OGC® DOCUMENT: 22-029

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

Open Geospatial Consortium

OGC API - 3D GEOVOLUMES

STANDARD
Implementation

DRAFT

Version: 1.0.0
Submission Date: 2029-12-31
Approval Date: 2029-12-31
Publication Date: 2029-12-31
Editor: Jeff Harrison, Ignacio Correas, Jerome Jacovella-St-Louis

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF RECOMMENDATIONS

I ABSTRACT

This document provides an Application Programming Interface (API) and encoding that organizes access to a variety of 2D / 3D content according to a hierarchy of 3D geospatial volumes (GeoVolumes). The goal of this specification is to establish an API and encoding that allows applications to request a variety of 3D content from different providers in an interoperable and standardized way.

The API described in this document is based on Open Geospatial Consortium (OGC) and OpenAPI principles. The API is consistent with OGC API — Common core building blocks and supports link-follow, bounding box and tile coordinate query methods of access to resources of interest.

An OGC API — 3D GeoVolumes specification is needed because a variety of solutions and standards exist to access and transfer 3D geospatial content (e.g. 3D Tiles, I3S, glTF and others). The OGC API — 3D GeoVolumes specification addresses this challenge by providing a resource model and corresponding API to integrate various approaches to accessing and transferring 2D / 3D geospatial content into a single, open standards-based solution. By outlining an implementation approach for the parameters of the API, the 3D GeoVolumes specification simplifies application development across geospatial enterprises by providing a single interface for requesting and receiving a variety of 2D / 3D datasets and their distribution. As such, the goal of this OGC API specification is not to replace existing distribution methods and models for 3D content, but to enable interoperability between them.

The content provided within this document is derived largely from work done during collaborative, hands-on engineering conducted by members of the OGC during late 2019 and early 2020. As additional methodologies mature, this OGC API will be updated to include those approaches.

Table 1 — Overview of Resources

RESOURCE	PATH	HTTP METHOD	DOCUMENT REFERENCE
Landing page	`/`	GET	API Landing Page
API definition	`/api`	GET	API Definition
Conformance classes	`/conformance`	GET	Declaration of Conformance Classes
Collections metadata	`/collections`	GET	Collections Metadata
Collection instance metadata	`/collections/{collection_id}`	GET	Collection Metadata

NOTE: In the context of this candidate Standard, `{collection_id}` is the `{3DContainerId}` identifier of the 3D Container.

The resources identified in Table 1 primarily support Discovery operations. Discovery operations allow clients to interrogate the API to determine its capabilities and obtain information (metadata) about a distribution of a resource. This includes the API definition of the server(s) as well as metadata about the resources provided by those servers.

This standard extends the common query operations listed in Table 1 by defining simple, coordinate-based, queries which are applicable to many spatio-temporal, including geospatial, resource types. Other OGC API standards may define additional query capabilities specific to their resource type. EDR Query operations allow resources or values to be retrieved from the underlying spatio-temporal resource data store. The information returned is based upon the selection criteria (query string) provided by the client.

II KEYWORDS

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

ogcdoc, OGC document, OGC API, 3D Tiles , I3S, geographic information, Geospatial API, GeoVolume, GeoVolumes, 3D, feature, geographic information, dataset, distribution, API, OpenAPI, HTML, glTF, bounding volume hierarchy

III PREFACE

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

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

IV SECURITY CONSIDERATIONS

No security considerations have been made for this document.

V SUBMITTING ORGANIZATIONS

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

- US Army Geospatial Center (AGC)
- Skymantics, LLC
- Ecere Corporation
- Cesium
- Cognitics
- Helyx SIS
- Strategic Alliance Consulting Inc.
- Steinbeis

VI SUBMITTERS

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

Name	Affiliation
Jeff Harrison *(editor)*	US Army Geospatial Center
Ignacio Correas *(editor)*	Skymantics, LLC
Jerome Jacovella-St-Louis *(editor)*	Ecere Corporation
Tom Boggess	Strategic Alliance Consulting Inc.
Volker Coors	Steinbeis
Ryan Gauthier	Army Geospatial Center
Anneley Hadland	Helyx
Michala Hill	Cognitics

Thomas Myers	Strategic Alliance Consulting Inc.
Amy Youmans	Army Geospatial Center (AGC)
Insert POC Here	Cesium

1 SCOPE

1 SCOPE

The OGC 3D GeoVolumes specification provides an Application Programming Interface (API) and encoding that organizes access to a variety of 3D content according to a hierarchy of 3D geospatial volumes (GeoVolumes).

The goal of this OGC API is to allow applications to request a variety of 3D content from different providers in an interoperable and standardized way.

OGC API — 3D GeoVolumes specification is based on Open Geospatial Consortium (OGC) and OpenAPI principles. The API is consistent with OGC API – Common – Part 1: Core building blocks and supports link-follow, bounding box query, and tile coordinate methods of access to resources of interest.

The objective of the 3D GeoVolumes API and resource model is not to replace existing distribution methods and models for 3D content such as OGC 3D Tiles and I3S, but to enable interoperability between them.

The content provided within this document is derived largely from work done during collaborative, hands-on engineering conducted by members of the OGC during late 2019 and early 2020.

1.1. Why 3D GeoVolumes

OGC API — 3D GeoVolumes is needed because a variety of solutions and standards exist to access and transfer 3D geospatial content (e.g. 3D Tiles, I3S, glTF and others). OGC API — 3D GeoVolumes addresses the challenge of accessing and transferring 3D geospatial content in a variety of standards by providing a resource model and single API for requesting, receiving, and distributing this content, thus simplifying application development across geospatial enterprises. The 3D GeoVolumes API and resource model use a space-centric perspective to allow efficient access to 3D content.

This OGC API is not intended to replace existing distribution methods and models for 3D content, but to enable interoperability between them.

A high-level representation of a geospatial enterprise implementing 3D GeoVolumes based on the needs of various types of entities and/or systems (‘A’), (‘B’), and (‘C’) is illustrated in the figure below. The demonstration scenario involves siting a field hospital in Central Park, New York City during a global pandemic.

