OGC API - Coverages - Part 1

Core

Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: 2019-12-31

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/).

1.1. Service Scope

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.

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 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.

1.2. Content Scope

The API-Coverages standard provides access to content which complies with the Coverage Implementation Schema (CIS) 1.1 (http://docs.opengeospatial.org/is/09-146r6/09-146r6.html). However, there are limitations:

- 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.
- Coverage Partitioning is not supported. The `Coverage-Partitioning` Requirements Class will be added 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 identifies five 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-Coverages are:

- **Core**
- **Subset**
- **HTML**
- **JSON**
- **OpenAPI 3.0**

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.

The *Subset* Requirements Class provides capabilities to select a sub-set of a `Coverage` using a "bounding box" which is suitable for any coordinate reference system.

The *HTML* and *JSON* Requirements Classes address support for encodings commonly used with APIs.

The *OpenAPI 3.0* Requirements Class addresses the use of the OpenAPI 3.0 standard to document and communicate the API Definition.

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
- 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 Sub-clause 5.3 of OGC Web Services Common (OGC 06-121r9), 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. 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 1. 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

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-Coverages standard defines an API with two goals:

1. Provide access to `Coverages` conformant to the OGC CIS standard.
2. Provide functionality comparable to that of the OGC WCS 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.

Table 2 summarizes the access paths and relation types defined in this standard.

Table 2. Coverage 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 (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.

Path Template	Relation	Resource
{root}/collections/{coverageid}/coverage/description	none	returns the whole coverage description consisting of domainset, rangetype, and metadata (but not the rangeset)
{root}/collections/{coverageid}/coverage/domainset	none	returns the coverage's domain set definition
{root}/collections/{coverageid}/coverage/rangetype	none	returns the coverage's range type information (i.e., a description of the data semantics)
{root}/collections/{coverageid}/coverage/metadata	none	returns the coverage's metadata (may be empty)
{root}/collections/{coverageid}/coverage/rangeset	none	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	none	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)

6.2. Coverage Implementation Schema

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 statistical data.

This OGC API - Coverages standard 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.

