

OGC API - Coverages - Part 1

Core

Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: 2020-08-10

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: 0.0.6

Category: OGC® Implementation Specification

Editor: Charles Heazel

OGC API - Coverages - Part 1: Core

Copyright notice

Copyright © 2019, 2020 Open Geospatial Consortium

To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>

Warning

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

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

Document type: OGC® Implementation
Specification

Document stage: Draft

Document language: English

License Agreement

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

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

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

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

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

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

Table of Contents

1. Scope	7
1.1. Service Scope	7
1.2. Content Scope	7
2. Conformance	8
3. References	9
4. Terms and Definitions	11
4.1. Coverage	11
4.2. Regular grid	11
4.3. Irregular grid	11
4.4. Displaced grid	11
4.5. Mesh	11
4.6. Partition [of a coverage]	11
4.7. Sensor model	11
4.8. Transformation grid	11
5. Conventions	12
5.1. Identifiers	12
5.2. Examples	12
5.3. Schema	12
5.4. UML Notation	12
5.5. Namespace Prefix Conventions	13
6. Overview	14
6.1. General	14
6.2. Coverage Implementation Schema	16
6.3. API Behavior Model	17
6.4. Dependencies	18
7. Requirements Class "GeoData-Coverage"	19
7.1. Overview	19
7.2. Dependencies	19
7.3. Platform	19
7.3.1. API landing page	20
7.3.2. API definition	22
7.3.3. Declaration of conformance classes	23
7.4. Collection Access	25
7.4.1. Collections	25
7.4.2. Collection Information	26
7.5. Coverage Access	29
7.5.1. Coverage	30
7.6. Parameters	33

7.6.1. Coverage Domain Set	33
7.6.2. Coverage Range Type	40
7.6.3. Coverage Range Set	44
7.6.4. Coverage Metadata	46
7.6.5. Parameter bbox	48
7.6.6. Parameter datetime	48
7.6.7. Parameter Limit	49
7.6.8. Combinations of Filter Parameters	49
7.6.9. Paged Response	49
7.7. General	50
7.7.1. HTTP Response	50
7.7.2. HTTP status codes	51
8. Media Types	53
8.1. HTML Encoding	53
8.2. CIS JSON Encoding	53
8.3. Binary	54
8.4. Media Types	54
8.5. Default Encodings	55
9. Requirements Class Coverage Subset	56
9.1. Subsetting Examples	57
10. Requirements Class HTML	58
10.1. Common	58
10.2. Coverage	59
10.3. Coverage Domain Set	59
10.4. Coverage Range Type	59
10.5. Coverage Range Set	60
10.6. Coverage Metadata	60
11. Requirements Class CIS JSON	62
11.1. Common	62
11.2. Coverage	63
11.3. Coverage Domain Set	63
11.4. Coverage Range Type	63
11.5. Coverage Range Set	64
11.6. Coverage Metadata	64
12. Requirements class "OpenAPI 3.0"	65
Annex A: Conformance Class Abstract Test Suite (Normative)	66
A.1. Conformance Class A	66
A.1.1. Requirement 1	66
A.1.2. Requirement 2	66
Annex B: Revision History	67
Annex C: Glossary	68

Annex D: Bibliography	70
-----------------------------	----

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 or multi-point coverages are addressed, not Curve/Surface/SolidCoverages. The reason is that these are common use cases that receive the most attention today. Gridded coverages or 'point-cloud' multi-point coverages are the primary data types considered but other kinds of datasets are not excluded.
- 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 multi-dimensional "bounding box" which is suitable for any coordinate reference system and any dimension.