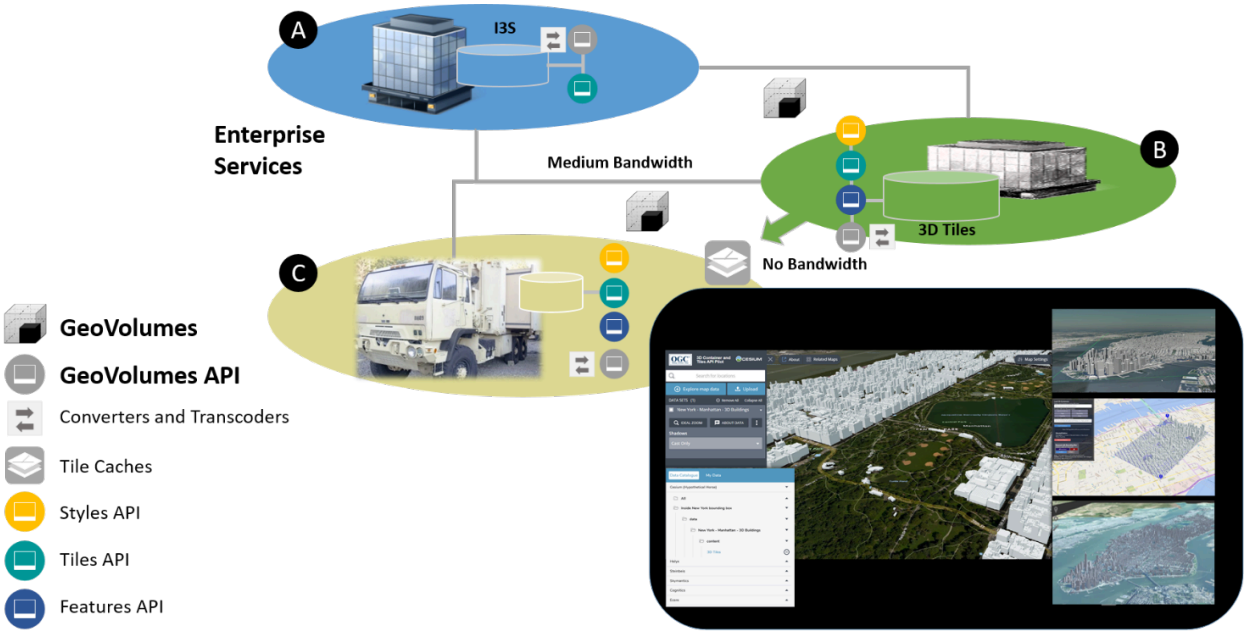


Figure 1 — The GeoVolumes API allows access to a variety of 3D content from different providers in a standardized way

The three systems differ in the 3D content standards implemented, volume of data that can be stored and processed, supported analytics and available bandwidth for data transport between entities, with 'A' having the highest bandwidth and 'C' having the lowest bandwidth.

Despite these differences, 3D GeoVolumes provides a model that allows offering, discovering, requesting and accessing data at each entity using a common API on top of a single organizational model of 3D geospatial resources. This common API leverages available 3D geospatial data formats and distribution standards such as 3D Tiles [https://www.ogc.org/standards/3DTiles], I3S [https://www.ogc.org/standards/i3s], CityGML [https://www.ogc.org/standards/citygml], and CDB [https://www.ogc.org/standards/cdb] to ensure users can work with 3D geospatial content in the optimal distribution format and interaction method for their application task. High bandwidth capacity data centers ('A') have 3D content available for broad regions or on a global scale. Content can be made available in multiple datasets and distributions based on conversion and transcoding workflows. Access to the data can be offered to other entities in the enterprise by means of the 3D GeoVolumes API. Depending on users' needs, data can be made available through alternative API methods in 2D or in raw format, such as the draft OGC API — Tiles specification or the OGC API — Features Standard.

Medium bandwidth capacity data centers ('B') do not require all data that is available at the data center ('A') but are more selective according to their role. The amount of data transferred to ('B') depends on the available bandwidth and specific needs for data analysis and re-distribution. To obtain required data in the best suited format and minimum size, medium bandwidth capacity data centers make use of the specific space-centric indexing scheme that is the fundamental idea of a GeoVolume resource and the corresponding 3D GeoVolumes API offered by ('A').

The high-level architecture defines a third enterprise entity, low bandwidth capacity field operations ('C'). These are connected at various bandwidths including intermittently connected or completely offline situations. In these cases, offline data packaging mechanisms and data

volume are operational considerations and optimized data selection and transmission processes are essential. Customers at this level want to go back and forth between 3D geospatial content distributions optimized for visualization via low bandwidth connections and attribute-loaded data that provides detailed information about selected elements in each view.

1.2. What are 3D GeoVolumes

The previous section provided a high-level representation of a geospatial enterprise implementation of the 3D GeoVolumes API based on the needs of various types of entities and/or systems. The technology described in this API supports this scenario efficiently because 3D GeoVolumes follow a common conceptual organization of space applied by humans, which is a collection of spaces where the spaces contain either sub-spaces or a set of objects. This representation of space is called the Bounding Volume, which is a closed volume containing the union of a set of geometric objects. The figure below illustrates typical Bounding Volumes (Box, Region, Sphere) with enclosed cuboid objects.

Figure 2 — Bounding Volumes (Box, Region, Sphere) with enclosed cuboid objects

The space organized in this manner may describe a collection of disjoint GeoVolumes, hierarchical collections of GeoVolumes or GeoVolumes accessed by tile coordinates organized in an OGC tiling scheme. The concepts of disjoint and hierarchical collections of GeoVolumes are illustrated in the figure below where the GeoVolume “North America” contains two child GeoVolumes “Montreal” and “New York City”. Both are spatially disjoint. The GeoVolume “New York City” contains a single 3D Dataset representing buildings. These buildings are available in multiple distribution formats (3D Tiles, I3S and CityGML).

