OGC API - Coverages - Part 1

Core

Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: 2019-12-18

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

Internal reference number of this OGC® document: 19-087

Version: 0.0.5

Category: OGC® Implementation Specification

Editor: Charles Heazel

OGC API - Coverages - Part 1: Core

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

<Insert Abstract Text here>

ii. Keywords

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

ogcdoc, OGC document, <tags separated by commas>

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)

v. Submitters

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

Name	Affiliation
Charles Heazel	HeazelTech
Stephan Meissl <stephan.meissl@eox.at>	EOX IT Services GmbH

Chapter 1. Scope

This OGC API Coverages (https://github.com/opengeospatial/ogc_api_coverages) specification establishes how to access coverages as defined by the Coverage Implementation Schema (CIS) 1.1 (http://docs.opengeospatial.org/is/09-146r6/09-146r6.html) through a Web API such as those described by the OpenAPI specification (https://www.openapis.org/).

The functionality provided by API-Coverages resembles that of the OGC Web Coverage Service (WCS) 2.1 Interface Standard. It is expected that Coverage APIs and WCS services will be able to interoperate, allowing developers to pick the solution best suited for their requirements.

1.1. Current scope:

The OGC is using an incremental approach to their API development. The initial goal is to develop a relatively simple API standard which will meet the needs of a large percentage of implementors. Additional capabilities will be added based on community demand.

As a result, this API-Coverages standard does not provide a full duplication of the WCS capabilities. The restrictions are:

- Only gridded coverages are addressed, not MultiPoint/Curve/Surface/SolidCoverages. Reason is that gridded coverages receive most attention today.
- Only GeneralGridCoverage is addressed, other coverage types will follow later. Reason is to have a first version early which shows and allows to evaluate the principles.
- Only coverage extraction functionality is considered, not general processing (as is provided with Web Coverage Service (WCS) extensions such as the Processing Extension). Exceptions from this rule are subsetting including band subsetting, scaling, and CRS conversion and data format encoding, given their practical relevance.
- Subsetting is considered in the query component only for now. As typically all dimensions in a coverage are of same importance subsetting might not fit perfectly in the hierarchical nature of the path component. Further, subsetting may reference any axis and leave out any other, which makes positional parameters unsuitable. Nevertheless subsetting in the path component particularly limited to fixed subsets might be considered in a future version.

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 idenfies TBD 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 an implementation must meet.

The Requirements Classes for OGC API-Coverages are:

- **Core**

The *Core* Requirements Class is the minimal useful service interface for an OGC Coverages API. The requirements specified in this Requirements Class are mandatory for all implementations of API-Coverages.

2.1. Encodings

The *Core* Requirements Class does not mandate a specific encoding or format for representing resources. The encoding Requirements Classes defined for this version of API-Coverages are:

- **HTML**
- **GeoJSON**
- **RFD**
- **NetCDF**
- **GeoTIFF**
- **JPEG-2000**

The *HTML* and *GeoJSON* Requirements Classes specify commonly used encodings for spatial data on the web. Use of these encodings is recommended for resources which describe the coverage resources.

Many requirements in this standard include wording similar to the following:

```
"encoded using a schema equivalent to the JSON fragment at"
```

"Equivalent to" shall be interpreted to mean a schema which has the same information content as the JSON fragment.

NOTE	We may want to re-work the requirements so that they point to an appropriate requirements in the encoding Requirements Class.

NOTE	Question, Can I use a format which is not on the list of recognized Conformance Classes?

2.2. API Description

- **OpenAPI 3.0**

API-Coverages does not mandate any encoding of format for the formal definition of the API. The prefered option is the OpenAPI 3.0 specification. The *OpenAPI 3.0* requirements class has been specified for APIs implementing OpenAPI 3.0.

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
- IETF RFC 7946: **The GeoJSON Format**, eoJSON
- Nottingham, M.: IETF RFC 8288, **Web Linking**, RFC 8288
- International Telecommunication Union, **ITU-T.800 : Information technology - JPEG 2000 image coding system: Core coding system**, June, 2019, https://www.itu.int/rec/T-REC-T.800-201906-I/en
- OGC 19-072: **OGC API (OAPI) Common Specification**, (Draft) API Common
- OGC 09-146: **OGC Coverage Implementation Schema (CIS)**, version 1.1, CIS
- OGC 19-008: **OGC GeoTIFF Standard**, Version 1.1, http://docs.opengeospatial.org/is/19-008r4/19-008r4.html
- OGC Schema: **OGC JSON Schema for Coverage Implementation Schema**, version 1.1, 2017, CIS Schema
- OGC 10-090: **OGC Network Common Data Form (NetCDF) Core Encoding Standard**, Version 1.0, http://portal.opengeospatial.org/files/?artifact_id=43732
- OGC 17-089: **OGC Web Coverage Service (WCS) Interface Standard - Core**, Version 2.1, (WCS 2.1)
- 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/
- OGC: OGC 07-011, Abstract Specification Topic 6: The Coverage Type and its Subtypes, version 7.0 (identical to ISO 19123:2005), 2007
- 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
- OGC: OGC 08-094, OGC® SWE Common Data Model Encoding Standard, version 2, 2011
- OGC: OGC 12-000, OGC® SensorML: Model and XML Encoding Standard, version 2, 2014

- OGC: OGC 09-146r2, GML 3.2.1 Application Schema – Coverages, version 1.0.1, 2012
- OGC: OGC 16-083, Coverage Implementation Schema – ReferenceableGridCoverage Extension, version 1, 2017
- OGC: OGC 09-110r4, Web Coverage Service (WCS) Core Interface Standard, version 2, 2012
- OGC: OGC 13-102r2, Name type specification – Time and index coordinate reference system definitions (OGC Policy Document), version 1, 2014
- OGC: OGC 14-121, Web Information Service (WIS), version 1 (unpublished)
- 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 Clause 5 of [OGC 17-072] and in 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. Coverage

feature that acts as a function to return values from its range for any direct position within its spatiotemporal domain, as defined in OGC Abstract Topic 6

4.2. Regular grid

grid whose grid lines have a constant distance along each grid axis

4.3. Irregular grid

Grid whose grid lines have individual distances along each grid axis

4.4. Displaced grid

grid whose direct positions are topologically aligned to a grid, but whose geometric positions can vary arbitrarily

4.5. Mesh

coverage consisting of a collection of curves, surfaces, or solids, respectively

4.6. Partition [of a coverage]

separately stored coverage acting, by being referenced in the coverage on hand, as one of its components

4.7. Sensor model

mathematical model for estimating geolocations from recorded sensor data such as digital imagery

4.8. Transformation grid

grid whose direct positions are given by some transformation algorithm not further specified in this standard

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-coverages-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 Table 2. 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.