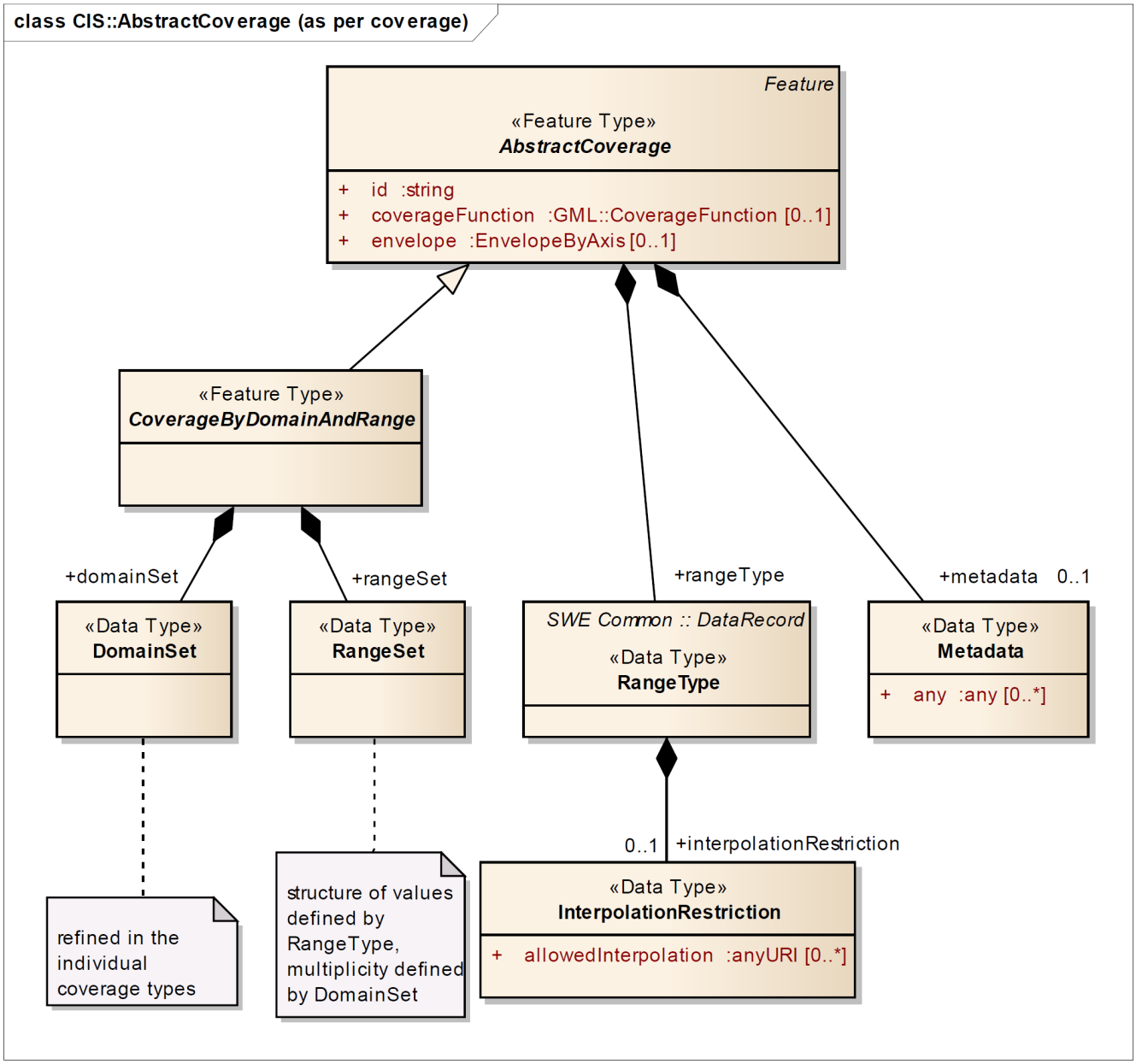


Figure 1. Abstract Coverage

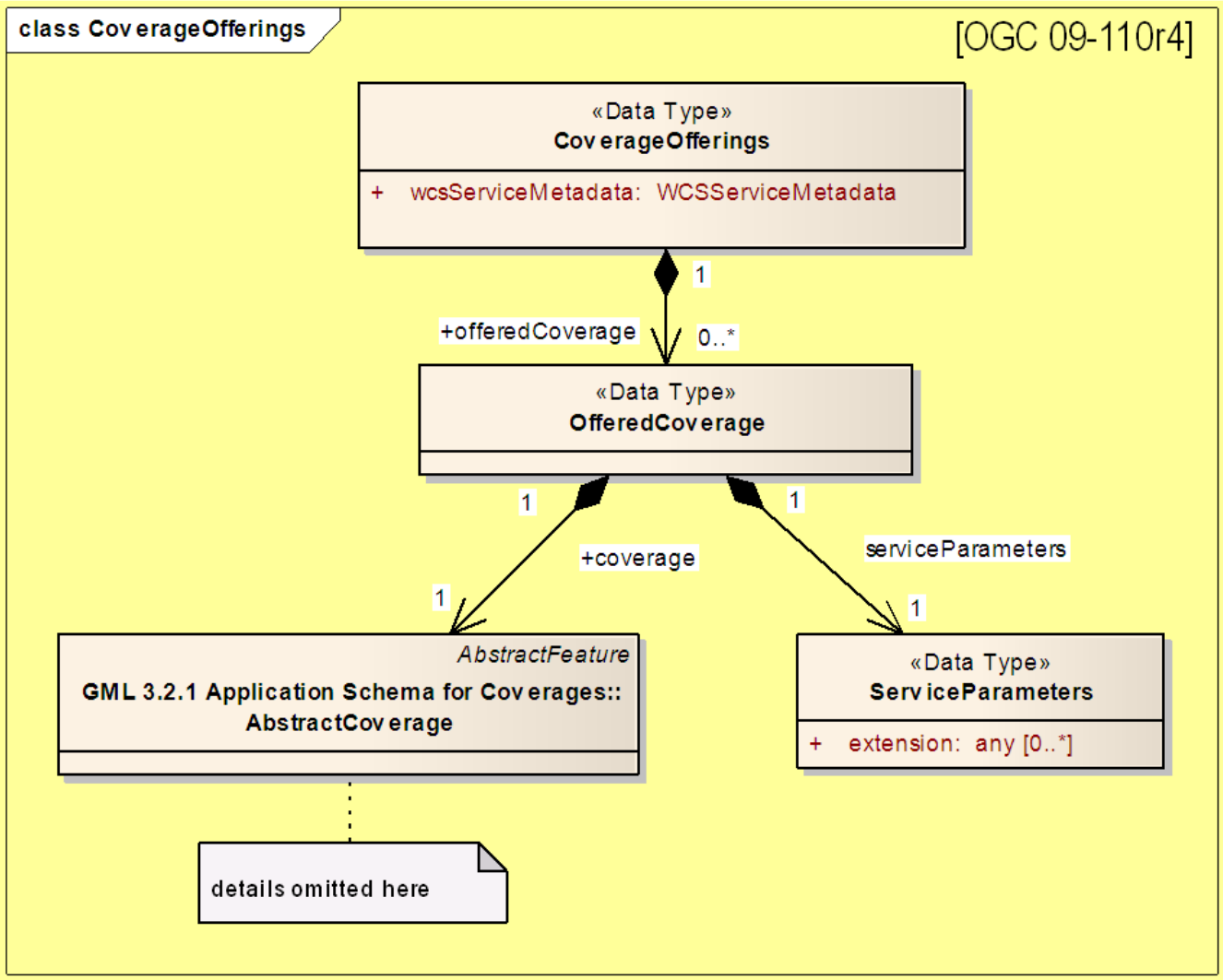
If you are unfamiliar with the term 'coverage', the explanations on the Coverages DWG Wiki provide more detail and links to educational material. Additionally, Coverages: describing properties that vary with location (and time) in the W3C/OGC Spatial Data on the Web Best Practice document may be considered.