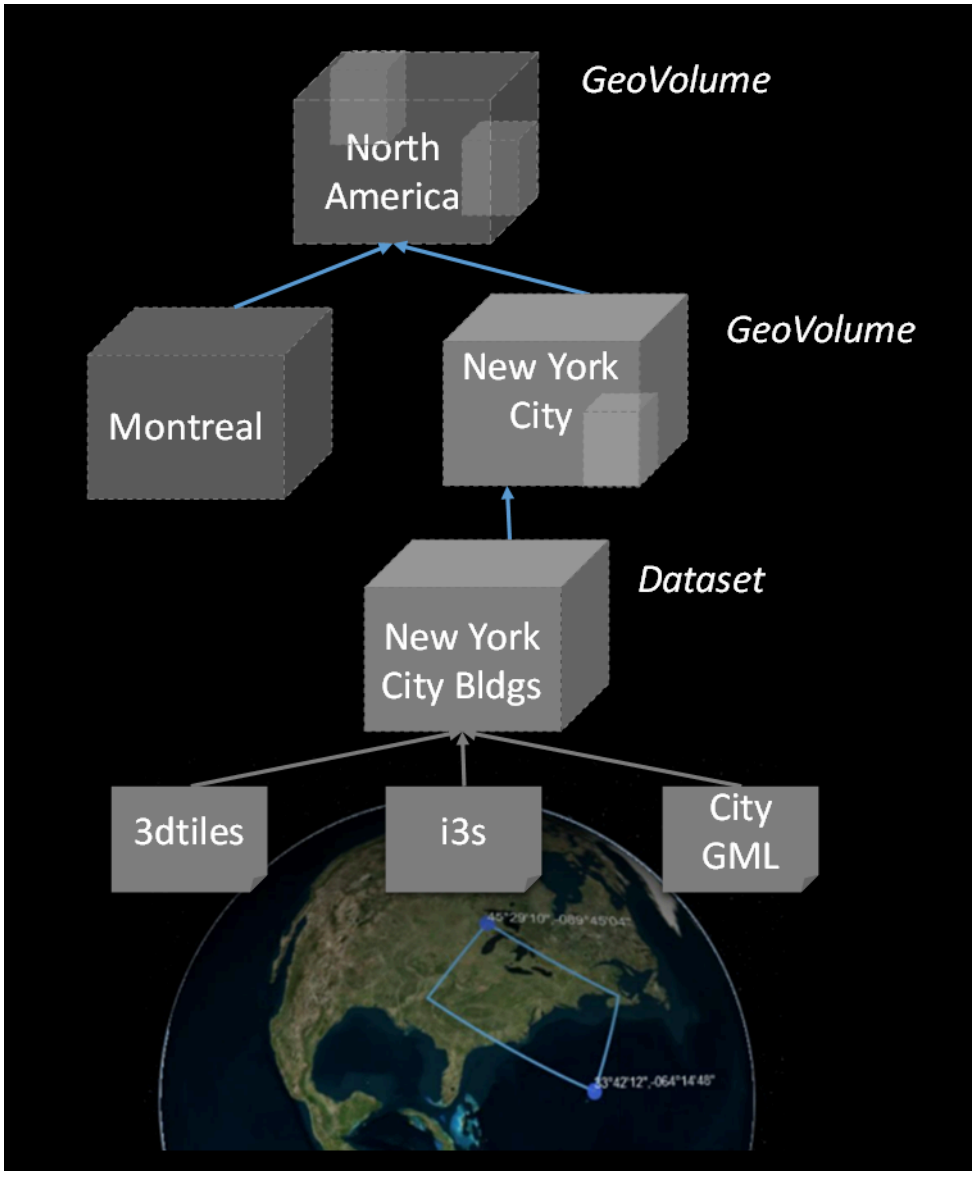


Figure 3 — GeoVolumes in nested hierarchy with a 3D dataset and multiple distributions

In these constructs, each GeoVolume may have one or more children whose extents may themselves overlap but in aggregate are completely contained in the parent volume extent. Each GeoVolume can contain references to and descriptions of the extent of dataset(s) of its contents and may provide links to multiple distributions of that dataset in different formats or encodings, e.g. 3D Tiles, I3S. GeoVolumes in this model may be accessed as /collections by an API. This API standard does not identify mandatory requirements for how specific distribution formats or encodings are composed and organized into spatial data structures (e.g. tiling schemes).

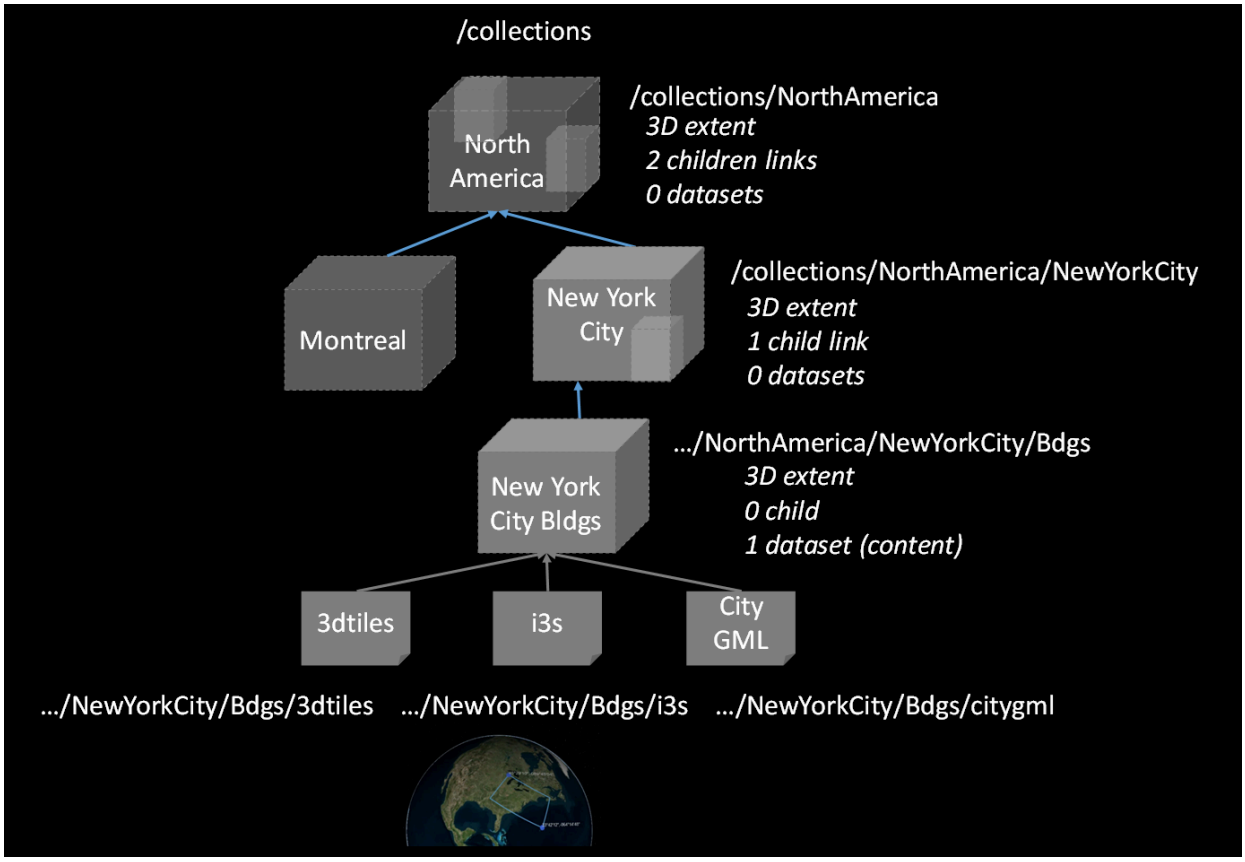


Figure 4 — GeoVolumes can reference extent of datasets and link to 'child' GeoVolumes

A GeoVolumes API may optionally access 3D geospatial content as tiles through extensions for tile coordinates. This may be achieved by using an extraDimensions property consisting of a list of identified objects, each with a description on how the additional dimensions are tiled. It may also be achieved by using Implicit Tiling as outlined in the emerging Community Standard for "3D Tiles Next". The two approaches will be harmonized in the GeoVolumes SWG and this specification updated.

If an extraDimensions property consisting of a list of identified objects is used this information can be added to either a TileMatrixSet's TileMatrix or to a TileSet's TileMatrixSetLimit. Including this property in the TileSet'sTileMatrixSetLimits allows for the re-use of common 2D TileMatrixSets for 3D geospatial content.