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](#)
- Nottingham, M.: IETF RFC 8288, **Web Linking**, [RFC 8288](#)
- OGC 19-072: **OGC API - Common - Part 1: Core**, (Draft) https://github.com/opengeospatial/oapi_common/blob/Master/19-072.pdf
- OGC 20-024: **OGC API - Common - Part 2: Geospatial Data**, (Draft) https://github.com/opengeospatial/oapi_common/blob/Master/20-024.pdf
- 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](#)
- 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 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
- IETF: RFC 7159, **The JavaScript Object Notation (JSON) Data Interchange Format**, <https://www.ietf.org/rfc/rfc7159.txt>, 2014
- W3C: W3C JSON-LD 1.0, **A JSON-based Serialization for Linked Data**. <http://www.w3.org/TR/json-ld/>, 2014
- W3C: W3C JSON-LD 1.0 **Processing Algorithms and API**. <http://www.w3.org/TR/json-ld-api>, 2014
- W3C: W3C RDF 1.1 **Concepts and Abstract Syntax**. <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>, 2014

Chapter 4. Terms and Definitions

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

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

4.1. 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:

- Dataset distribution API: The endpoint corresponding to a dataset distribution, where the landing page resource as defined in OGC API - Common - Part 1: Core is available (subsequently referred to as Base URI or `{datasetAPI}`)
- Access Paths: Unique paths to Resources
- Query: Parameters to adjust the representation of a Resource or Resources like encoding format or subsetting

Access Paths are used to build resource identifiers. It is recommended, but not required, that these paths

Most 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 Resources

Resource URI	Relation Type	Description
{datasetAPI}/	<code>ogc:common:dataset</code>	Landing page for this dataset distribution
{datasetAPI}/api	<code>service-desc</code>	API description (e.g. OpenAPI)
{datasetAPI}/api	<code>service-doc</code>	API documentation (optional, e.g. HTML)
{datasetAPI}/conformance	<code>conformance</code>	Conformance Classes
{datasetAPI}/collections	<code>data</code>	The list off all collections available, some or all of which may be accessible using this Coverage API. Each of these collection objects take the same form as that of the collection resource object described immediately below.

Resource URI	Relation Type	Description
{datasetAPI}/collections/{coverageId}	ogc:common:collection	resource corresponding to the collection with the unique identifier {coverageId} , which may be accessible as a coverage. The resource will describe key elements such as an id , title , description , available crs and extent (the coverage envelope) as well as links to resources pertaining to this collection. For coverages, it will either embed or link to a CIS JSON encoding of both the range type and the domain set. It is comparable to a WCS DescribeCoverage response, with the exception that the range type and domain set may have to be retrieved separately by following a link to accommodate the case where they may be considerably large, and the domain set may support query parameters to subset it.
Coverages		
{datasetAPI}/collections/{coverageId}/coverage	ogc:coverage:coverage	returns the coverage including all of its components (domain set, range type, range set and metadata), to the extent that it is supported by the selected representation (see format encoding for ways to retrieve in specific formats). It is comparable to a WCS GetCoverage response.
{datasetAPI}/collections/{coverageId}/coverage/rangeset	ogc:coverage:rangeset	if supported by the service and by the selected representation, returns only the coverage's range set, i.e., the actual values in the selected representation without any accompanying description or extra information.
{datasetAPI}/collections/{coverageId}/coverage/rangetype	ogc:coverage:rangetype	if available separately from the collection resource, returns the coverage's range type information, i.e., a description of the data semantics (their components and data type).
{datasetAPI}/collections/{coverageId}/coverage/domainset	ogc:coverage:domainset	if available separately from the collection resource, returns the coverage's domain set definition (the detailed n-dimensional space covered by the data).
{datasetAPI}/collections/{coverageId}/coverage/metadata	ogc:coverage:metadata	if available, returns the associated coverage metadata as defined by the CIS model, which may be e.g. domain specific metadata.

Where:

- {datasetAPI} = URI of the landing page for the API distributing the dataset

- {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](#).

