an external entity.	properties.externalid
created	O	The date the resource was created.	properties.created
updated	O	The more recent date on which the resource was changed.	properties.updated
publisher	O	The entity making the resource available.	properties.publisher
themes	O	A knowledge orgnaization system used to classify the resource.	properties.themes
formats	O	A list of available distributions for the resource.	properties.formats
contactpoint	O	An entity to contact about the resource.	properties.contactpoint
license	O	A legal document under which the resource is made available.	properties.license
rights	O	A statement that concerns all rights not addressed by the license such as a copyright statement.	properties.rights
extent	O	The spatio-temporal coverage of the resource.	properties.extent
associations	O	A list of links for accessing the resource, links to other resources associated with this resource, etc.	properties.associations

7.5.4. Paged Response

One consequence of the Limit parameter is that the full result set is not delivered to the user. However, users frequently want to know how big the result set it and how to access the rest of it. The following requirement add information to the response to address that need.

Requirement 14	/req/core/rc-paged-response

A	Responses to a filtered operation that only return a portion of the full selected resource set SHALL comply with API - Common requirements: • `/req/core/rc-response` • `/req/core/rc-links` • `/req/core/rc-rel-type` • `/req/core/rc-timestamp` • `/req/core/rc-numberMatched` • `/req/core/rc-numberReturned`

7.6. Record Access

7.6.1. Operation

Requirement 15	**/req/core/record-op**
A	For every record in a record collection (path `/collections/{catalogueId}`), the server SHALL support the HTTP GET operation at the path `/collections/{catalogueId}/items/{recordId}`.
B	The parameter `cataloagueId` is each `id` property in the collections response (JSONPath: `$.collections[*].id`) where the item type is specified as `record`. `recordId` is a local identifier of the record.

7.6.2. Response

Requirement 16	**/req/core/record-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.

7.7. General

7.7.1. HTTP Response

Each HTTP request shall result in a response that meets the following requirement.

Requirement 17	**/req/core/http-response**

A	An HTTP operation SHALL return a response which includes a `status code` and an optional `description` elements.
B	If the `status code` is not equal to 200, then the `description` element SHALL be populated.

The YAML schema for these results is provided in HTTP Response Schema.

HTTP Response Schema

```
type: object
required:
  - code
properties:
  code:
    type: string
  description:
    type: string
```

7.7.2. HTTP status codes

The **Status Codes** listed in [status_codes] are of particular relevance to implementors of this standard. Status codes 200, 400, and 404 are called out in API requirements. Therefore, support for these status codes is mandatory for all compliant implementations. The remainder of the status codes in [status_codes] are not mandatory, but are important for the implementation of a well functioning API. Support for these status codes is strongly encouraged for both client and server implementations.

Table 13. Typical HTTP status codes

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

Status code	Description
405	The request method is not supported. For example, a POST request was submitted, but the resource only supports GET requests.
406	The `Accept` header submitted in the request did not support any of the media types supported by the server for the requested resource.
500	An internal error occurred in the server.

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

The API Description Document describes the HTTP status codes generated by that API. This should not be an exhaustive list of all possible status codes. It is not reasonable to expect an API designer to control the use of HTTP status codes which are not generated by their software. Therefore, it is recommended that the API Description Document limit itself to describing HTTP status codes relevant to the proper operation of the API application logic. Client implementations should be prepared to receive HTTP status codes in addition to those described in the API Description Document.

Permission 3	/per/core/additional-status-codes
A	Servers MAY support other capabilities of the HTTP protocol and, therefore, MAY return other status codes than those listed in [status_codes], too.

Chapter 8. Requirements Class "Sorting"

8.1. Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-records-1/1.0/req/sorting	
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/sorting

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

8.2. Parameter sortby

Requirement 18	/req/sorting/sortby-definition

A	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_][A-Za-z0-9]*`
	`style: form`
	`explode: false`

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

NOTE 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: handling missing values; specifying a high value indicating that the corresponding property be sorted as if it were the highest possible value; specifying a low value indicating that the corresponding property be sorted as if it were the lowest possible value; allowing records to be omitted from the result set based on their sort order; specify a fixed value and a fixed value that sorts the corresponding property as if it were the specified fixed value.