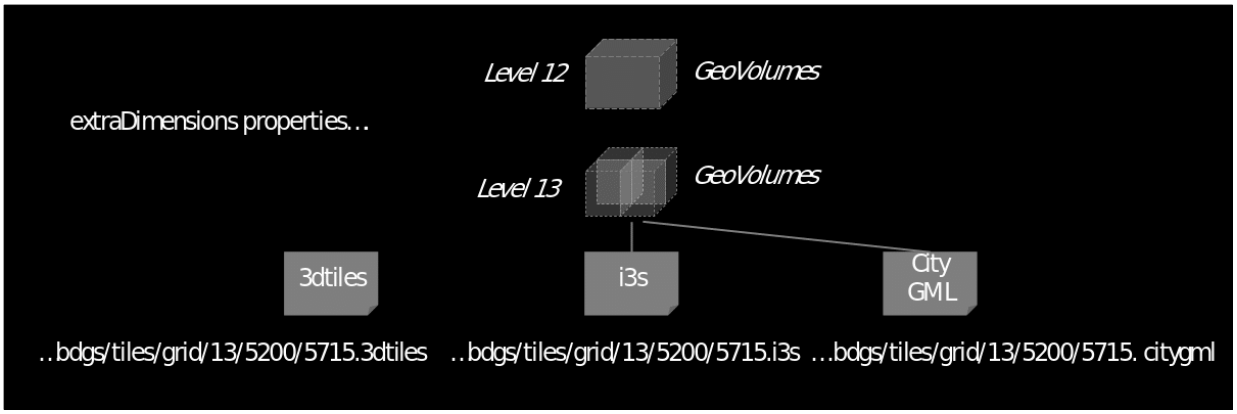


Figure 5 — GeoVolumes may optionally access 3D geospatial content via tile coordinates

The default representations of a GeoVolume are json/html information documents that define the bounding box/volume, link to an implicit tileset scheme if applicable, and provide links to the actual content. GeoVolumes are organized in collections as described above.

The basic architecture of server components to access 3D GeoVolume encodings is shown in the figure below. Each 3D / Globe client component can access 3D datasets in multiple distribution formats by means of components implementing the GeoVolumes API as an access interface. The API enables access to these resources using the HTTP protocol and its associated methods.

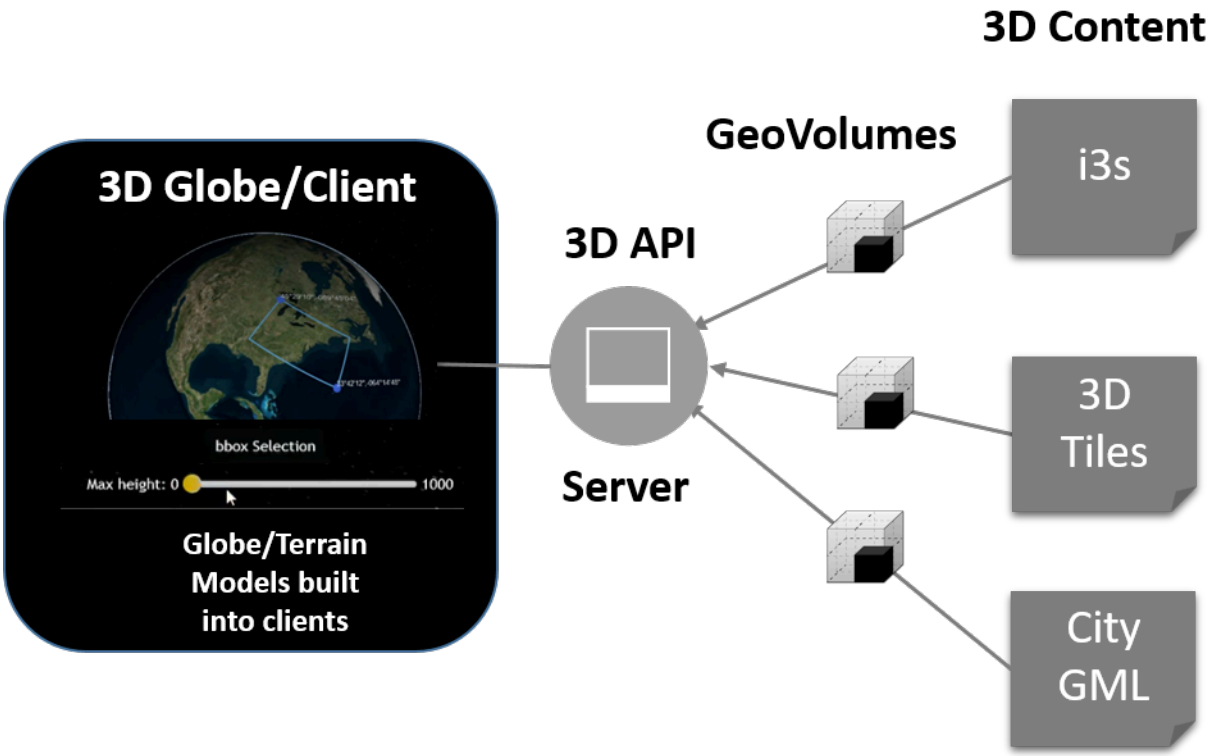


Figure 6 — Basic architecture of server components to access 3D GeoVolumes

Clients then visualize the returned content in the context of a 3D globe rendering either built into the client or assembled onto 2D tiles fetched separately from the GeoVolumes API,

using for example, a 2D Tile Server. In the future, 2D tiles could also be accessed through the GeoVolumes API.

2

CONFORMANCE

2 CONFORMANCE

Conformance with this API shall be checked using the tests specified in Annex A 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 - Part 1: Core defines an API module intended for re-use by other OGC Web API standards. The OGC API - 3D GeoVolumes is an extension of OGC API - Common - Part 1: Core. Conformance to the OGC API - 3D GeoVolumes requires demonstrated conformance to the applicable Conformance Classes of OGC API - Common.

OGC API - 3D GeoVolumes identifies a set of Conformance Classes. The Conformance Classes implemented by an API are advertised through the /conformance path on the landing page.

Each Conformance Class is defined by one or more Requirements Classes defined in Section 6. The requirements in Section 6 are organized by Requirements Class.

2.1. Mandatory Requirements Classes

The mandatory requirements class of OGC API - 3D GeoVolumes is the Requirements Class "OGC API – 3D GeoVolumes – Core" specified in Section 6.

This requirements class inherits from the *Core Requirements Class* of OGC API — Common — Part 1: Core which specifies the minimal useful service interface for an OGC API.

The requirements specified in the Requirements Class "OGC API – 3D GeoVolumes – Core" focus on minimal capabilities for a static web server.

The requirements specified in the Requirements Class "OGC API – 3D GeoVolumes – Core" are mandatory for all implementations of the 3D GeoVolumes API.

2.2. Optional Requirements Classes

The optional requirements classes of OGC API - 3D GeoVolumes include Requirements Class "OGC API – 3D GeoVolumes – Extension" specified in Section 6.

The requirements specified in the Requirements Class "OGC API – 3D GeoVolumes – Extension" adds the option for a spatial query parameter or tile coordinates to limit the result.