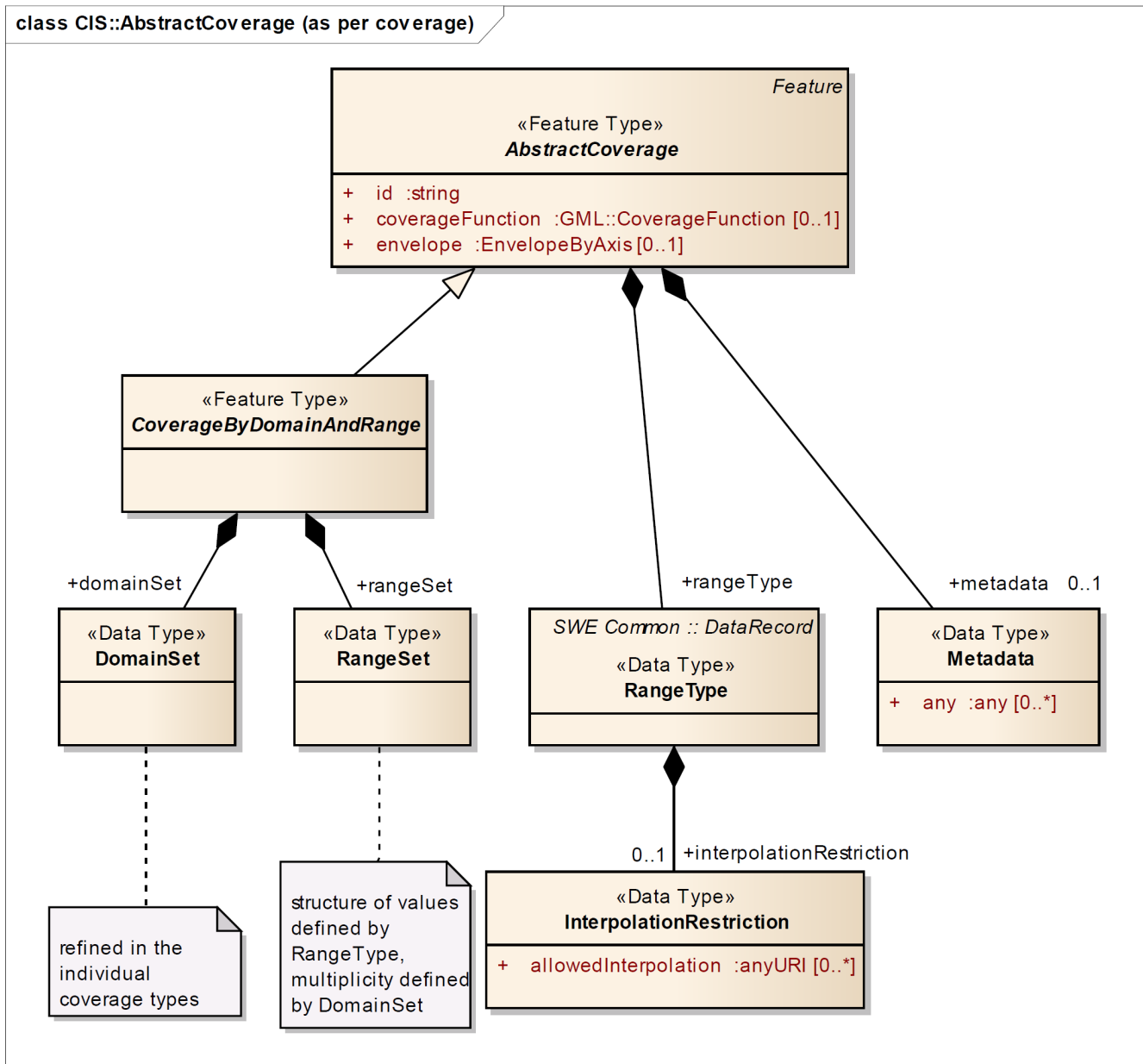


Figure 1. Abstract Coverage

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
Collection Access	http://www.opengis.net/spec/ogcapi_common-2/1.0/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/json
HTML	http://www.opengis.net/spec/ogcapi_common-1/1.0/req/html

Chapter 7. Requirements Class "GeoData-Coverage"

7.1. Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-coverages-1/1.0/req/geodata-coverage	
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-2/1.0/req/collections

The **GeoData-Coverage** Requirements Class defines the requirements for locating, understanding, and accessing a geospatial data resource as a coverage. The **GeoData-Coverage** 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 - Part 1: Core and Part 2: Geospatial Data Standards. 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-2/1.0/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 purpose of the landing page is to provide clients with a starting point for using the API. It provides the basic information needed to understand the purpose, structure, and capabilities of the API. The landing page also serves as the starting point for navigation. Any resource exposed through an API can be accessed using a path or link starting from the landing page.