Requirement 19	/req/core/rc-sortby-response
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 used for sorting SHALL be specified by the `/collections/{catalogueId}/sortables` resource.

Requirement 20	/req/sorting/sortables-op
A	For every collection (path `/collections/{catalogueId}`), the server SHALL support the HTTP GET operation at the path `/collections/{catalogueId}/sortables`.
B	The parameter `catalogue` is each `id` property in the collections response (JSONPath: `$.collections[*].id`) where the item type is specified as `record`.

Requirement 21	/req/core/sortables-success
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The content of that response SHALL contain a document that validates against the OpenAPI 3.0 schema document `sortables.yaml`.

Schema for the list of sortables (sortables.yaml)

```
type: object
required:
- sortables
properties:
   sortables:
      type: array
      items:
         $ref: sortable.yaml
```

Schema for a sortable description(sortable.yaml)

```
type: object
required:
   - id
   - type
```

```
properties:
   id:
      description: the identifier/name for the sortable
      type: string
   title:
      description: a human readable title for the sortable
      type: string
   description:
      description: a human-readable narrative describing the sortable
      type: string
   language:
      description: the language used for the title and description
      type: string
   type:
      description: the data type of the sortable
      type: string
      enum:
      - string
      - number
      - integer
      - boolean
      - spatial
      - temporal
   spatial-types:
      description: the list of allowed spatial geometry types, if known
      type: string
      enum:
      - point
      - multi-point
      - curve
      - multi-curve
      - surface
      - multi-surface
      - solid
      - multi-solid
      - aggregate
      - any
   temporal-types:
      description: the list of allowed temporal geometry types, if known
      type: string
      enum:
      - instant
      - interval
   links:
      description: additional links, e.g. to documentation or to the schema of t
he values
      type: array
      items:
         $ref: 'https://raw.githubusercontent.com/opengeospatial/ogcapi-features
/master/core/openapi/ogcapi-features-1.yaml#/components/schemas/link'
```

Chapter 9. Examples

sortby Example

```
.../collections/mycat/items?...&sortby=-updated,recordid&...
```

sortby Extent Example

```
.../collections/mycat/items?...&sortby=+extent&...
```

Chapter 10. Requirements Class "OpenSearch"

10.1. Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-records-1/1.0/req/opensearch	
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/core
Dependency	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/collections

Chapter 11. Requirements Class JSON

The following requirements apply to an OGC API - Records implementation when the following conditions apply:

1. The API advertises conformance to the JSON Conformance Class
2. The client negotiates a JSON or GeoJSON format

The JSON Requirements Class restricts requirements defined in the `Core` Requirements Class by imposing encoding-specific requirements. At this time, these additional requirements only apply to the HTTP response payloads. The sections below identify the scope of each new requirement and the `Core` requirements which lay within each scope.

Requirements Class	
http://www.opengis.net/spec/ogcapi-records-1/1.0/req/json	
Target type	Web API
Dependency	Requirements Class "API - Common Core"
Dependency	API - Common GeoJSON
Dependency	GeoJSON
Pre-conditions	1) The API advertises conformance to the JSON Conformance Class 2) The client negotiates use of the JSON or GeoJSON encoding.

11.1. Common

This section covers the requirements inherited from the API - Common standard. Its scope includes responses for the following operations:

- `{root}/`: Landing Page
- `{root}/api`: API Description
- `{root}/conformance`: Conformance Classes
- `{root}/collections`: Collections
- `{root}/collections/{collectionid}`: Collection Information

Requirement 22	**/req/json/api-common**
Extends	/req/core/api-common
The API SHALL demonstrate conformance with the following Requirements Class of the OGC API - Common version 1.0 Standard.	
A	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/geojson

It is also necessary to advertise conformance with this Requirements Class.