6.3. API Behavior Model

The Coverages API is designed to be compatible but not conformant with the OGC Web Coverage Service. This allows API-Coverage and WCS implementations to co-exist in a single processing environment.

OGC Web Coverage Service standard version 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 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. 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. Table 3 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 3. 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-1/1.0/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-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 `Core` Requirements Class defines the requirements for locating, understanding, and accessing coverage resources. The `Core` Requirements Class is presented in five sections:

1. API Platform: a set of common capabilities
2. Collection Access: operations for accessing collections of `Coverages`
3. Coverage Access: operations for accessing `Coverage` resources
4. Parameters: parameters for use in the API-Coverage operations.
5. General: general principles for use with this standard.

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 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 Coverage 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 4. 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 `Coverage` 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-Coverages Landing Page operation.

Landing Page Example


```
{
  "title": "string",
  "description": "string",
  "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",
      "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 5. 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 `Coverage` 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 6. 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 `Coverage` 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-coverages-1/1.0/conf/core

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"
  ]
}
```

7.3.3.3. 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.

API-Coverages observes that a coverage is a collection of measured values. Therefore, a coverage is

a collection.

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

7.4.1. Collections

The `Collections` operation returns a set of metadata which describes the collections available from this API. Each collection on a Coverages API will be a coverage.

Table 7. 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 `Coverage` 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.