Whenever a data item from a CIS-external namespace is referenced this constitutes a normative dependency on the data structure imported together with all requirements defined in the namespace referenced.

Table 1. Namespace mapping conventions

UML prefix	GML prefix	Namespace URL	Description
CIS	cis	http://www.opengis.net/cis/1.1	Coverage Implementation Schema 1.1
CIS10	cis10	http://www.opengis.net/gmlcov/1.0	Coverage Implementation Schema 1.0
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
SWE Common	swe	http://www.opengis.net/swe/2.0	SWE Common 2.0
SML	sml	http://www.opengis.net/sensorml/2.0	SensorML 2.0

Chapter 6. Overview

6.1. General

OGC API standards define modular API building blocks to spatially enable Web APIs in a consistent way. OpenAPI is used to define the reusable API building blocks with responses in JSON and HTML.

The OGC API family of standards is organized by resource type. OGC API-Coverages specifies the fundamental API building blocks for interacting with coverages. The spatial data community uses the term 'coverage' for homogeneous collections of values located in space/time, such as spatio-temporal sensor,image, simulation, and statistics data.

If you are unfamiliar with the term 'coverage', the explanations on the Coverages DWG Wiki provide more detail and links to educational material. Additionally, Coverages: describing properties that vary with location (and time) in the W3C/OGC Spatial Data on the Web Best Practice document may be considered.

This OGC API - Coverages specification establishes how to access coverages as defined by the Coverage Implementation Schema (CIS) 1.1 through Web APIs.

6.1.1. Current scope

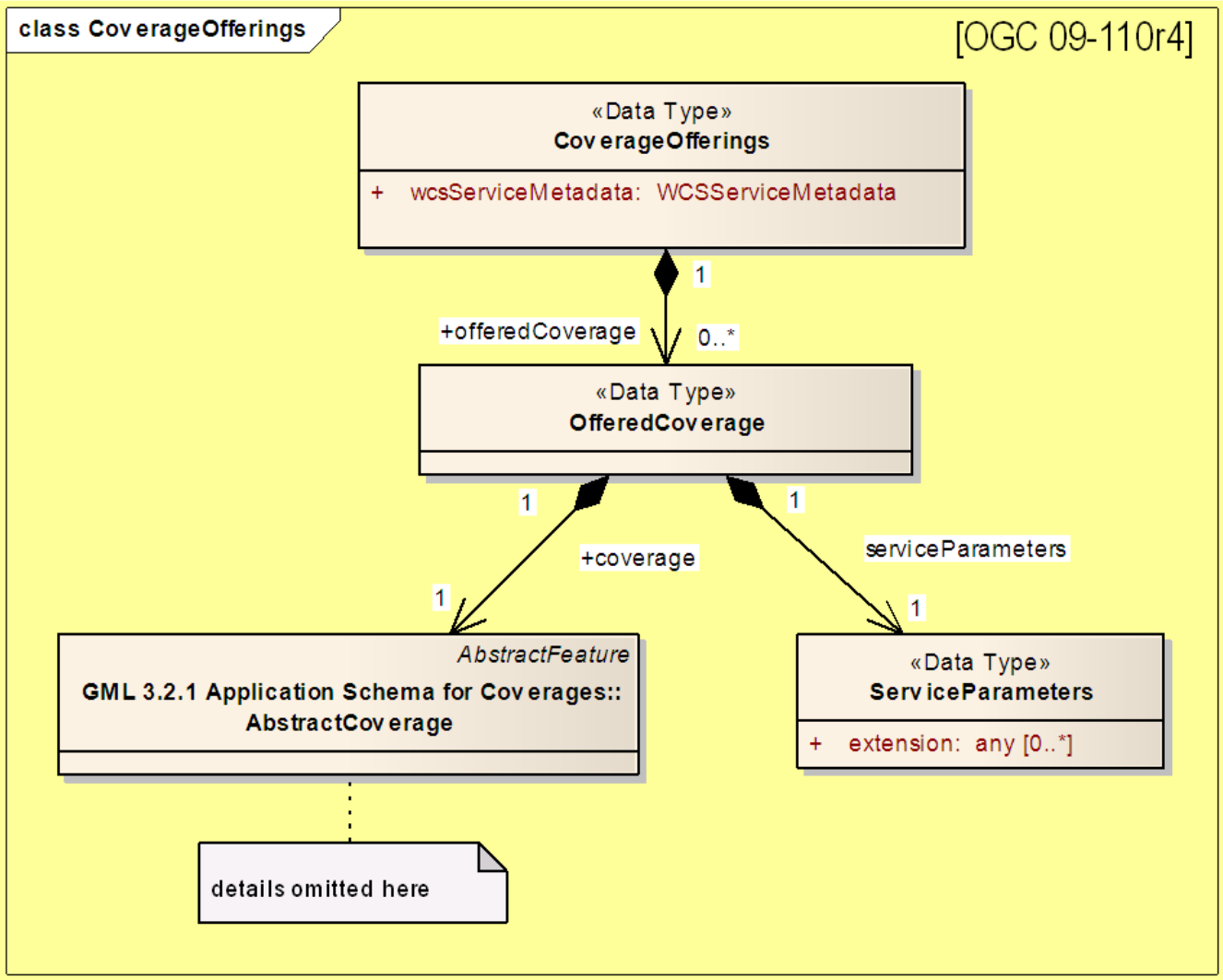
- Only gridded coverages are addressed, not MultiPoint/Curve/Surface/SolidCoverages. Reason is that griddedcoverages receive most attention today.
- Only CIS 1.1 GeneralGridCoverage is addressed, other coverage types will follow later. Reason is to have a first version early which showsand allows to evaluate the principles.
- Only coverage extraction functionality is considered, not general processing (as is provided with Web Coverage Service (WCS) extensionssuch as the Processing Extension). Exceptions from this rule aresubsetting including band subsetting, scaling, and CRS conversion anddata format encoding, given their practical relevance.
- Subsetting is considered in the query component only for now. As typically all dimensions in a coverage are of same importance subsetting might not fit perfectly in the hierarchical nature of the pathcomponent. Further, subsetting may reference any axis and leave out anyother, which makes positional parameters unsuitable. Neverthelesssubsetting in the path component particularly limited to fixed subsetsmight be considered in a future version. The principles of the `/tiles` path defined in OGC API - Maps & Tiles might be a good fit and shall be explored.