The resources exposed by API-Common Core include:

- A description of the API,
- A declaration of conformance classes,
- A list of coverages available.

Those resources 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 **{datasetAPI}/** path

Support for **GET** on the **{datasetAPI}/** 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.

```
{
  "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": "The list of available collections",
      "rel": "data"
    }
  ]
}
```

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 **{datasetAPI}/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 **{datasetAPI}/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/geodata-coverage/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-2/1.0/conf/geodata
C	http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/geodata-coverage

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/json",
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/html",
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/oas3",
    "http://www.opengis.net/spec/ogcapi-common-2/1.0/conf/geodata",
    "http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/geodata-coverage",
    "http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/cisjson",
    "http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/cisrdf",
    "http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/netcdf",
    "http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/geotiff",
    "http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/html",
    "http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/geodata-subset",
    "http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/geodata-bbox",
    "http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/geodata-datetime"
  ]
}

```

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. The coverages returned in the response can be limited using the **Bounding Box**, **Date-Time**, and **Limit** parameters.

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 **{datasetAPI}/collections** path
2. Follow the **data** link on the landing page

Support for both the **{datasetAPI}/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 through the API. To keep the size of this response manageable, the summary metadata should be limited to just those elements which are essential for clients and those which are needed to support the filtering parameters.

Recommendation 1	/rec/geodata-coverage/coverageinfo
A	<p>The coverage metadata provided through the /collections resource SHOULD be limited to:</p> <ul style="list-style-type: none">• TBD• TBD• TBD

```
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.

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

1. Issue a **GET** request on the `{datasetAPI}/collections/{coverageid}` path
2. Follow the `ogc:common:collection` link on the ?????

NOTE | What is the origin of the "ogc:common:collection" link?

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 the `/collections/{coverageid}` path is recommended by API-Common. Support for the `ogc:common:collection` link is required.

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. This metadata is an extension to the set defined by API-Common. The additional elements are specific to Coverage resources.

```
type: object
required:
  - id
  - links
  - extent
  - domainSet
  - rangeType
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
  domainSet:
    $ref: coverage_domainset.json#/domainSet
  rangeType:
    $ref: coverage_rangetype.json#/rangeType
  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 Resource Example

```
{
  "id": "CIS123456789",
  "title": "Bonn Germany",
  "description": "Image over the city of Bonn.",
  "extent": {
    "spatial": [ 7.01, 50.63, 7.22, 50.78 ],
    "temporal": [ "2010-02-15T12:34:56Z", "2018-03-18T12:11:00Z" ]
  },
  "itemType": "coverage",
  "CRS": [
    "http://www.opengis.net/def/crs/OGC/1.3/CRS84",
    "http://www.opengis.net/def/crs/EPSSG/0/4326"
  ],
  "domainSet": { "$href":
"http://data.example.org/collections/CIS123456789/coverages/domainSet"},
  "rangeType": { "$href":
"http://data.example.org/collections/CIS123456789/coverages/rangeType"},
  "links": [
    { "href": "http://data.example.org/collections/CIS123456789/coverages",
      "rel": "items", "type": "application/json",
      "title": "CIS Image" }
  ]
}
```

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.

A coverage containing the **DomainSet**, **RangeType**, **RangeSet**, and **Metadata** components is accessible at

- The relative path **/coverage**
- The link relation **ogc:coverage:coverage**

The **DomainSet** and **RangeType** are either defined as properties of the **/collections/{coverageid}** resource, or linked using specific relation types to their own resource.

The **RangeSet** may be available as its own resource as well, if supported by the selected representation (format).

If **Metadata** is available, it will also be available as its own resource.

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

7.5.1. Coverage

The **Coverage** operation returns all the components of the coverage (rangeset, domain set, range type, metadata). It is comparable to a WCS GetCoverage operation.

7.5.1.1. Operation

The **Coverage** operation is defined by the following requirement.

Requirement 3	/req/geodata-coverage/coverage-op
A	<p>The API SHALL support the HTTP GET operation at the path /collections/{coverageid}/coverage.</p> <ul style="list-style-type: none">• 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** operation shall meet the following requirement.

Requirement 4	/req/geodata-coverage/coverage-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 rangeset of the coverage equivalent to that defined in the JSON schema coverage.json .

C	The content of that response SHALL also include the domainset and rangetype as well, to the extent that the negotiated format can describe it.
D	The content of that response MAY also include the metadata for that coverage. The server SHOULD return that metadata if it is available and the negotiated format can describe it.
E	The response SHALL be encoded using the format(s) negotiated through the HTTP protocol.
F	If no format is negotiated, then the response SHALL be encoded using the format associated with the media type described in the link object which links to this resource, contained within the coverage (collection) resource.