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"
  ]
}
```

7.4.1.3. 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, or in the the case of API-Coverages, a single Coverage. An abbreviated copy of this information is returned for each Coverage in the `/collections` response.

Table 8. 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 `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.

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

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

Support for both the `/collections/{coverageid}` 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 {coverageid} 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-Coverages Collection Information operation.

Collection Information Example

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

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. Coverage Access

In this clause, API-Common is extended to support `Coverage` resources.

A `Coverage` is a collection of measured values. The structure of that collection is defined by the CIS standard. CIS contains four principle components:

- A `DomainSet` component describing the coverage's domain (the set of "direct positions", i.e., the locations for which values are stored in the coverage)
- A `RangeType` component which describes the coverage's `RangeSet` data structure (in the case of images usually called the "pixel data type").
- A `RangeSet` component containing the stored values (often referred to as "pixels", "voxels") of the coverage.
- A `Metadata` component which represents an extensible slot for metadata. The intended use is to hold any kind of application-specific metadata structures.

Each component is directly accessible through the API using standard paths. In addition, collections of elements can be accessed through the following paths:

- `/description`: Returns `DomainSet`, `RangeType`, and `Metadata`
- `/all`: Returns `DomainSet`, `RangeType`, `RangeSet`, and `Metadata`

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

7.5.1. Coverage Offering

The `Coverage Offering` operation returns a general coverage offering description consisting of envelope, rangetype, and service metadata such as the coverage's native format

7.5.1.1. Operation

The `Coverage Offering` operation is defined by the following requirement.

Requirement 3	/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.

7.5.1.2. Response

A successful response to the `Coverage Offering` operation shall meet the following requirement.

Requirement 4	/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 be encoded using the format(s) negotiated through the HTTP protocol.
D	If no format is negotiated, then the response SHALL be encoded using the default format as described in Media Types.

Coverage Offering Response Schema

```
$schema: http://json-schema.org/draft-07/schema#
coverageOffering:
  title: Coverage Offering
  description: Description of the Coverage Offering
  required:
    - links
    - name
    - envelope
    - rangetype
    - servicemetadata
  type: object
  properties:
    name:
      type: string
      description: identifier of the collection used, for example, in URIs
      example: dem
    title:
      type: string
      description: human readable title of the collection
      example: Digital Elevation Model
    description:
      type: string
      description: a description of the data in the collection
      example: Digital Elevation Model
    links:
      type: array
      items:
        $ref: '#/components/schemas/link'
    envelope:
      $ref: '#/components/schemas/envelope'
    rangetype:
      $ref: '#/components/schemas/rangetype'
    servicemetadata:
      $ref: '#/components/schemas/serviceMetadata'
```

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

Coverage Offering Example

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

7.5.1.3. 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.

7.5.2. Coverage Description

The `Coverage Description` operation returns the whole coverage description consisting of domainset, rangetype, and metadata (but not the rangeset)

7.5.2.1. Operation

The `Coverage Description` operation is defined by the following requirement.

Requirement 5	**/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.

7.5.2.2. Response

A successful response to the `Coverage Description` operation shall meet the following requirement.

Requirement 6	**/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 be encoded using the format(s) negotiated through the HTTP protocol.
D	If no format is negotiated, then the response SHALL be encoded using the default format as described in Media Types.

Coverage Description Response Schema

```
$schema: http://json-schema.org/draft-07/schema#
coverageDescription:
  title: Coverage Description
  description: The whole coverage description including domain set, range type and
    metadata. It does not include the range set.
  type: object
  properties:
    domainSet:
      "$ref": coverage_domainset.json#/domainSet
    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 Description request.

Coverage Description Example

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

7.5.2.3. 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.

7.5.3. Coverage Domain Set

The `Coverage Domain Set` operation returns the coverage's domain set definition

7.5.3.1. Operation

The `Coverage Domain Set` operation is defined by the following requirement.

Requirement 7	**/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.

7.5.3.2. Response

A successful response to the `Coverage Domain Set` operation shall meet the following requirement.

Requirement 8	/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 be encoded using the format(s) negotiated through the HTTP protocol.
D	If no format is negotiated, then the response SHALL be encoded using the default format as described in Media Types.

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"
  ]
}
```

7.5.3.3. 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.

7.5.4. Coverage Range Type

The `Coverage Range Type` operation returns the coverage's range type information (i.e., a description of the data semantics)

7.5.4.1. Operation

The `Coverage Range Type` operation is defined by the following requirement.

Requirement 9	/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.

7.5.4.2. Response

A successful response to the `Coverage Range Type` operation shall meet the following requirement.

Requirement 10	/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 be encoded using the format(s) negotiated through the HTTP protocol.
D	If no format is negotiated, then the response SHALL be encoded using the default format as described in Media Types.

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: quantity
          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"
  ]
}
```

7.5.4.3. 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.