As such, the functionality provided below resembles OGC Web Coverage Service (WCS) 2.1 Interface Standard - Core.

6.1.2. Background Information: WCS Service Model

OGC WCS 2 has an internal model of its storage organization based on which the classic operations GetCapabilities, DescribeCoverage, and GetCoverage can be explained naturally. This model consists of a single CoverageOffering resembling the complete WCS data store. It holds some service metadata describing service qualities (such as WCS extensions, encodings, CRSs, and interpolations supported, etc.). At its heart, this offering holds any number of OfferedCoverages. These contain the

coverage payload to be served, but in addition can hold coverage-specific service-related metadata (such as the coverage's Native CRS).



Discussion has shown that the OpenAPI functionality 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.

6.1.3. Prinicples

OpenAPI establishes URL-based access patterns, as defined by RFC 3986 "Uniform Resource Identifier (URI): Generic Syntax", RFC 3987 "Internationalized Resource Identifiers (IRIs)", and RFC 6570 "URI Template" following a syntax like http://www.acme.com/path/to/resource/{id}{?query,parameters} whereby

- `/path/to/resource/{id}` defines the local path to the resource to be retrieved on the node identified by the head part, http://www.acme.com.
- the `{?query,parameters}` represent key/value pairs with further parameters passed to the request.

As a general rule,

- accessing a coverage component is done in the path section,
- subsetting is done in the query parameter section
- format encoding is controlled via HTTP headers

As the coverage model normatively is given by the corresponding XML schema (JSON and RDF are built in sync with XML) specification of the OpenAPI access paths follows this schema. Note, though, that OWS Common, while normatively referenced in CIS, is not followed by OpenAPI, so here deviations will occur.

For path expressions abbreviations (i.e., aliases) may be defined for convenience.

Chapter 7. Requirements Class "Core"

7.1. Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-coverages-1/1.0/req/core	
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

The OGC API family of standards is organized by resource type. OGC API - Coverages specifies the fundamental API building blocks for interacting with coverages. The spatial data community uses the term 'coverage' for homogeneous collections of values located in space/time, such as spatio-temporal sensor, image, simulation, and statistics data.

If you are unfamiliar with the term 'coverage', the explanations on the Coverages DWG Wiki provide more detail and links to educational material. Additionally, Coverages: describing properties that vary with location (and time) in the W3C/OGC Spatial Data on the Web Best Practice document may be considered.

This OGC API - Coverages specification establishes how to access coverages as defined by the Coverage Implementation Schema (CIS) 1.1 through Web APIs. A high-level view of the CIS data model is provided in Figure 1.