The requirements specified in the Requirements Class “OGC API – 3D GeoVolumes – Extension” are optional for implementations of the 3D GeoVolumes API.

3

NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

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

Bray, T.: IETF RFC 8259, The JavaScript Object Notation (JSON) Data Interchange Format, https://tools.ietf.org/rfc/rfc8259.txt

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

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: IETF RFC 2616, HTTP/1.1, https://tools.ietf.org/rfc/rfc2616.txt

Charles Heazel: OGC 19-072, *OGC API — Common — Part 1: Core*. Open Geospatial Consortium (2023). https://docs.ogc.org/is/19-072/19-072.html.

Heazel, C.: OGC API — Common — Part 2: Geospatial Data (Draft). OGC 20-024, Open Geospatial Consortium, http://docs.ogc.org/DRAFTS/20-024.html

Benjamin Hagedorn, Simon Thum, Thorsten Reitz, Voker Coors, Ralf Gutbell: OGC 15-001r4, *OGC® 3D Portrayal Service 1.0*. Open Geospatial Consortium (2017). https://docs.ogc.org/is/15-001r4/15-001r4.html.

Carl Reed, Tamrat Belayneh: OGC 17-014r7, *OGC Indexed 3d Scene Layer (I3S) and Scene Layer Package Format Specification*. Open Geospatial Consortium (2020). https://docs.ogc.org/cs/17-014r7/17-014r7.html.

Clemens Portele, Panagiotis (Peter) A. Vretanos, Charles Heazel: OGC 17-069r3, *OGC API — Features — Part 1: Core*. Open Geospatial Consortium (2019). https://docs.ogc.org/is/17-069r3/17-069r3.html.

Patrick Cozzi, Sean Lilley, Gabby Getz: OGC 18-053r2, *OGC 3D Tiles Specification 1.0*. Open Geospatial Consortium (2019). https://docs.ogc.org/cs/18-053r2/18-053r2.html.

OGC® Two Dimensional Tile Matrix Set and Tile Set Metadata Standard http://www.ogc.org/standards/requests/240.html

4

TERMS AND DEFINITIONS

4 TERMS AND DEFINITIONS

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

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

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

4.1. bounding volume

typically, a shape like a sphere, rectangular box, or convex hull that can simply be tested for intersection or overlap.

4.2. dataset

collection of data. (source: ISO 19168-1:2020)

4.3. distribution

specific representation of a dataset. (source: ISO 19168-1:2020)

4.4. feature

abstraction of real-world phenomena. (source: ISO 19101-1:2014)

4.5. geovolume

a hierarchy of geospatial bounding volumes.

4.6. static web server

a computer with an HTTP server containing hosted files that are sent ‘as-is’ to an application.

4.7. Web API

API using an architectural style that is founded on the technologies of the Web. [derived from the W3C Data on the Web Best Practices]

5

ACRONYMS

5 ACRONYMS

The acronyms relevant to this document are specified in the following list.

API	API Application Programming Interface
BBOX	Bounding Box
BVH	Bounding Volume Hierarchy
CDB	Common Database
COTS	Commercial Off The Shelf
CRS	Coordinate Reference System
glTF	GL Transmission Format
HTTP	Hypertext TransferProtocol
JSON	JavaScript Object Notations
OGC	Open GeospatialConsortium
SWG	Standards Working Group

6

CONVENTIONS

6 CONVENTIONS

This section provides details and examples for conventions used in this document.

6.1. Identifiers

The normative provisions in this specification are denoted by the URI

http://www.opengis.net/spec/{standard}/{m.n}

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

7

REQUIREMENTS CLASSES

7 REQUIREMENTS CLASSES

This section outlines requirements for an implementation of the 3D GeoVolumes API.

In addition to these basic requirements, the following requirements/conformance classes apply:

- **Core: Minimal GeoVolumes API capabilities (static web server)**
- **Extension: Adds the option for a spatial query parameter or tile coordinate to limit the result set.**

The 3D GeoVolumes API is designed to be a specialized API based on the OGC API principles that can significantly reduce the burden of implementing new clients and APIs as the specification only requires a minimal set of content/parameters needed to transfer 3D data, features, and attributes through an API.

This 3D GeoVolumes API follows the standards and conventions provided in the OGC API — Common for Web APIs. The 3D GeoVolumes API conforms to the OGC API — Common foundation resources: landing page, API definition, conformance, and collections (spatial resources).

The 3D GeoVolumes APIsupports the resources and operations listed in Table 1 with the associated conformance class and the link to the document section that specifies the requirements.

Table 2 — GeoVolumes API — Overview of resources and applicable HTTP methods

RESOURCE	PATH	HTTP METHOD	DESCRIPTION
Landing Page	/	GET	The landing page
Conformance Declaration	/conformance	GET	The conformance information
API Definition	/api	GET	The API Definition document
Collections	/collections	GET	Collections
3D Container	/collections/{3DContainerId}	GET	3D-Container

7.1. Requirements Class “Core”

All resources are available in a ‘Core’ requirements class, i.e. all GeoVolumes APIs will support them.

REQUIREMENTS CLASS 1	
TARGET TYPE	Web API
PREREQUISITES	OGC API – Common – Part 1: Core JSON
LABEL	http://www.opengis.net/spec/ogcapi-geovolumes-1/1.0/req/core

7.1.1. Landing page

The entry point to the API (/). The landing page provides links to the service description, API definition (/api), conformance declaration (/conformance) and collections (/collections).

7.1.1.1. Operation

REQUIREMENT 1	
LABEL	/req/core/root-op
A	The server SHALL support the HTTP GET operation on the URI {root}/.

7.1.1.2. Response

REQUIREMENT 2	
LABEL	/req/core/root-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 be based upon the schema landingPage.yaml and include links to the following resources:

REQUIREMENT 2

- the API Definition (relation type 'service-desc' or 'service-doc')
- the Conformance Declaration (relation type /conformance (relation type ' http://www.opengis.net/def/rel/ogc/1.0/conformance ')
- /collections (relation type 'data')

A sample schema for the landing page of the GeoVolumes API is provided below.

```
type: object
required:
  - links
properties:
  title:
    type: string
    example:
  description:
    type: string
    example:
  links:
    type: array
    items:
      $ref: link.yaml
```

A sample landing page response document of an GeoVolumes API is provided below.

```
{
"title": "OGC 3D Pilot",
"description": "A Pilot of an API for GeoVolumes",
"links": [
    {
      "href": " http://data.example.org/[http://data.example.org/]",
      "rel": "service-desc",
      "type": "application/json",
      "title": "Service Description"
    },
    {
      "href": " http://data.example.org/api[http://data.example.org/api]",
      "rel": "service",
      "type": "application/json",
      "title": "the API definition"
    },
    {
      "href": " http://data.example.org/conformance[http://data.example.org/
conformance]",
      "rel": "conformance",
      "type": "application/json",
      "title": "Conformance"
    },
    {
      "href": " http://data.example.org/collections[http://data.example.org/
collections]",
      "rel": "data",
      "type": "application/json",
      "title": "Collections"
    }
  ]
}
```

7.1.1.3. Error situations

See HTTP Status Codes in Appendix A — Web APIAnnex A for general guidance.