7.5.5. Coverage Range Set

The `Coverage Range Set` operation returns the coverage's range set, i.e., the actual values in the coverage's Native Format (see Media Types for ways to retrieve inspecific formats)

7.5.5.1. Operation

The `Coverage Range Set` operation is defined by the following requirement.

Requirement 11	**/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.

7.5.5.2. Response

A successful response to the `Coverage Range Set` operation shall meet the following requirement.

Requirement 12	**/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 be encoded using the format(s) negotiated through the HTTP protocol.
D	If no format is negotiated, then the response SHALL be encoded using the default format as described in Media Types.

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"
  ]
}
```

7.5.5.3. 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.

7.5.6. Coverage Metadata

The `Coverage Metadata` operation returns the coverage's metadata (may be empty)

7.5.6.1. Operation

The `Coverage Metadata` operation is defined by the following requirement.

Requirement 13	**/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.

7.5.6.2. Response

A successful response to the `Coverage Metadata` operation shall meet the following requirement.

Requirement 14	**/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 be encoded using the format(s) negotiated through the HTTP protocol.

D	If no format is negotiated, then the response SHALL be encoded using the default format as described in Media Types.

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"
  ]
}
```

7.5.6.3. 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.

7.5.7. Coverage All

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

7.5.7.1. Operation

The `Coverage All` operation is defined by the following requirement.

Requirement 15	**/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.

7.5.7.2. Response

A successful response to the `Coverage All` operation shall meet the following requirement.

Requirement 16	**/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 be encoded using the format(s) negotiated through the HTTP protocol.
D	If no format is negotiated, then the response SHALL be encoded using the default format(s) as described in Media Types.

Coverage Response Schema

```
$schema: http://json-schema.org/draft-04/schema#
title: Coverage object
description: 'Component of OGC Coverage Implementation Schema 1.1. Last updated: 2016-
may-18.
  Copyright (c) 2016 Open Geospatial Consortium, Inc. All Rights Reserved. To obtain
  additional rights of use, visit http://www.opengeospatial.org/legal/.'
type: object
required:
- type
- domainSet
- rangeSet
- rangeType
properties:
  id:
    type: string
  type:
    enum:
    - CoverageByDomainAndRangeType
  envelope:
    "$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
```

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

Coverage All Example

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

7.5.7.3. 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.

7.6. Parameters

The API-Coverages standard inherits basic query and subsetting parameters from API-Common. This section provides a short description of each parameter and identifies the relevant requirements.

All of the permissions and recommendations in API-Common regarding the these parameters also apply to API-Coverages implementations.

7.6.1. Parameter bbox

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

Requirement 17	**/req/core/cov-bbox-parameter**
A	A Coverage API SHALL support the Bounding Box (bbox) parameter for `/collections` and `/collections/{coverageid}` requests.
B	Requests which include the Bounding Box parameter SHALL comply with API-Common requirement `/req/core/rc-bbox-definition`.
C	Responses to Bounding Box requests SHALL comply with API-Common requirement `/req/core/rc-bbox-response`.

7.6.2. Parameter datetime

The Date-Time (datetime) parameter is defined in API-Common. The following requirement governs use of that parameter in a Coverage API.

Requirement 18	**/req/core/cov-datetime-parameter**
A	A Coverage API SHALL support the Date-Time (datetime) parameter for `/collections` and `/collections/{coverageid}` requests.
B	Requests which include the Date-Time parameter SHALL comply with API-Common requirement `/req/core/rc-time-definition`.
C	Responses to Date-Time requests SHALL comply with API-Common requirement `/req/core/rc-time-response`.

7.6.3. Parameter Limit

The Limit (limit) parameter is defined in API-Common. The following requirement governs use of that parameter in a Coverage API.