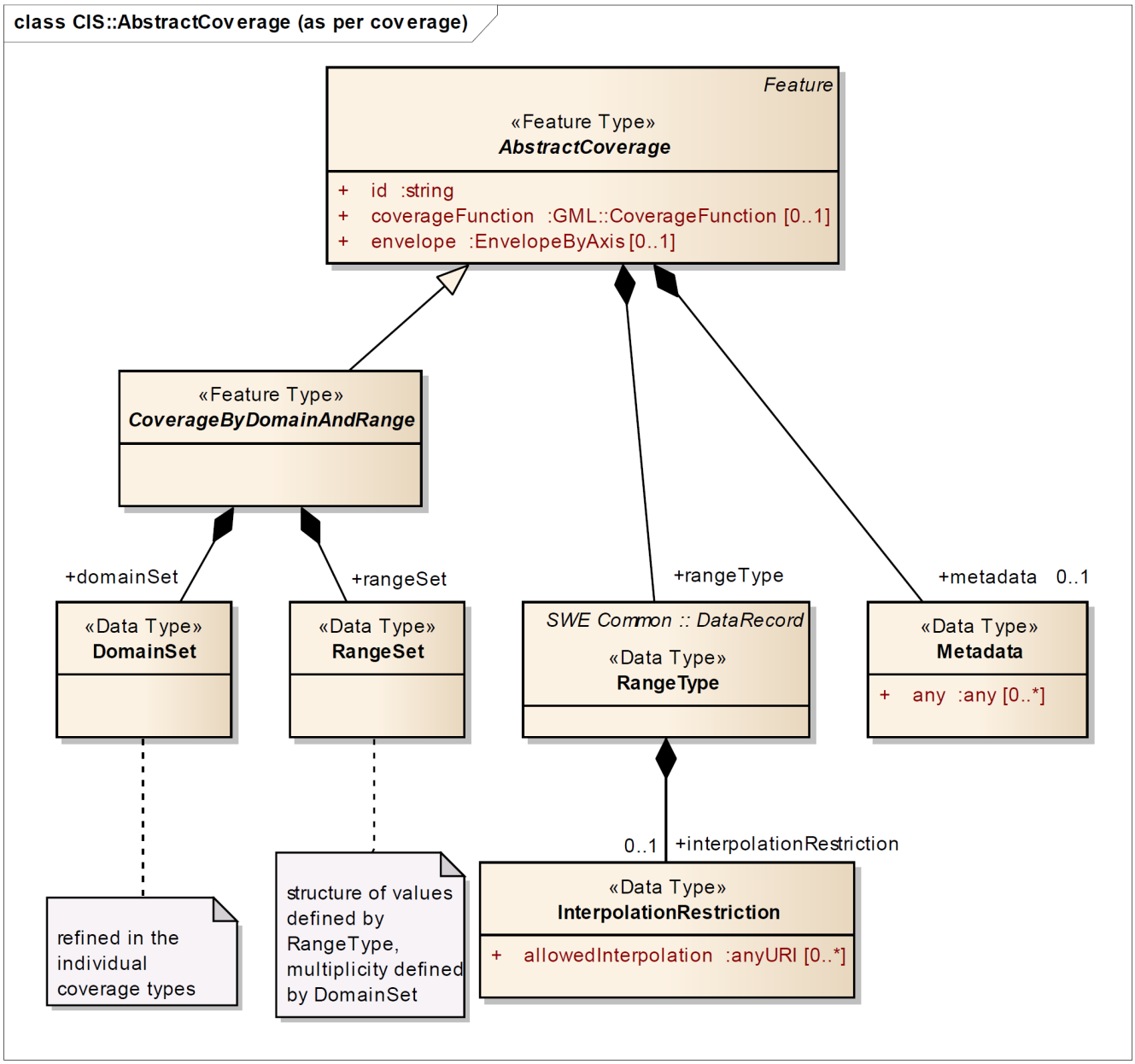


Figure 1. Abstract Coverage

This `Core` Requirements Class defines the requirements for locating, understanding, sub-setting, and accessing coverage resources. However, it does not specify any formats for the returned resources. The formats available to implementors of this standard are specified in additional Requirements Classes.

7.2. Dependencies

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

Requirement 1	**/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/core
B	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/collections

Table 2 Identifies the 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 2. Mapping API-Coverages Sections to API-Common Requirements Classes

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

7.3. Path Summary

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

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

Table 3 summarizes the access paths and relation types defined in this standard.

Table 3. Coverage API Paths

Path Template	Relation	Resource
Common		
{root}/	none	Landing page

Path Template	Relation	Resource
{root}/api	service-desc or service-doc	API Description (optional)
{root}/conformance	conformance	Conformance Classes
{root}/collections	data	Metadata describing the spatial collections (coverages) available from this API.
{root}/collections/{coverageId}		Metadata describing the coverge which has the unique identifier {coverageId}
Coverages		
{root}/collections/{coverageId}/coverage	items	A general description of the coverage identified by {coverageId} including the coverage's envelope.
{root}/collections/{coverageId}/coverage/description	tbd	returns the whole coverage description consisting of domainset, rangetype, and metadata (but not the rangeset)
{root}/collections/{coverageId}/coverage/domainset	tbd	returns the coverage's domain set definition
{root}/collections/{coverageId}/coverage/rangetype	tbd	returns the coverage's range type information (i.e., a description of the data semantics)
{root}/collections/{coverageId}/coverage/metadata	tbd	returns the coverage's metadata (may be empty)
{root}/collections/{coverageId}/coverage/rangeset	tbd	returns the coverage's range set, i.e., the actual values in the coverage's Native Format (see format encoding for ways to retrieve in specific formats)
{root}/collections/{coverageId}/coverage/all	tbd	returns all of the above namely the coverage's domainset, rangetype, metadata, and rangeset comparable to a GetCoverage response

Where:

- {root} = Base URI for the API server
- {coverageId} = an identifier for a specific coverage (collection)

7.4. Platform

7.4.1. API landing page

The landing page provides links to start exploring the resources offered by the API instance. It mainly consists of a list of links. OGC API-Common already requires some common links. Those links are sufficient for this core.

Table 4. Dependencies

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

7.4.1.1. Operation

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

1. Follow the `{root}/` path

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

7.4.1.2. Response

Requirement 2	**/req/core/root-success**
A	A successful execution of the operation SHALL be reported as a response with an HTTP status code `200`.
B	The content of that response SHALL provide the information needed to begin using the API as defined in the JSON schema landingPage.json
C	The response SHALL include links to the following resources: • the API definition (relation type 'service-desc' or 'service-doc') • `/conformance` (relation type 'conformance') • one or more information resources (relation type 'data')

The Response to the Landing Page operation is defined in API-Common. However, this resource is extended for use with Coverages. The schema for this resource is provided in Landing Page Response Schema.

Landing Page Response Schema

```
type: object
required:
  - links
properties:
  title:
    type: string
    example: Coverages over Bonn
  description:
    type: string
    example: Access to coverages which include the city of Bonn via a Web API that
conforms to the OGC API Coverages specification.
  links:
    type: array
    items:
      $ref: common/link.yaml
```

The following JSON fragment is an example of a response to an OGC API-Coverages 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/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" }
  ]
}
```

7.4.1.3. Error situations

See HTTP Status Codes for general guidance.

7.4.2. API definition

Every API is expected to provide a definition that describes the capabilities of that API. This definition 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 5. Dependencies

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

7.4.2.1. Operation

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

1. Follow the /api path
2. Follow the `service-desc` or `service-doc` link on the landing page

Only the link is required by API-Common.

7.4.2.2. Response

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, there is only one option, OpenAPI version 3.0.

7.4.2.3. Error situations

See HTTP status codes for general guidance.

7.4.3. 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 implements and conforms to.

Table 6. Dependencies

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

7.4.3.1. Operation

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

1. Follow the /conformance path
2. Follow the `conformance` link on the landing page

Both techniques are required by API-Common.

7.4.3.2. Response

Requirement 3	**/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-coverages-1/1.0/conf/core

The 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.

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-Coverages 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-common-1/1.0/conf/html",
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/geojson",
    "http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/core"
  ]
}
```

7.4.3.3. Error situations

See HTTP status codes for general guidance.

7.5. Coverage Access Paths

In this clause, the path component of OGC API - Coverages access is established.

Access paths follow the XML Schema of CIS in their structure.

To access coverage service constituents, such as formats supported, OGC 14-121 Web Query Service provides guidance, see https://github.com/opengeospatial/ogc_api_coverages/blob/master/CIS%2BWCS-standards/14-121_Web-Query-Service_2016-06-19.pdf.

7.5.1. Collections

TBD

Table 7. Dependencies

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

7.5.1.1. Operation

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

1. Follow the `/collections` path
2. Follow the `data` link on the landing page

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

7.5.1.2. Response

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-Coverages Collections operation.

Collections Example

```
{
  "TBD": [
    "filler1",
    "filler2"
  ]
}
```

7.5.1.3. Error situations

See HTTP status codes for general guidance.