7.1.2. Declaration of Conformance Classes

The Conformance Declaration states the conformance classes from standards or community specifications, identified by a URI, to which the API conforms. The conformance resource requires no parameters. The HTTP /conformance GET response returns the list of URIs of conformance classes implemented by the server in JSON.

7.1.2.1. Operation

REQUIREMENT 3	
LABEL	/req/core/conformance-op
A	The server SHALL support the HTTP GET operation on the URI /{root}/conformance.
B	The server SHALL support the HTTP GET operation on all links from the landing page that have the relation type http://www.opengis.net/def/rel/ogc/1.0/conformance.
C	The responses to all HTTP GET requests issued in A and B server SHALL satisfy requirement **/req/core/conformance-success.**

REQUIREMENT 4	
LABEL	/req/core/conformance-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 be based upon the schema confClasses.yaml and list all OGC API conformance classes that the API conforms to.

7.1.2.2. Response

A sample schema for the list of conformance classes is provided below.

```
type: object
required:
  - conformsTo
properties:
```

```
  conformsTo:
    type: array
    items:
      type: string
```

The following example of the conformance declaration of a 3D GeoVolumes API was taken from the “OGC API – Tiles — 3D (GeoVolumes) Engineering Report”:

```
{
  "conformsTo": [
    " http://www.opengis.net/spec/ogcapi-geovolumes-1/1.0/conf/core[http://www.
opengis.net/spec/ogcapi-geovolumes-1/1.0/conf/core]",
    " http://www.opengis.net/spec/ogcapi-geovolumes-1/1.0/conf/oas30[http://
www.opengis.net/spec/ogcapi-geovolumes-1/1.0/conf/oas30]",
    " http://www.opengis.net/spec/ogcapi-geovolumes-1/1.0/conf/json[http://www.
opengis.net/spec/ogcapi-geovolumes-1/1.0/conf/json]"
    ]
}
```

7.1.2.3. Error situations

See HTTP Status Codes in Appendix A — Web APIAnnex A for general guidance.

7.1.3. API Definition

The *API Definition* describes the capabilities of the server that can be used by clients to connect to the server or by development tools to support the implementation of servers and clients. Accessing the *API Definition* using HTTP GET returns a description of the API.

7.1.3.1. Operation

REQUIREMENT 5	
LABEL	/req/core/api-definition-op
A	The URIs of all API definitions referenced from the landing page SHALL support the HTTP GET method.

PERMISSION 1	
LABEL	/per/core/api-definition-uri
A	The API definition is metadata about the API and strictly not part of the API itself, but it MAY be hosted as a sub-resource to the base path of the API, for example, at path /api. There is no need to include the path of the API definition in the API definition itself.

Note that multiple API definition formats can be supported.

7.1.3.2. Response

RECOMMENDATION 1	
LABEL	/rec/core/api-definition-oas
A	A JSON representation of the API definition document SHOULD conform to the OpenAPI Specification 3.0, the document.

REQUIREMENT 6	
LABEL	/req/core/api-definition-success
A	A GET request to the URI of an API definition linked from the landing page (link relations service-desc or service-doc) with an Accept header with the value of the link property type SHALL return a document consistent with the requested media type.

If the server hosts the API definition under the base path of the API (for example, at path /api, see above), there is no need to include the path of the API definition in the API definition itself.

The idea is that any 3D GeoVolumes API implementation can be used by developers that are familiar with the API definition language(s) supported by the server. The developer may need to learn about 3D data types, etc., but it should not be required to read this specification to access the data via the API.

The following is an example of the API definition.

```
{
"links": [
    {
      "href": " http://data.example.org/[http://data.example.org/]",
      "rel": "self",
      "type": "application/json",
      "title": "this document"
    },
    {
      "href": " http://data.example.org/api[http://data.example.org/api]",
      "rel": "service",
      "type": "application/json",
      "title": "the API definition"
    },
    {
      "href": " http://data.example.org/conformance[http://data.example.org/
conformance]",
      "rel": "conformance",
      "type": "application/json",
      "title": "conformance classes implemented by this server"
    },
```

```
    {
      "href": " http://data.example.org/collections[http://data.example.org/
collections]",
      "rel": "data",
      "type": "application/json",
      "title": "Metadata about the collections"
    }
  ]
}
```

7.1.3.3. Error situations

See HTTP Status Codes in Appendix A — Web APIAnnex A for general guidance.

7.1.4. Collections

Collections provides the information to access a collection of GeoVolumes (3D Containers). The collection resources accept the 2D or 3D bounding box (bbox) and format parameter. The resource accepts query or header parameters for the format parameter. The bounding box query parameter lower left: x, y, {z}, and upper right x, y, {z} (z-coordinate is optional) returns GeoVolumes that are within the area. The HTTP /collections GET response returns JSON containing two properties, links (link: URI, type, relationship) and 3D Container.

REQUIREMENT 7	
LABEL	/req/core/collections-op
A	The server SHALL support the HTTP GET operation at the path /collections.

REQUIREMENT 8	
LABEL	/req/core/collections-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 conform to the media type stated in the Content-Type header.
C	The content of that response SHALL conform to the media type stated in the query string.
D	The content of that response SHALL be constrained by the bbox stated in the query string.

E	The content of that response SHALL be based upon the following OpenAPI 3.0 schema:
	`type: object`
	`properties:`
	`  links:`
	`  type: array`

REQUIREMENT 8

```
            items:
              type: object
              required:
                - href
                - rel
              properties:
                href:
                  type: string
                title:
                  type: string
                  nullable: true
                rel:
                  type: string
                type:
                  type: string
                  nullable: true
                hreflang:
                  type: string
                  nullable: true
            collections:
              type: array
              items:
                $ref: '3dcontainer'
```

7.1.5. 3D-Container

The collection resources support access to a 3D-Container with a unique identifier (/collections/{3DContainerId}). The format and bounding box parameters in the collections request can be applied to a specific GeoVolume request. The bbox query on a GeoVolume will apply filtering on the contents within the GeoVolume. The HTTP /collections/{3DContainerId} GET response returns JSON representing the 3D-Container (GeoVolume).

REQUIREMENT 9

LABEL	/req/core/collections/{3DContainerId}-op
A	The server SHALL support the HTTP GET operation at the path /collections/{3DContainerId}.

REQUIREMENT 10

LABEL	/req/core/collections/{3DContainerId}-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 conform to the media type stated in the Content-Type header.
C	The content of that response SHALL conform to the media type stated in the query string.

REQUIREMENT 10

D The content of that response SHALL be constrained by the bbox stated in the query string.