Requirement 19	**/req/core/cov-limit-parameter**
A	A Coverage API SHALL support the Limit (limit) parameter for `/collections` and `/collections/{coverageid}` requests.
B	Requests which include the Limit parameter SHALL comply with API-Common requirement `/req/core/rc-limit-definition`.
C	Responses to Limit requests SHALL comply with API-Common requirements: • `/req/core/rc-limit-response` • `/req/core/rc-numberReturned` • `/req/core/rc-numberMatched`

7.6.4. Combinations of Filter Parameters

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

7.6.5. 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 20	**/req/core/cov-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/fc-links` • `/req/core/fc-rel-type` • `/req/core/fc-timestamp` • `/req/core/fc-numberMatched` • `/req/core/fc-numberReturned`

7.7. General

7.7.1. HTTP Response

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

Requirement 21	**/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 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.

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 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.

Chapter 8. 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.

In addition to the Requirements Classes, there are additional coverage formats which implementors should be aware of. These formats apply to encodings of pixel data. Since this data is typically binary, it is largely opaque to the API.

8.1. 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.

8.2. 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

8.2.1. 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.

8.2.2. CIS JSON

This API-Coverages standard is built around the OGC Coverage Implementation Schema (CIS). CIS content often includes multi-dimensional coordinates and coordinate reference systems in sensor and analytic space. These "Engineering" coordinate reference systems cannot be represented using WGS-84. Therefore, an alternative to GeoJSON is needed.

The OGC JSON Schema for CIS standard addresses that need by defining a JSON schema for the CIS standard. This format should be used for all JSON encodings from the `{root}/collections/{coverageid}/coverage/*` paths.

8.3. Binary

A coverage does not need to be delivered in JSON or HTML in its entirety. Multipart encoding is also defined in OGC CIS. This allows the result to have a "canonical" header while components can be factored out and represented in some (more efficient) binary format. Any suitable container format (such as zip, miltipart/mime, SAFE, etc.) can "bundle" these components into one coverage file ready for shipping.

Commonly used formats for binary encoding are: * GeoTIFF * JPEG * JPEG-2000 * NetCDF

8.4. 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-Coverages standard in provided in Table 5.

Coverages can be encoded in any suitable data format, including formats as GML, JSON, GeoTIFF, and NetCDF. Further, coverages can be represented by a single document (stream or file) or by a hierarchically organized set of documents, each of which can be encoded individually – for example, the domain set, range type, and metadata may be encoded in easily parseable GML, JSON, or RDF while the range set is encoded in some compact binary format like NetCDF or JPEG2000. Such partitioning allows for coverages tiled in space, time, or mixed, thereby enabling mosaics, time-interleaved coverages, and efficiently subsettable datacubes.

Table 10. API-Coverages MIME Types

Encoding	MIME Type
HTML	text/html
JSON	application/json
GeoJSON	application/geo+json
GeoTIFF	image/tiff; application=geotiff
JPEG	image/jpeg
JPEG-2000	image/jp2
NetCDF	application/x-netcdf

NOTE Consider adding a table showing valid MIME-types for each CIS component.

8.5. 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 22	/req/core/cov-mediatype-default
A	The default media type for Range Set data SHALL be GeoTIFF.
B	IF the `JSON` Conformance Class is advertised, then the default media type for content other than a Range Set SHALL be JSON or GeoJSON.
C	IF the `JSON` Conformance Class is not advertised, then the default media type for content other than a Range Set SHALL be HTML.
D	The default media type for responses to `all` requests SHALL be multi-part using the appropriate media type for each component resource.

Chapter 9. Requirements Class Subset

The `core` Requirements Class provides parameters for filtering the results of an API-Coverages request.

The Subset Requirements Class defines parameters for filtering n-dimensional Range Sets. Subsetting parameters may be mixed with other parameters, in no particular order, in the query part of a URL.