7.5.2. Collection Information

TBD

Table 8. Dependencies

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

7.5.2.1. Operation

This operation is defined in the `Collections` conformance class of API-Common. No modifications are required to support `Coverage` resources. However, on a coverages API the the collections are also coverages. So in this standard the term `coverageId` is used instead of `collectionId`. The two terms are equivalent.

The `Collections` conformance class describes two ways of performing this operation:

1. Follow the `/collections/{coverageId}` path
2. Follow the `dataset` link on the collections info page

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

The {coverageId} parameter is the unique identifier for a single coverage on the API.

7.5.2.2. Response

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-Coverages Collection Information operation.

Collection Information Example

```
{
  "TBD": [
    "filler1",
    "filler2"
  ]
}
```

7.5.2.3. Error situations

See HTTP status codes for general guidance.

7.6. Coverage Access

This section is about coverage access, which is driven by the coverage structure.

The paths discussed in this section are all branches off of the `/collections/{coverageId}/coverage` root.

7.6.1. Coverage Offering

Returns a general coverage offering description consisting of envelope, rangetype, and service metadata such as the coverage's native format

7.6.1.1. Operation

Requirement 4	**/req/core/cov-offer-op**
A	The API SHALL support the HTTP GET operation at the path `/collections/{coverageId}/coverage`. • `coverageId` is the local identifier for a `Coverage`. It serves the same role and is subject to the same requirements as the `collectionId` parameter defined in API-Common.

7.6.1.2. Response

Requirement 5	**/req/core/cov-offer-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 provide a general description of the coverage, including the coverage's envelope, as defined in the JSON schema coverage_offering.json.

C	The response SHALL include links to the following resources using the specified link relations: • Resource 1: `reltype` • Resource 2: `reltype`

Coverage Offering Response Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "coverageOffering": {
        "title": "Coverage Offering",
        "description": "Description of the Coverage Offering",
        "type": "object",
        "properties": {
            "propertyName": {
                "description": "The propertyName description",
                "type": "string"
            }
        }
    }
}
```

The following JSON fragment is an example of a response to a Coverage Offering request.

Coverage Offering Example

```
{
  "TBD": [
    "filler1",
    "filler2"
  ]
}
```

7.6.1.3. Error situations

See HTTP status codes for general guidance.

7.6.2. Coverage Description

Returns the whole coverage description consisting of domainset, rangetype, and metadata (but not the rangeset)

7.6.2.1. Operation

Requirement 6	**/req/core/cov-desc-op**

A	The API SHALL support the HTTP GET operation at the path `/collections/{coverageId}/coverage/description`. • `coverageId` is the local identifier for a `Coverage`. It serves the same role and is subject to the same requirements as the `collectionId` parameter defined in API-Common.

7.6.2.2. Response

Requirement 7	**/req/core/cov-desc-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 provide a description of the coverage as defined in the JSON schema coverage_description.json.
C	The response SHALL include links to the following resources using the specified link relations: • Resource 1: `reltype` • Resource 2: `reltype`

Coverage Description Response Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "coverageDescription": {
        "title": "Coverage Description",
        "description": "Description of the Coverage",
        "type": "object",
        "properties": {
            "propertyName": {
                "description": "The propertyName description",
                "type": "string"
            }
        }
    }
}
```

The following JSON fragment is an example of a response to a Coverage Description request.

Coverage Description Example

```
{
  "TBD": [
    "filler1",
    "filler2"
  ]
}
```

7.6.2.3. Error situations

See HTTP status codes for general guidance.

7.6.3. Coverage Domain Set

returns the coverage's domain set definition

7.6.3.1. Operation

Requirement 8	**/req/core/cov-ds-op**
A	The API SHALL support the HTTP GET operation at the path `/collections/{coverageId}/coverage/domainset`. • `coverageId` is the local identifier for a `Coverage`. It serves the same role and is subject to the same requirements as the `collectionId` parameter defined in API-Common.

7.6.3.2. Response

Requirement 9	**/req/core/cov-ds-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 provide the `Domain Set` definition of the coverage as defined in the JSON schema coverage_domainset.json.
C	The response SHALL include links to the following resources using the specified link relations: • Resource 1: `reltype` • Resource 2: `reltype`

Coverage Domain Set Response Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "domainSet": {
        "title": "domainSet",
        "description": "The domainSet describes the *direct positions* of the
coverage, i.e., the locations for which values are available.",
        "type": "object",
        "oneOf": [
            {
                "required": [
                    "type",
                    "generalGrid"
                ],
                "properties": {
                    "type": {
                        "enum": ["DomainSetType"]
                    },
                    "generalGrid": {
                        "title": "General Grid",
                        "description": "A general n-D grid is defined through a
sequence of axes, each of which can be of a particular axis type.",
                        "type": "object",
                        "required": ["type"],
                        "additionalProperties": false,
                        "properties": {
                            "type": {
                                "enum": ["GeneralGridCoverageType"]
                            },
                            "id": {"type": "string"},
                            "srsName": {
                                "type": "string",
                                "format": "uri"
                            },
                            "axisLabels": {
                                "type": "array",
                                "items": {"type": "string"}
                            },
                            "axis": {
                                "type": "array",
                                "items": {
                                    "type": "object",
                                    "oneOf": [
                                        {
                                            "title": "Index Axis",
                                            "description": "An Index Axis is an axis
with only integer positions allowed.",
                                            "required": [
                                                "type",
                                                "axisLabel",
                                                "lowerBound",
                                                "upperBound"
```

```
                ],
                "additionalProperties": false,
                "properties": {
                    "type": {
                        "enum": ["IndexAxisType"]
                    },
                    "id": {"type": "string"},
                    "axisLabel": {"type": "string"},
                    "lowerBound": {"type": "number"},
                    "upperBound": {"type": "number"}
                }
            },
            {
                "title": "Regular Axis",
                "description": "A Regular Axis is an axis
where all direct coordinates are at a common distance from its immediate neighbors.",
                "required": [
                    "type",
                    "axisLabel",
                    "lowerBound",
                    "upperBound",
                    "resolution",
                    "uomLabel"
                ],
                "additionalProperties": false,
                "properties": {
                    "type": {
                        "enum": ["RegularAxisType"]
                    },
                    "id": {"type": "string"},
                    "axisLabel": {"type": "string"},
                    "lowerBound": {
                        "type": [
                            "number",
                            "string",
                            "null",
                            "boolean"
                        ]
                    },
                    "upperBound": {
                        "type": [
                            "number",
                            "string",
                            "null",
                            "boolean"
                        ]
                    },
                    "uomLabel": {"type": "string"},
                    "resolution": {"type": "number"}
                }
            },
```