E The content of that response SHALL be based upon the following OpenAPI 3.0 schema:

```
type: object
required:
  - id
  - extent
  - links
properties:
  id:
    type: string
  title:
    type: string
    nullable: true
  description:
    type: string
    nullable: true
    collectionType:
      type: string
      default: '3d-container'
    itemType:
      type: string
      default: 'unknown'
    extent:
      type: object
      properties:
        spatial:
          type: object
          properties:
            bbox:
              type: array
              minItems: 4
              maxItems: 6
              items:
                type: number
            crs:
              type: string
              default:
  'link:++http://www.opengis.net/def/crs/OGC/1.3/CRS84'++[]
          temporal:
            type: object
            properties:
              interval:
                type: array
                nullable: true
                minItems: 1
                  items:
                    type: array
                    minItems: 2
                    maxItems: 2
                    items:
                        type: string
                        format: date-time
                        nullable:  true
             trs:
                  type: string
                nullable: true
                default:
  'link:++http://www.opengis.net/def/uom/ISO-8601/0/Gregorian'++[]
      contentExtent:
          type: array
          nullable: true
          items:
              type: number
              format: double
              minItems: 4
```

REQUIREMENT 10

```
            maxItems: 12
    crs:
        type: string
        default: 'link:++http://www.opengis.net/def/crs/OGC/1.3/CRS84'++[]
    links:
    type: array
    items:
      type: object
      required:
          - href
          - rel
      properties:
          href:
              type: string
          title:
              type: string
              nullable: true
          rel:
              type: string
          type:
              type: string
              nullable: true
          hreflang:
              type: string
              nullable: true
      children:
        type: array
        items:
          $ref: 3dcontainer
      content:
        type: array
        items:
          type: object
          required:
              - href
              - rel
          properties:
              href:
                  type: string
              title:
                  type: string
                  nullable: true
              rel:
                  type: string
              type:
                  type: string
                  nullable: true
              hreflang:
                  type: string
                  nullable: true
```

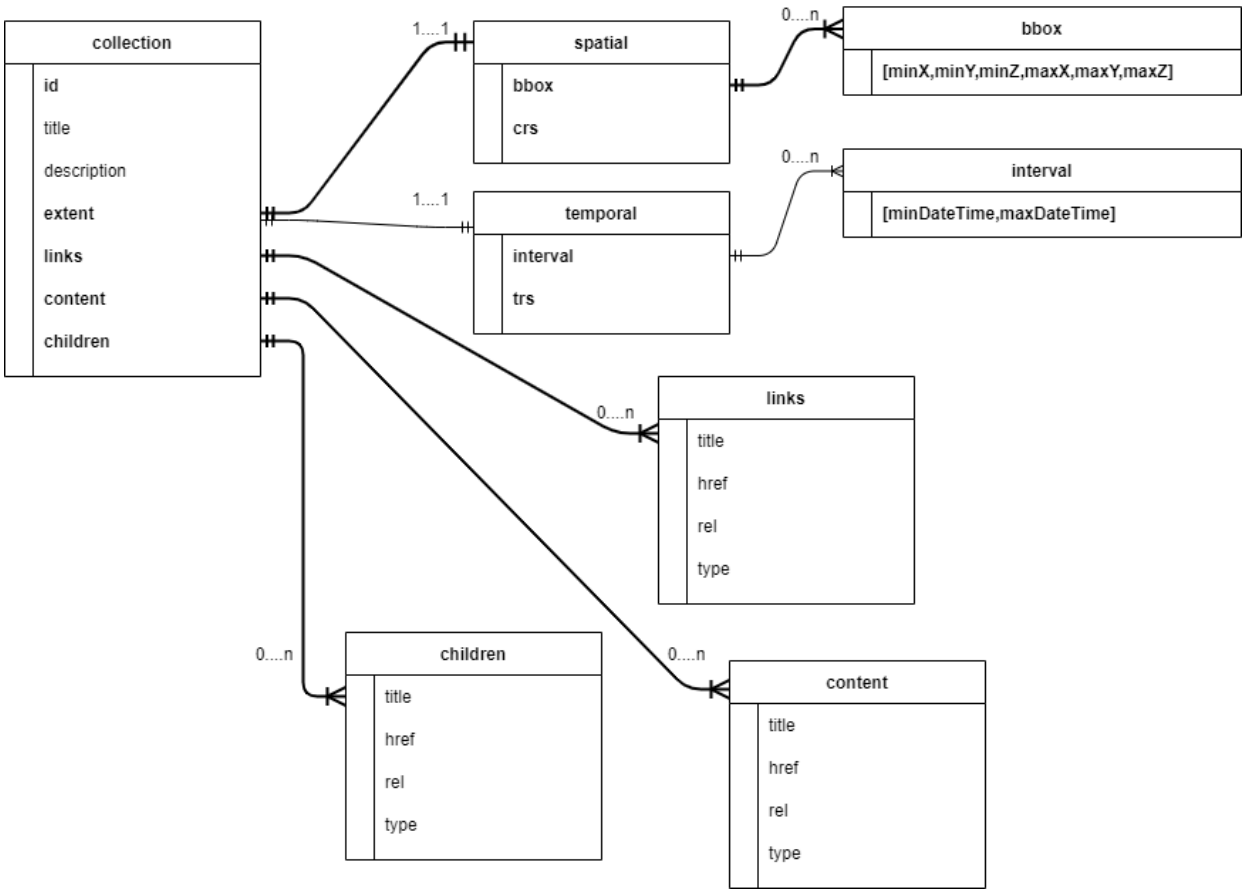


Figure 9 — UML diagram of a 3D-Container (GeoVolume)

See Table B.4 in Annex B for additional guidance.

7.2. Requirements Class "Extension"

This class provides specifics on the extensions to the Collections requirement class. The extensions to the Collections requirement class are as follows:

- The collections path (/collections) is extended by the addition of a bounding box query parameter.
- The collections path (/collections/{3DContainerId}) is extended by the addition of a bounding box query parameter.

The resulting API has the resources listed in the Table below:

Table 3 — Overview of resources and applicable HTTP methods with “bbox” extension

RESOURCE	PATH	HTTP METHOD	CHANGES
Landing page	/	GET	None
Conformance declaration	/conformance	GET	Returns additional conformance classes
API	/api	GET	API definition
Collections	/collections?bbox	GET	Bounding Box parameter added.
3D Container	/collections/{3DContainerId}?bbox	GET	Bounding Box parameter added.

The following is an example of the conformance declaration of a 3D GeoVolumes API that implements all requirement classes:

```
{
"conformsTo": [
" http://www.opengis.net/spec/ogcapi-geovolumes-1/1.0/conf/core[http://www.
opengis.net/spec/ogcapi-geovolumes-1/1.0/conf/core]",
" http://www.opengis.net/spec/ogcapi-geovolumes-1/1.0/conf/oas30[http://www.
opengis.net/spec/ogcapi-geovolumes-1/1.0/conf/oas30]",
" http://www.opengis.net/spec/ogcapi-geovolumes-1/1.0/conf/html[http://www.
opengis.net/spec/ogcapi-geovolumes-1/1.0/conf/html]",
"http://www.opengis.net/spec/ogcapi-geovolumes-
1/1.0/conf/spatialquery"
]
}
```

7.2.1. Requirements Class “spatial query extension”

The requirement class ‘spatialquery’ is an extension to the ‘Core’ requirement class which allows query by spatial and temporal constraints. The server shall return 3D content if any part of the content lies inside the query bbox when querying by spatial constraints.