```
$schema: http://json-schema.org/draft-07/schema#
title: Coverage object
description: 'Component of OGC Coverage Implementation Schema 1.1. Last updated: 2016-
may-18.
  Copyright (c) 2016 Open Geospatial Consortium, Inc. All Rights Reserved. To obtain
  additional rights of use, visit http://www.opengeospatial.org/legal/.'
type: object
oneOf:
- required:
  - type
  - domainSet
  - rangeSet
  - rangeType
properties:
  id:
    type: string
  type:
    enum:
    - CoverageByDomainAndRangeType
  envelope:
    "$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_partitionset.json#/partitionSet
  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 request.

Coverage 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](#).

7.6. 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 these parameters also apply to API-Coverages implementations.

7.6.1. Coverage Domain Set

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

7.6.1.1. Operation

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

Requirement 5	/req/geodata-coverage/domainset-op
A	<p>If the API does not include a "DomainSet" property within the coverage (collection) resource, it The API SHALL support the HTTP GET operation at the path /collections/{coverageid}/coverage/domainset.</p> <ul style="list-style-type: none">• 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

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

Requirement 6	/req/geodata-coverage/domainset-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 equivalent to that 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 CIS JSON, which SHALL also be the format associated with the media type described in the link object which links to this resource, contained within the coverage (collection) resource.

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"
  ]
}

```

7.6.1.3. 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](#).

7.6.2. Coverage Range Type

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

7.6.2.1. Operation

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

Requirement 7	/req/geodata-coverage/rangetype-op
A	<p>If the API does not include a "RangeType" property within the coverage (collection) resource, it SHALL support the HTTP GET operation at the path /collections/{coverageid}/coverage/rangetype.</p> <ul style="list-style-type: none"> • 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

A successful response to the [Coverage Range Type](#) operation shall meet the following requirement.

Requirement 8	/req/geodata-coverage/rangetype-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 equivalent to that 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 CIS JSON, which SHALL also be the format associated with the media type described in the link object which links to this resource, contained within the coverage (collection) resource.

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"
  ]
}

```

7.6.2.3. 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](#).

7.6.3. 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)

7.6.3.1. Operation

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

Requirement 9	/req/geodata_coverage/rangeset-op
A	The API MAY support the HTTP GET operation at the path /collections/{coverageid}/coverage/rangeset .
B	<p>The API SHOULD support it for selected formats which can describe the raw data values without any additional information.</p> <ul style="list-style-type: none">• 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

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

Requirement 10	/req/geodata_coverage/rangeset-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 equivalent to that 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 format associated with the media type described in the link object which links to this resource, contained within the coverage (collection) resource.

```

$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
              - CVDaataBlockType
          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"
  ]
}
```

7.6.3.3. 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](#).

7.6.4. Coverage Metadata

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

7.6.4.1. Operation

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

Requirement 11	/req/geodata-coverage/metadata-op
A	The API MAY support the HTTP GET operation at the path /collections/{coverageid}/coverage/metadata .
B	<p>The API SHOULD support it, if available, for selected formats which can describe it.</p> <ul style="list-style-type: none">• 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

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

Requirement 12	/req/geodata-coverage/metadata-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 equivalent to that 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 format associated with the media type described in the link object which links to this resource, contained within the coverage (collection) resource.