```
                                        {
                                            "title": "Irregular Axis",
                                            "description": "An irregular axis
enumerates all possible direct position coordinates.",
                                            "required": [
                                                "type",
                                                "axisLabel",
                                                "uomLabel",
                                                "coordinate"
                                            ],
                                            "additionalProperties": false,
                                            "properties": {
                                                "type": {
                                                    "enum": ["IrregularAxisType"]
                                                },
                                                "id": {"type": "string"},
                                                "axisLabel": {"type": "string"},
                                                "uomLabel": {"type": "string"},
                                                "coordinate": {
                                                    "type": "array",
                                                    "items": {
                                                        "type": [
                                                            "number",
                                                            "string",
                                                            "boolean"
                                                        ]
                                                    }
                                                }
                                            }
                                        }
                                    ]
                                }
                            },
                            "displacement": {
                                "title": "Displacement",
                                "description": "A Displacement is a warped axis nest
where points on the grid all have their individual direct position coordinates. The
sequenceRule element describes linearization order.",
                                "type": "object",
                                "oneOf": [
                                    {
                                        "required": [
                                            "type",
                                            "axisLabels",
                                            "uomLabels",
                                            "coordinates"
                                        ],
                                        "properties": {
                                            "type": {
                                                "enum": ["DisplacementAxisNestType"]
                                            },
```

```
                                        "id": {"type": "string"},
                                        "axisLabel": {"type": "string"},
                                        "srsName": {
                                            "type": "string",
                                            "format": "uri"
                                        },
                                        "axisLabels": {
                                            "type": "array",
                                            "items": {"type": "string"}
                                        },
                                        "uomLabels": {
                                            "type": "array",
                                            "items": {"type": "string"}
                                        },
                                        "coordinates": {
                                            "type": "array",
                                            "items": {
                                                "type": "array",
                                                "items": {
                                                    "type": [
                                                        "number",
                                                        "string",
                                                        "boolean"
                                                    ]
                                                }
                                            }
                                        }
                                    }
                                },
                                {
                                    "required": [
                                        "type",
                                        "axisLabels",
                                        "uomLabels",
                                        "coordinatesRef"
                                    ],
                                    "properties": {
                                        "type": {
                                            "enum":
["DisplacementAxisNestTypeRef"]
                                        },
                                        "id": {"type": "string"},
                                        "axisLabel": {"type": "string"},
                                        "srsName": {
                                            "type": "string",
                                            "format": "uri"
                                        },
                                        "axisLabels": {
                                            "type": "array",
                                            "items": {"type": "string"}
                                        },
```

```
                                        "uomLabels": {
                                            "type": "array",
                                            "items": {"type": "string"}
                                        },
                                        "coordinatesRef": {
                                            "type": "string",
                                            "format": "uri"
                                        }
                                    }
                                }
                            ]
                        },
                        "model": {
                            "title": "Sensor model",
                            "description": "A Transformation By Sensor Model is a
transformation definition which is given by a SensorML 2.0 transformation
specification.",
                            "type": "object",
                            "required": [
                                "type",
                                "sensorModelRef"
                            ],
                            "properties": {
                                "type": {
                                    "enum": ["TransformationBySensorModelType"]
                                },
                                "id": {"type": "string"},
                                "axisLabels": {
                                    "type": "array",
                                    "items": {"type": "string"}
                                },
                                "uomLabels": {
                                    "type": "array",
                                    "items": {"type": "string"}
                                },
                                "sensorModelRef": {
                                    "type": "string",
                                    "format": "uri"
                                },
                                "sensorInstanceRef": {
                                    "type": "string",
                                    "format": "uri"
                                }
                            }
                        },
                        "gridLimits": {
                            "title": "Grid limits",
                            "description": "This is the boundary of the array
underlying the grid, given by its diagonal corner points in integer _60_3D. The grid
limits can be omitted in case all axes are of type index axis, because then it repeats
the grid information in a redundant way. The purpose of the axisLabels attribute,
```

```
which lists the axis labels of all axisExtent elements in proper sequence, is to
enforce axis sequence also in XML systems which do not preserve document order.",
                                "type": "object",
                                "required": ["type"],
                                "properties": {
                                    "indexAxis": {
                                        "title": "Index Axis",
                                        "description": "An Index Axis is an axis with
only integer positions allowed.",
                                        "type": "object",
                                        "required": [
                                            "type",
                                            "lowerBound",
                                            "upperBound"
                                        ],
                                        "additionalProperties": false,
                                        "properties": {
                                            "type": {
                                                "enum": ["IndexAxisType"]
                                            },
                                            "id": {"type": "string"},
                                            "axisLabel": {"type": "string"},
                                            "lowerBound": {"type": "number"},
                                            "upperBound": {"type": "number"}
                                        }
                                    },
                                    "srsName": {
                                        "type": "string",
                                        "format": "uri"
                                    },
                                    "axisLabels": {
                                        "type": "array",
                                        "items": {"type": "string"}
                                    }
                                }
                            }
                        }
                    }
                }
            },
            {
                "required": [
                    "type",
                    "directMultiPoint"
                ],
                "properties": {
                    "type": {
                        "enum": ["DomainSetType"]
                    },
                    "directMultiPoint": {
                        "oneOf": [
```

```
                        {
                            "required": [
                                "type",
                                "coordinates"
                            ],
                            "properties": {
                                "type": {
                                    "enum": ["DirectMultiPointType"]
                                },
                                "coordinates": {
                                    "type": "array",
                                    "items": {
                                        "type": "array",
                                        "items": {
                                            "type": [
                                                "number",
                                                "string",
                                                "boolean"
                                            ]
                                        }
                                    }
                                }
                            }
                        },
                        {
                            "required": [
                                "type",
                                "coordinatesRef"
                            ],
                            "properties": {
                                "type": {
                                    "enum": ["DirectMultiPointTypeRef"]
                                },
                                "coordinatesRef": {
                                    "type": "string",
                                    "format": "uri"
                                }
                            }
                        }
                    ]
                }
            }
        },
        {
            "required": [
                "type",
                "fileReference"
            ],
            "properties": {
                "type": {
                    "enum": ["DomainSetRefType"]
```