Requirements Class	
http://www.opengis.net/spec/ogcapi-coverages-1/1.0/req/subset	
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi_coverage-1/1.0/req/core
Dependency	OGC Coverage Implementation Schema (CIS)

The subset parameter is defined in the following Requirement:

Requirement 23	**/req/subset/definition**

A	The operation SHALL support a parameter `subset` with the following characteristics (using an Backus Naur Form (BNF) fragment):
	`SubsetSpec: subset=axisName(intervalOrPoint)`
	`axisName: {NCName}`
	`intervalOrPoint: interval | point`
	`interval: low : high`
	`low: point | *`
	`high: point | *`
	`point: {number} | "{text}"`
	`Where:`
	`" = double quote = ASCII code 0x42,`
	`{NCName} is an XML-style identifier not containing ":" (colon) characters,`
	`{number} is an integer or floating-point number, and`
	`{text} is some general ASCII text (such as a time and date notation in ISO 8601).`

B	The axis name SHALL be the same as one of the `axisLabels` defined in the DomainSet or else return a 400 status code.

C	The intervalOrPoint values SHALL fall within the range of valid values defined by the DomainSet for the identified axis or else return a 400 status code

The results of using a subset parameter are defined by the following Requirement:

Requirement 24	**/req/subset/subset-success**
A	The `subset` parameter SHALL only apply to resources offered as coverages.
B	Only that part of the coverage addressed SHALL be returned that falls within the bounds of the subset expression. The DomainSet shall be adjusted accordingly to the new boundaries (in case of trimming) and the reduced dimension (in case of slicing).

9.1. Subsetting Examples

- http://acme.com/oapi/collections/{coverageid}/coverage/rangeset?SUBSET=Lat(40,50)&SUBSET=Long(10,20) — returns a coverage cutout between (40,10) and (50,20), in the coverage's Native Format
- http://acme.com/oapi/collections/{coverageid}/coverage/rangeset?SUBSET=time("2019-03-27") — returns a coverage slice at the timestamp given, with all data available in the other dimensions (in case the coverage is Lat/Long/time the result will be a 2D image with its full Lat/Lon extent)
- http://acme.com/oapi/collections/{coverageid}/coverage?SUBSET=Lat((40,50)&SUBSET=Long(10,20) — returns a cutout from the coverage identified with extent between corner coordinates (40,10) and (50,20) (note: replace the _ in the URL by a comma character - an adoc issue)
- http://acme.com/oapi/collections/{coverageid}/coverage/rangeset?SUBSET=Lat(40,50)&SUBSET=Long(10,20) — returns the range set of a coverage cutout between (40,10) and (50,20), in the coverage's Native Format; no domain set, range type, and metadata will be returned (note: replace the _ in the URL by a comma character - an adoc issue)
- http://acme.com/oapi/collections/{coverageid}/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)

Chapter 10. Requirements Class HTML

The following requirements apply to an OGC API-Coverage 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-coverages-1/1.0/req/html	
Target type	Web API
Dependency	Conformance Class "Core"
Dependency	API-Common HTML
Dependency	HTML5
Dependency	Schema.org

10.1. 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/{coverageid}`: Collection Information

Requirement 25	**/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 26	/req/html/conformance
The list of Conformance Classes advertised by the API SHALL include:	
A	http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/html

10.2. Coverage Offering

This section covers the Coverage Offering response for the `{root}/collections/{coverageid}/coverage` operation.

Requirement 27	/req/html/cov-offer-success
Restricts	/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 response SHALL be a vaild HTML document
C	The response SHALL include content equivalent to that defined in JSON schema coverage_offering.json.

10.3. Coverage Description

This section covers the Coverage Description response for the `{root}/collections/{coverageid}/coverage/description` operation.

Requirement 28	/req/html/cov-desc-success
Restricts	/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 response SHALL be a vaild HTML document
C	The response SHALL include content equivalent to that defined in JSON schema coverage_description.json.

10.4. Coverage Domain Set

This section covers the Coverage Domain Set response for the

`{root}/collections/{coverageid}/coverage/domainset` operation.

Requirement 29	**/req/html/cov-ds-success**
Restricts	/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 response SHALL be a vaild HTML document
C	The response SHALL include content equivalent to that defined in JSON schema coverage_domainset.json.

10.5. Coverage Range Type

This section covers the Coverage Range Type response for the `{root}/collections/{coverageid}/coverage/rangetype` operation.