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"
  ]
}
```

7.6.4.3. 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](#).

TODO: Move **geodata-filteredList** conformance class into its own clause, which may end up only in OGC API - Common - Part 2: Geospatial data has Coverage doesn't add anything.

TODO: Create new clauses for conformance classes for **coverage-bbox** and **coverage-dateTime** (or potentially make this a single one), which are additional ways to sub-select the actual coverage data being returned that a service may opt to support.

TODO: Limit and Paged response may apply to both listing collections ([geodata-filteredList](#)) and the coverage itself. Is it a requirement to support Limit for either of these? Is it separate conformance classes? It would be difficult for static servers, and paging makes less sense for coverages than it does for many small features.

7.6.5. 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 13	/req/geodata-filteredList/bbox-parameter
A	An API implementing Common - Part 2: Geospatial data MAY support the Bounding Box (bbox) parameter for /collections requests.
B	Requests which include the Bounding Box parameter SHALL comply with API-Common requirement /req/core/rc-bbox-definition .
C	Responses SHALL be appropriately filtered, returning only collections whose spatial extent intersects with the specified bounding box parameter.

7.6.6. 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 14	/req/geodata-filteredList/datetime-parameter
A	An API implementing Common - Part 2: Geospatial data MAY support the Date-Time (datetime) parameter for /collections requests.
B	Requests which include the Date-Time parameter SHALL comply with API-Common requirement /req/geodata-filteredList/rc-time-definition .
C	Responses SHALL be appropriately filtered, returning only collections whose temporal extent intersects with the specified date time parameter.

7.6.7. Parameter Limit

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

Requirement 15	/req/geodata-filteredList/limit-parameter
A	A Coverage API SHALL support the Limit (limit) parameter for /collections 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: <ul style="list-style-type: none">• /req/core/rc-limit-response• /req/core/rc-numberReturned• /req/core/rc-numberMatched

7.6.8. 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.'

7.6.9. 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 16	/req/geodata-coverage/paged-response
----------------	--------------------------------------

A	<p>Responses to a filtered operation that only return a portion of the full selected resource set SHALL comply with API-Common requirements:</p> <p>TODO: These Features resources pertained to individual features being returned, and are not currently defined in Common. Their paging mechanism likely does not apply well to coverages as well.</p> <ul style="list-style-type: none"> • <code>/req/core/rc-response</code> • <code>/req/core/fc-links</code> • <code>/req/core/fc-rel-type</code> • <code>/req/core/fc-timestamp</code> • <code>/req/core/fc-numberMatched</code> • <code>/req/core/fc-numberReturned</code>
---	--

7.7. General

7.7.1. HTTP Response

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

Requirement 17	<code>/req/core/http-response</code>
A	An HTTP operation SHALL return a response which includes a <code>status code</code> and an optional <code>description</code> elements.
B	If the <code>status code</code> is not equal to 200, then the <code>description</code> 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/geospatial-data/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:

- [CIS 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. CIS JSON Encoding

Support for CIS 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>

When coverage resources (the coverage as a whole, range set, domain set, range type, meta data) advertise the `application/json` media type, it refers to the CIS JSON encoding.

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.

The [OGC JSON Schema for CIS](#) standard addresses that need by defining a JSON schema for the CIS standard. This format should be used when the `application/json` media type is specified to encode all of the `{datasetAPI}/collections/{coverageid}/coverage*` resources. A CIS JSON representation of the range type and domain set for the coverage is also required to be either embedded in properties or linked from the `/collections/{collectionId}` resource.

Other encodings such as CoverageJSON must use an alternate media type.

8.3. Binary

A coverage can also be distributed in binary form, for which a number of formats are commonly used:

- CIS RDF
- NetCDF
- GeoTIFF
- PNG
- JPEG
- JPEG-2000

OGC CIS also defines multipart encoding of the different components of a coverage. 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, multipart/mime, SAFE, etc.) can "bundle" these components into one coverage file ready for shipping.

With OGC API - Coverages, given that the description of the coverage is easily available at the /collections/{collectionId} resource, with the range type and domain set either embedded or linked from that resource, an implementation may prefer to keep single-part binary formats encoding all components to the best it can, as they might be more directly interoperable.