```
                    },
                    "id": {
                        "type": "string",
                        "format": "uri"
                    },
                    "fileReference": {
                        "type": "string",
                        "format": "uri"
                    }
                }
            }
        ]
    }
}
```

The following JSON fragment is an example of a response to a Coverage DomainSet request.

Coverage DomainSet Example

```
{
  "TBD": [
    "filler1",
    "filler2"
  ]
}
```

7.6.3.3. Error situations

See HTTP status codes for general guidance.

7.6.4. Coverage Range Type

returns the coverage's range type information (i.e., a description of the data semantics)

7.6.4.1. Operation

Requirement 10	**/req/core/cov-rt-op**
A	The API SHALL support the HTTP GET operation at the path `/collections/{coverageId}/coverage/rangetype`. • `coverageId` is the local identifier for a `Coverage`. It serves the same role and is subject to the same requirements as the `collectionId` parameter defined in API-Common.

7.6.4.2. Response

Requirement 11	/req/core/cov-rt-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 describe the `Range Type` of the coverage as defined in the JSON schema coverage_rangetype.json.
C	The response SHALL include links to the following resources using the specified link relations: • Resource 1: `reltype` • Resource 2: `reltype`

Coverage Range Type Response Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "rangeType": {
        "title": "rangeType",
        "description": "The rangeType element describes the structure and semantics of
a coverage's range values, including (optionally) restrictions on the interpolation
allowed on such values.",
        "type": "object",
        "oneOf": [
            {
                "required": [
                    "type",
                    "field"
                ],
                "properties": {
                    "type": {
                        "enum": ["DataRecordType"]
                    },
                    "field": {
                        "type": "array",
                        "items": {
                            "title": "quantity",
                            "description": "quantiy",
                            "type": "object",
                            "required": ["type"],
                            "properties": {
                                "type": {
                                    "enum": ["QuantityType"]
                                },
                                "id": {
                                    "type": "string",
                                    "format": "uri"
```

```
                    },
                    "name": {"type": "string"},
                    "definition": {
                        "type": "string",
                        "format": "uri"
                    },
                    "uom": {
                        "title": "units of measure",
                        "description": "units of measure",
                        "type": "object",
                        "required": [
                            "type",
                            "code"
                        ],
                        "properties": {
                            "type": {
                                "enum": ["UnitReference"]
                            },
                            "id": {
                                "type": "string",
                                "format": "uri"
                            },
                            "code": {"type": "string"}
                        }
                    },
                    "constraint": {
                        "title": "Constraint",
                        "description": "Constraint",
                        "type": "object",
                        "required": ["type"],
                        "properties": {
                            "type": {
                                "enum": ["AllowedValues"]
                            },
                            "id": {
                                "type": "string",
                                "format": "uri"
                            },
                            "interval": {
                                "type": "array",
                                "items": {
                                    "type": [
                                        "number",
                                        "string",
                                        "boolean"
                                    ]
                                }
                            }
                        }
                    }
                }
```

```
                        }
                    },
                    "interpolationRestriction": {
                        "title": "interpolationRestriction",
                        "description": "Interpolation restriction",
                        "type": "object",
                        "required": ["type"],
                        "properties": {
                            "type": {
                                "enum": ["InterpolationRestrictionType"]
                            },
                            "id": {
                                "type": "string",
                                "format": "uri"
                            },
                            "allowedInterpolation": {
                                "type": "array",
                                "items": {
                                    "type": "string",
                                    "format": "uri"
                                }
                            }
                        }
                    }
                }
            },
            {
                "required": [
                    "type",
                    "fileReference"
                ],
                "properties": {
                    "type": {
                        "enum": ["RangeTypeRefType"]
                    },
                    "id": {
                        "type": "string",
                        "format": "uri"
                    },
                    "fileReference": {
                        "type": "string",
                        "format": "uri"
                    }
                }
            }
        ]
    }
}
```

The following JSON fragment is an example of a response to a Coverage RangeType request.

Coverage RangeType Example

```
{
  "TBD": [
    "filler1",
    "filler2"
  ]
}
```

7.6.4.3. Error situations

See HTTP status codes for general guidance.

7.6.5. Coverage Range Set

returns the coverage's range set, i.e., the actual values in the coverage's Native Format (see format encoding for ways to retrieve inspecific formats)

7.6.5.1. Operation

Requirement 12	**/req/core/cov-rs-op**
A	The API SHALL support the HTTP GET operation at the path `/collections/{coverageId}/coverage/rangeset`. • `coverageId` is the local identifier for a `Coverage`. It serves the same role and is subject to the same requirements as the `collectionId` parameter defined in API-Common.

7.6.5.2. Response

Requirement 13	**/req/core/cov-rs-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 the `Range Set` of the coverage as defined in the JSON schema coverage_rangeset.json.
C	The response SHALL include links to the following resources using the specified link relations: • Resource 1: `reltype` • Resource 2: `reltype`

Coverage Range Set Response Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "rangeSet": {
        "title": "rangeSet",
        "description": "The rangeSet lists a value for each of the coverage's direct
positions. Values resemble the *payload* information of some particular direct
positions. Values can be composite (with a single nesting level, i.e.: composites
always consist of atomics), or atomic (emulated through single-component composites)
whereby the sequence, structure, and meaning of every value is defined through the
rangeType. Values can be represented in-line or by reference to an external file which
may have any suitable encoding.",
        "type": "object",
        "oneOf": [
            {
                "required": [
                    "type",
                    "dataBlock"
                ],
                "properties": {
                    "type": {
                        "enum": ["RangeSetType"]
                    },
                    "dataBlock": {
                        "title": "dataBlock",
                        "description": "Data block objects",
                        "type": "object",
                        "required": [
                            "type",
                            "values"
                        ],
                        "properties": {
                            "type": {
                                "enum": [
                                    "VDataBlockType",
                                    "CVDataBlockType"
                                ]
                            },
                            "values": {
                                "type": "array",
                                "items": {
                                    "type": [
                                        "number",
                                        "string",
                                        "null",
                                        "boolean"
                                    ]
                                }
                            }
                        }
                    }
                }
```