Requirement 30	**/req/html/cov-rt-success**
Restricts	/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 response SHALL be a vaild HTML document
C	The response SHALL include content equivalent to that defined in JSON schema coverage_rangetype.json.

10.6. Coverage Range Set

This section covers the Coverage Range Set response for the `{root}/collections/{coverageid}/coverage/rangeset` operation.

Requirement 31	**/req/html/cov-rs-success**
Restricts	/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 response SHALL be a vaild HTML document
C	The response SHALL include content equivalent to that defined in JSON schema coverage_rangeset.json.

10.7. Coverage Metadata

This section covers the Coverage Metadata response for the `{root}/collections/{coverageid}/coverage/metadata` operation.

Requirement 32	**/req/html/cov-md-success**
Restricts	/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 response SHALL be a vaild HTML document
C	The response SHALL include content equivalent to that defined in JSON schema coverage_metadata.json.

10.8. Coverage All

This section covers the Coverage All response for the `{root}/collections/{coverageid}/coverage/all` operation.

Requirement 33	**/req/html/cov-all-success**
Restricts	/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 response SHALL be a vaild HTML document
C	The response SHALL include content equivalent to that defined in JSON schema coverage.json.

Chapter 11. Requirements Class JSON

The following requirements apply to an OGC API-Coverage 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_coverages-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/{coverageid}`: Collection Information

Requirement 34	**/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 35	/req/json/conformance
The list of Conformance Classes advertised by the API SHALL include:	
A	http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/json

11.2. Coverage Offering

This section covers the Coverage Offering response for the `{root}/collections/{coverageid}/coverage` operation.

Requirement 36	/req/json/cov-offer-success
Restricts	/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 response SHALL be a JSON or GeoJSON document which validates against the JSON schema coverage_offering.json.

11.3. Coverage Description

This section covers the Coverage Description response for the `{root}/collections/{coverageid}/coverage/description` operation.

Requirement 37	/req/json/cov-desc-success
Restricts	/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 response SHALL be a JSON or GeoJSON document which validates against the JSON schema coverage_description.json.

11.4. Coverage Domain Set

This section covers the Coverage Domain Set response for the `{root}/collections/{coverageid}/coverage/domainset` operation.

Requirement 38	/req/json/cov-ds-success

Restricts	/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 response SHALL be a JSON or GeoJSON document which validates against the JSON schema coverage_domainset.json.

11.5. Coverage Range Type

This section covers the Coverage Range Type response for the `{root}/collections/{coverageid}/coverage/rangetype` operation.

Requirement 39	**/req/json/cov-rt-success**
Restricts	/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 response SHALL be a JSON or GeoJSON document which validates against the JSON schema coverage_rangetype.json.

11.6. Coverage Range Set

This section covers the Coverage Range Set response for the `{root}/collections/{coverageid}/coverage/rangeset` operation.

Requirement 40	**/req/json/cov-rs-success**
Restricts	/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 response SHALL be a JSON or GeoJSON document which validates against the JSON schema coverage_rangeset.json.

11.7. Coverage Metadata

This section covers the Coverage Metadata response for the `{root}/collections/{coverageid}/coverage/metadata` operation.

Requirement 41	/req/json/cov-md-success
Restricts	/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 response SHALL be a JSON or GeoJSON document which validates against the JSON schema coverage_metadata.json.

11.8. Coverage All

This section covers the Coverage All response for the `{root}/collections/{coverageid}/coverage/all` operation.

Requirement 42	/req/json/cov-all-success
Restricts	/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 response SHALL be a JSON or GeoJSON document which validates against the JSON schema coverage.json.

Chapter 12. Requirements class "OpenAPI 3.0"

Requirements Class	
http://www.opengis.net/spec/ogcapi-coverages/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-Coverages as well. So an implementation of API-Coverages which supports OpenAPI 3.0 as an API Description format must also comply with the API-Common oas30 Conformance Class.

Requirement 43	**/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 44	**/req/oas30/conformance**
The list of Conformance Classes advertised by the API SHALL include:	
A	http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/oas30

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