Requirement 23	**/req/json/conformance**
The list of Conformance Classes advertised by the API SHALL include:	
A	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/json

11.2. Record

TBD: Requirements for the GeoJSON encoding of a record.

Chapter 12. Requirements Class HTML

The following requirements apply to an OGC API - Records implementation when the following conditions apply:

1. The API advertises conformance to the HTML Conformance Class
2. The client negotiates an HTML format

The HTML Requirements Class restricts requirements defined in the `Core` Requirements Class by imposing encoding-specific requirements. At this time, these additional requirements only apply to the HTTP response payloads. The sections below identify the scope of each new requirement and the `Core` requirements which lay within each scope.

Requirements Class	
http://www.opengis.net/spec/ogcapi-records-1/1.0/req/html	
Target type	Web API
Dependency	Conformance Class "Core"
Dependency	API - Common HTML
Dependency	HTML5
Dependency	Schema.org

12.1. Common

This section covers the requirements inherited from the API - Common standard. Its scope includes responses for the following operations:

- `{root}/`: Landing Page
- `{root}/api`: API Description
- `{root}/conformance`: Conformance Classes
- `{root}/collections`: Collections
- `{root}/collections/{collectionid}`: Collection Information

Requirement 24	**/req/html/api-common**
Extends	/req/core/api-common
The API SHALL demonstrate conformance with the following Requirements Class of the OGC API - Common version 1.0 Standard.	
A	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/html

It is also necessary to advertise conformance with this Requirements Class.

Requirement 25	**/req/html/conformance**
The list of Conformance Classes advertised by the API SHALL include:	
A	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/html

12.2. Record

RBDL Requirements for the HTML encoding of a record.

Chapter 13. Requirements Class ATOM

The following requirements apply to an OGC API - Records implementation when the following conditions apply:

1. The API advertises conformance to the ATOM Conformance Class
2. The client negotiates a ATOM format

The ATOM Requirements Class restricts requirements defined in the `Core` Requirements Class by imposing encoding-specific requirements. At this time, these additional requirements only apply to the HTTP response payloads. The sections below identify the scope of each new requirement and the `Core` requirements which lay within each scope.

Requirements Class	
http://www.opengis.net/spec/ogcapi-records-1/1.0/req/atom	
Target type	Web API
Dependency	Requirements Class "API - Common Core"
Dependency	The Atom Publishing Protocol
Pre-conditions	1) The API advertises conformance to the ATOM Conformance Class 2) The client negotiates use of the ATOM encoding.

13.1. Common

This section covers the requirements inherited from the API - Common standard. Its scope includes responses for the following operations:

- `{root}/`: Landing Page
- `{root}/api`: API Description
- `{root}/conformance`: Conformance Classes
- `{root}/collections`: Collections
- `{root}/collections/{collectionid}`: Collection Information

Requirement 26	**/req/xml/api-common**
Extends	/req/core/api-common
The API SHALL demonstrate conformance with the following Requirements Class of the OGC API - Common version 1.0 Standard.	
A	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/xml

It is also necessary to advertise conformance with this Requirements Class.

Requirement 27	**/req/xml/conformance**
The list of Conformance Classes advertised by the API SHALL include:	
A	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/xml

13.2. Record

TBD: Requirements for the ATOM encoding of a record.

Chapter 14. Requirements class "OpenAPI 3.0"

Requirements Class	
http://www.opengis.net/spec/ogcapi-records/1.0/req/oas30	
Target type	Web API
Dependency	Conformance Class "Core"
Dependency	OGC API - Common Standard 1.0
Dependency	OpenAPI Specification 3.0.2

The OpenAPI 3.0 Requirements Class is applicable to API - Records as well. So an implementation of API - Records which supports OpenAPI 3.0 as an API Description format must also comply with the API - Common oas30 Conformance Class.