```
            },
            {
                "required": [
                    "type",
                    "fileReference"
                ],
                "properties": {
                    "type": {
                        "enum": ["RangeSetRefType"]
                    },
                    "fileReference": {
                        "type": "array",
                        "items": {
                            "type": "string",
                            "format": "uri"
                        }
                    }
                }
            }
        ]
    }
}
```

The following JSON fragment is an example of a response to a Coverage RangeSet request.

Coverage RangeSet Example

```
{
  "TBD": [
    "filler1",
    "filler2"
  ]
}
```

7.6.5.3. Error situations

See HTTP status codes for general guidance.

7.6.6. Coverage Metadata

returns the coverage's metadata (may be empty)

7.6.6.1. Operation

Requirement 14	**/req/core/cov-md-op**

A	The API SHALL support the HTTP GET operation at the path `/collections/{coverageId}/coverage/metadata`. • `coverageId` is the local identifier for a `Coverage`. It serves the same role and is subject to the same requirements as the `collectionId` parameter defined in API-Common.

7.6.6.2. Response

Requirement 15	**/req/core/cov-md-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The content of that response SHALL contain metadata which describes the coverage as defined in the JSON schema coverage_metadata.json.
C	The response SHALL include links to the following resources using the specified link relations: • Resource 1: `reltype` • Resource 2: `reltype`

Coverage Metadata Response Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "metadata": {
        "description": "The metadata element is a container of any (not further
specified) information which should be transported along with the coverage on hand,
such as domain-specific metadata.",
        "type": "object",
        "properties": {}
    }
}
```

The following JSON fragment is an example of a response to a Coverage Metadata request.

Coverage Metadata Example

```
{
  "TBD": [
    "filler1",
    "filler2"
  ]
}
```

7.6.6.3. Error situations

See HTTP status codes for general guidance.

7.6.7. Coverage All

returns all of the above namely the coverage's domainset, rangetype, meatadata, and rangeset comparable to a GetCoverage response

7.6.7.1. Operation

Requirement 16	**/req/core/cov-all-op**
A	The API SHALL support the HTTP GET operation at the path `/collections/{coverageId}/coverage/all`. • `coverageId` is the local identifier for a `Coverage`. It serves the same role and is subject to the same requirements as the `collectionId` parameter defined in API-Common.

7.6.7.2. Response

Requirement 17	**/req/core/cov-all-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 include the `domainset`, `rangetype`, `metadata`, and `rangeset` of the coverage as defined in the JSON schema coverage.json.
C	The response SHALL include links to the following resources using the specified link relations: • Resource 1: `reltype` • Resource 2: `reltype`

Coverage Response Schema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Coverage object",
    "description": "Component of OGC Coverage Implementation Schema 1.1. Last updated:
2016-may-18. Copyright (c) 2016 Open Geospatial Consortium, Inc. All Rights Reserved.
To obtain additional rights of use, visit http://www.opengeospatial.org/legal/.",
    "type": "object",
    "oneOf": [
        {
            "required": [
                "type",
                "domainSet",
                "rangeSet",
                "rangeType"
            ],
            "properties": {
                "id": {"type": "string"},
                "type": {
                    "enum": ["CoverageByDomainAndRangeType"]
                },
                "envelope": {"$ref": "coverage_envelope.json#/envelope"},
                "domainSet": {"$ref": "coverage_domainset.json#/domainSet"},
                "rangeSet": {"$ref": "coverage_rangeset.json#/rangeSet"},
                "rangeType": {"$ref": "coverage_rangetype.json#/rangeType"},
                "metadata": {"$ref": "coverage_metadata.json#/metadata"}
            }
        },
        {
            "required": [
                "type",
                "partitionSet",
                "rangeType"
            ],
            "properties": {
                "id": {"type": "string"},
                "type": {
                    "enum": ["CoverageByPartitioningType"]
                },
                "envelope": {"$ref": "coverage_envelope.json#/envelope"},
                "partitionSet": {"$ref":
"coverage_partitioningset.json#/partitioningSet"},
                "rangeType": {"$ref": "coverage_rangetype.json#/rangeType"},
                "metadata": {"$ref": "coverage_metadata.json#/metadata"}
            }
        }
    ]
}
```

The following JSON fragment is an example of a response to a Coverage All request.

Coverage All Example

```
{
  "TBD": [
    "filler1",
    "filler2"
  ]
}
```

7.6.7.3. Error situations

See HTTP status codes for general guidance.

7.7. Coverage Subsetting

The third part is about query parameters:

- http://acme.com/oapi/collections/{collectionid}/coverage?SUBSET=Lat(40 — returns a coverage cutout between (40,10) and (50,20), as multipart coverage
- http://acme.com/oapi/collections/{collectionid}/coverage/rangeset?SUBSET=Lat(40 — returns a coverage cutout between (40,10) and (50,20), in the coverage's Native Format
- http://acme.com/oapi/collections/{collectionid}/coverage?SUBSET=time("2019-03-27") — returns a coverage slice at the timestamp given (in case the coverage is Lat/Long/time the result will be a 2D image)

7.8. Coverage Query Parameters

In this clause, the query component of OGC API - Coverages access is established.

Subsetting parameters may be mixed in a query part, in no particular order.

7.9. General

7.9.1. HTTP status codes

This API standard does not impose any restictions on which features of the HTTP and HTTPS protocols may be used. API clients should be prepared to handle any legal HTTP or HTTPS status code.

The **Status Codes** listed in Table 4 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 Table 4 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 9. Typical HTTP status codes

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

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

Permission 1	**/per/core/additional-status-codes**
A	Servers MAY support other capabilities of the HTTP protocol and, therefore, MAY return other status codes than those listed in Table 4, too.

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.

Chapter 8. Media Types for any data encoding(s)

A section describing the MIME-types to be used is mandatory for any standard involving data encodings. If no suitable MIME type exists in http://www.iana.org/assignments/media-types/index.html then this section may be used to define a new MIME type for registration with IANA.

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: Revision History

Date	Release	Editor	Primary clauses modified	Description
2019-03-06	Template	C. Heazel	all	initial template

Annex C: 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