8.4. Media Types

A description of the media types is mandatory for any OGC standard which involves data encodings. The list of suitable media types for the API-Coverages standard is provided in [Table 5](#).

Coverages can be encoded in any suitable data format, including formats as GML, CIS 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, CIS 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 subtable datacubes.

Table 10. API-Coverages media Types

Encoding	media type
HTML	text/html
JSON / CIS JSON	application/json
CIS RDF	TODO:
CoverageJSON	TODO:
GeoTIFF	image/tiff; application=geotiff
PNG	image/png

JPEG	image/jpeg
JPEG-2000	image/jp2
NetCDF	application/x-netcdf

NOTE | Consider adding a table showing valid media 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. If not using content negotiation, the encoding must follow the media type described in the link to the resource from the collection.

Chapter 9. Requirements Class Coverage Subset

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/coverage-subset	
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi_coverage-1/1.0/req/geodata-coverage
Dependency	OGC Coverage Implementation Schema (CIS)

The subset parameter is defined in the following Requirement:

Requirement 18	/req/coverage-subset/definition
A	<p>The operation SHALL support a parameter subset with the following characteristics (using an Backus Naur Form (BNF) fragment):</p> <div><pre>SubsetSpec: subset=axisName(intervalOrPoint) axisName: {NCName} intervalOrPoint: interval point interval: low : high low: point * high: point * point: {number} "{text}"</pre><p>Where:</p><ul style="list-style-type: none">\ " = 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).</div>
B	<p>The axis name SHALL be the same as one of the axisLabels defined in the DomainSet or else return a 400 status code.</p>

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 19	/req/coverage-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 whereby a "*" (asterisk) in the position of a lower or upper limit along an axis denotes the coverage's minimum or maximum extent along this axis, respectively. 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\(10,20\)](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\("2019-03-27"\)](http://acme.com/oapi/collections/{coverageid}/coverage/rangeset?subset=time() — 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\(10,20\)](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\(10,20\)](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\("2019-03-27"\)](http://acme.com/oapi/collections/{coverageid}/coverage?subset=time() — 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 **GeoData-Coverage** 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 **GeoData-Coverage** 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:

- **{datasetAPI}/**: Dataset API Landing Page
- **{datasetAPI}/api**: API Description
- **{datasetAPI}/conformance**: Conformance Classes
- **{datasetAPI}/collections**: Collections
- **{datasetAPI}/collections/{coverageid}**: Collection Information

Requirement 20	/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 21	/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

This section covers the [Coverage](#) response for the `{datasetAPI}/collections/{coverageid}/coverage` operation.

Requirement 22	/req/html/coverage-success
Restricts	/req/core/coverage-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 valid HTML document
C	The response SHALL include content equivalent to that defined in JSON schema coverage.json .

10.3. Coverage Domain Set

This section covers the [Coverage Domain Set](#) response for the `{datasetAPI}/collections/{coverageid}/coverage/domainset` operation.

Requirement 23	/req/html/domainset-success
Restricts	/req/core/domainset-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 valid HTML document
C	The response SHALL include content equivalent to that defined in JSON schema coverage_domainset.json .

10.4. Coverage Range Type

This section covers the [Coverage Range Type](#) response for the

`{datasetAPI}/collections/{coverageid}/coverage/rangetype` operation.

Requirement 24	/req/html/rangetype-success
Restricts	/req/core/rangetype-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 valid HTML document
C	The response SHALL include content equivalent to that defined in JSON schema coverage_rangetype.json .

10.5. Coverage Range Set

This section covers the [Coverage Range Set](#) response for the `{datasetAPI}/collections/{coverageid}/coverage/rangeset` operation.

Requirement 25	/req/html/rangeset-success
Restricts	/req/core/rangeset-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 valid HTML document
C	The response SHALL include content equivalent to that defined in JSON schema coverage_rangeset.json .

10.6. Coverage Metadata

This section covers the [Coverage Metadata](#) response for the `{datasetAPI}/collections/{coverageid}/coverage/metadata` operation.

Requirement 26	/req/html/metadata-success
Restricts	/req/core/metadata-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 valid HTML document
C	The response SHALL include content equivalent to that defined in JSON schema coverage_metadata.json .

Chapter 11. Requirements Class CIS JSON

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