Requirement 28	**/req/oas30/oas-common**
Extends	/req/core/api-common
A	The API SHALL demonstrate conformance with the following Requirements Class of the OGC API - Common version 1.0 Standard. http://www.opengis.net/spec/ogcapi-common-1/1.0/req/oas30.

Implementations must also advertise conformance with this Requirements Class.

Requirement 29	**/req/oas30/conformance**
The list of Conformance Classes advertised by the API SHALL include:	
A	http://www.opengis.net/spec/ogcapi-records-1/1.0/conf/oas30

Chapter 15. Media Types

This standard does not mandate any particular encoding or format. However, it does provide extensions for encodings which are commonly used in OGC APIs. These extensions include:

- JSON
- HTML

Neither of these encodings are mandatory. An implementor of this standard may choose to implement neither of them, selecting different encodings instead.

15.1. HTML Encoding

Support for HTML is recommended. HTML is the core language of the World Wide Web. An API that supports HTML will support browsing the spatial resources with a web browser and will also enable search engines to crawl and index those resources.

15.2. JSON Encoding

Support for JSON is recommended. JSON is a commonly used format that is simple to understand and well supported by tools and software libraries.

JSON structures documented in this standard are defined using JSON Schema. These schema are available in JSON and YAML formats from http://schemas.opengis.net/tbd

15.2.1. GeoJSON

"GeoJSON is a geospatial data interchange format based on JavaScript Object Notation (JSON). It defines several types of JSON objects and the manner in which 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 represening OGC Featues in JSON. Due to its simplicity, however, it is not suitable for all feature data. It is best used for content which has a spatial extent that can be used with the World Geodetic System 1984 Coordinate Reference System.

15.3. Media Types

A description of the MIME-types is mandatory for any OGC standard which involves data encodings. The list of suitable MIME-types for the API - Records standard in provided in [api-records-mime-types].

Table 14. API - Records MIME Types

Encoding	MIME Type
HTML	text/html

JSON	application/json
GeoJSON	application/geo+json
ATOM	application/atom+xml

15.4. Default Encodings

The media type used to encode a response to a request shall be determined through the HTTP content negotiation protocol as specified in API - Common. However, content negotiation is not required by the HTTP standard. So default encodings must be established.

Requirement 30	**/req/core/rc-mediatype-default**
A	IF the `JSON` Conformance Class is advertised, then the default media type for content SHALL be JSON or GeoJSON.
C	IF the `JSON` Conformance Class is not advertised, then the default media type for content SHALL be HTML.

Chapter 16. Security Considerations

TBD: Discuss any security considerations.

See:

- https://github.com/opengeospatial/oapi_common/blob/master/core/clause_12_security_considerations.adoc
- https://github.com/opengeospatial/ogcapi-features/blob/master/core/standard/clause_11_security_considerations.adoc

Annex A: Conformance Class Abstract Test Suite (Normative)

NOTE	Ensure that there is a conformance class for each requirements class and a test for each requirement (identified by requirement name and number)

A.1. Conformance Class A

A.1.1. Requirement 1

Test id:	/conf/conf-class-a/req-name-1
Requirement:	/req/req-class-a/req-name-1
Test purpose:	Verify that…
Test method:	Inspect…

A.1.2. Requirement 2

Annex B: Bibliography

- W3C/OGC: Spatial Data on the Web Best Practices, W3C Working Group Note 28 September 2017, https://www.w3.org/TR/sdw-bp/
- W3C: Data on the Web Best Practices, W3C Recommendation 31 January 2017, https://www.w3.org/TR/dwbp/
- W3C: Data Catalog Vocabulary, W3C Recommendation 16 January 2014, https://www.w3.org/TR/vocab-dcat/
- IANA: Link Relation Types, https://www.iana.org/assignments/link-relations/link-relations.xml

Annex C: Revision History

Date	Release	Editor	Primary clauses modified	Description
2020-01-13	Template	C. Heazel	all	initial template
2020-04-22	1.0-dev	T. Kralidis	all	changes to reflect Records, editorial updates, add Q parameter to clause 7