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

The CIS JSON Requirements Class restricts requirements defined in the **GeoData-Coverage** 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 **GeoData-Coverage** requirements which lay within each scope.

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

11.1. Common

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

- **{datasetAPI}/**: Dataset API Landing Page
- **{datasetAPI}/api**: API Description
- **{datasetAPI}/conformance**: Conformance Classes
- **{datasetAPI}/collections**: Collections
- **{datasetAPI}/collections/{coverageid}**: Collection Information

Requirement 27	/req/cisjson/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/json

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

Requirement 28	/req/cisjson/conformance
The list of Conformance Classes advertised by the API SHALL include:	
A	http://www.opengis.net/spec/ogcapi-coverages-1/1.0/conf/cisjson

11.2. Coverage

This section covers the [Coverage](#) response for the `{datasetAPI}/collections/{coverageid}/coverage` operation.

Requirement 29	/req/cisjson/coverage-success
Restricts	/req/core/coverage-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 CIS JSON document which validates against the JSON schema coverage.json .

11.3. Coverage Domain Set

This section covers the [Coverage Domain Set](#) response for the `{datasetAPI}/collections/{coverageid}/coverage/domainset` operation.

Requirement 30	/req/cisjson/domainset-success
Restricts	/req/core/domainset-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 CIS JSON document which validates against the JSON schema coverage_domainset.json .

11.4. Coverage Range Type

This section covers the [Coverage Range Type](#) response for the `{datasetAPI}/collections/{coverageid}/coverage/rangetype` operation.

Requirement 31	/req/cisjson/rangetype-success
-----------------------	---------------------------------------

Restricts	/req/core/rangetype-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 CIS JSON document which validates against the JSON schema coverage_rangetype.json .

11.5. Coverage Range Set

This section covers the [Coverage Range Set](#) response for the [{datasetAPI}/collections/{coverageid}/coverage/rangeset](#) operation.

Requirement 32	/req/cisjson/rangeset-success
Restricts	/req/core/rangeset-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 CIS JSON document which validates against the JSON schema coverage_rangeset.json .

11.6. Coverage Metadata

This section covers the [Coverage Metadata](#) response for the [{datasetAPI}/collections/{coverageid}/coverage/metadata](#) operation.

Requirement 33	/req/cisjson/metadata-success
Restricts	/req/core/metadata-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 document which validates against the JSON schema coverage_metadata.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 34	/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 35	/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: Glossary

- **Conformance Test Module**

set of related tests, all within a single conformance test class (OGC 08-131r3)

NOTE:	When no ambiguity is possible, the word test may be omitted. i.e. conformance test module is the same as conformance module . Conformance modules may be nested in a hierarchical way. This term and those associated to it are included here for consistency with ISO 19105.
--------------	---

- **Conformance Test Class; Conformance Test Level**

set of **conformance test modules** that must be applied to receive a single **certificate of conformance**. (OGC 08-131r3)

NOTE:	When no ambiguity is possible, the word test may be left out, so conformance test class may be called a conformance class .
--------------	--

- **Executable Test Suite (ETS)**

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

- **Recommendation**

expression in the content of a document conveying that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required, or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited (OGC 08-131r3)

NOTE:	"Although using normative language, a recommendation is not a requirement . The usual form replaces the shall (imperative or command) of a requirement with a should (suggestive or conditional)." (ISO Directives Part 2)
--------------	---

- **Requirement**

expression in the content of a document conveying criteria to be fulfilled if compliance with the document is to be claimed and from which no deviation is permitted (OGC 08-131r3)

- **Requirements Class**

aggregate of all requirement modules that must all be satisfied to satisfy a conformance test class (OGC 08-131r3)

- **Requirements Module**

aggregate of **requirements** and **recommendations** of a specification against a single **standardization target** type (OGC 08-131r3)

- **Standardization Target**

entity to which some requirements of a standard apply (OGC 08-131r3)

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

Annex D: 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>
- 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>
- W3C, **RDF 1.1 Semantics**, February 2014, <https://www.w3.org/TR/rdf11-mt/>
- OGC: **OGC 13-102r2, Name type specification – Time and index coordinate reference system definitions** (OGC Policy Document), version 1, 2014