REQUIREMENTS CLASS 2	
TARGET TYPE	Web API
PREREQUISITES	OGC API – Common – Part 1: Core JSON
LABEL	http://www.opengis.net/spec/ogcapi-geovolumes-1/1.0/req/spatialquery

REQUIREMENT 11

LABEL	/req/spatialquery/op-1
A	The server SHALL support the HTTP GET operation at the path/collections?bbox for a 3D Container (Geo Volume).

REQUIREMENT 12

LABEL	/req/spatialquery/op-2
A	The server SHALL support the HTTP GET operation at the path/collections/{3DContainerId}?bbox for each 3D Container(GeoVolume).

REQUIREMENT 13

LABEL	/req/spatialquery/success
A	A successful execution of the operation SHALL be reported as aresponse with a HTTP status code 200.
B	The content of the response SHALL be based upon the following OpenAPI 3.0 schema:

```
type: object
required:
  - id
  - extent
  - links
properties:
  id:
    type: string
  title:
    type: string
    nullable: true
  description:
    type: string
    nullable: true
  collectionType:
    type: string
    default: '3d-container'
  itemType:
    type: string
    default: 'unknown'
  extent:
    type: object
    properties:
      spatial:
        type: object
        properties:
          bbox:
            type: array
            minItems: 4
            maxItems: 6
            items:
              type: number
          crs:
            type: string
            default:
```

REQUIREMENT 13

```
        'http://www.opengis.net/def/crs/OGC/1.3/CRS84'
              temporal:
                type: object
                properties:
                  interval:
                    type: array
                    nullable: true
                    minItems: 1
                      items:
                        type: array
                        minItems: 2
                        maxItems: 2
                      items:
                        type: string
                        format: datetime
                        nullable: true
                    trs:
                      type: string
                      nullable: true
                      default:
        'http://www.opengis.net/def/uom/ISO-
        8601/0/Gregorian'
             contentExtent:
               type: array
               nullable: true
               items:
                 type: number
                 format: double
                 minItems: 4
                 maxItems: 12
            crs:
              type: string
              default:
        'http://www.opengis.net/def/crs/OGC/1.3/CRS84'
            links:
              type: array
              items:
                type: object
                required:
                  - href
                  - rel
                properties:
                  href:
                    type: string
                  title:
                    type: string
                    nullable: true
                  rel:
                    type: string
                  type:
                    type: string
                    nullable: true
                  hreflang:
                    type: string
                    nullable: true
                children:
                  type: array
                  items:
                    $ref: 3dcontainer
```

C The id member of each spatial query SHALL be unique.

8

MEDIA TYPES FOR ANY DATA ENCODING(S)

8 MEDIA TYPES FOR ANY DATA ENCODING(S)

The following media types are applicable to the 3D GeoVolumes API:

- application/json – is the JSON media type used for all content.
- text/html – is the HTML media type for all “web pages” provided by the API.

The media types in the following sections are informative to the 3D GeoVolumes API.

8.1. application/json+i3s

Type name: application

Subtype name: json+i3s

```
Required parameters: n/a
Optional parameters: n/a
Encoding considerations: See RFC 8259, The JavaScript Object Notation (JSON)
Data
Interchange Format
Security considerations: See Section 12 of RFC 8259
Interoperability considerations: n/a
Published specification:
link: n/a
Applications that use this media type: n/a
Additional information:
  Deprecated alias names for this type: n/a
  Magic number(s): n/a
  File extension(s): .json
  Macintosh file type code(s): n/a
Person to contact for further information: n/a
Intended usage: COMMON
Restrictions on usage: none
Author: n/a
Change controller: n/a
```

Figure 11

8.2. application/json+3dtiles

Type name: application

Subtype name: json+3dtiles

```
Required parameters: n/a
Optional parameters: n/a
```

```
Encoding considerations: See RFC 8259, The JavaScript Object Notation (JSON)
Data Interchange Format
Security considerations: See Section 12 of RFC 8259
Interoperability considerations: n/a
Published specification: n/a
link: n/a
Applications that use this media type:
Additional information:
  Deprecated alias names for this type: n/a
  Magic number(s): n/a
  File extension(s): .json
  Macintosh file type code(s): n/a
Person to contact for further information: n/a
Intended usage: COMMON
Restrictions on usage: none
Author: n/a
Change controller: n/a
```

Figure 12

A

ANNEX A (NORMATIVE) ABSTRACT TEST SUITE (NORMATIVE)

A

ANNEX A (NORMATIVE) ABSTRACT TEST SUITE (NORMATIVE)

A.1. Introduction

The Abstract Test Suite (ATS) provides a basis for developing an Executable Test Suite to verify that the implementation under test conforms to all the requirements specified in the requirements class associated with a conformance class.

A.2. Conformance Class Core

CONFORMANCE CLASS A.1	
SUBJECT	Requirements Class “Core”
TARGET TYPE	Web API
LABEL	http://www.opengis.net/spec/ogcapi-3dgeovolumes-1/1.0/conf/core

ABSTRACT TEST A.1	
SUBJECT	/req/core/collections-op,/req/core/collections-success
LABEL	/conf/core/fc-md-op
TEST PURPOSE	Verify that the collections resources can be fetched.
DESCRIPTION	**Example 1:** Issue an HTTP GET request to the path /collections with header Accept: application/json.

ABSTRACT TEST A.1

	Issue an HTTP GET request to the path /collections?f=json. Issue an HTTP GET request to the path `/collections?bbox=x1,y1,{z1},x2,y2,{z2}. Validate that the response has a status code 200. Validate the contents of the returned document against the schema in: /collections, item E. Verify that each container id #/collections/{i} (where {i} is the index of the style in the array) is unique. Verify that each collection has at least one link with rel=self. Verify that for each link with rel=self that the href value links to a resource at the path /collections/{3DContainerId} where {3DContainerId} is the id member of the 3d Container. If each collection has content greater than one, verify each content contains rel=original or rel=alternate

ABSTRACT TEST A.2

SUBJECT	/req/core/collections/{3DContainerId}-op,/req/core/collections/{3DContainerId}-Success
LABEL	/conf/core/3dcontainer
TEST PURPOSE	Verify that the 3D Container resources can be fetched.
DESCRIPTION	**Example 2:** Issue an HTTP GET request to the path /collections/{3DContainerId} with header Accept: application/json. Issue an HTTP GET request to the path /collections/{3DContainerId}?f=json. Issue an HTTP GET request to the path `/collections/{3DContainerId}?bbox=x1,y1,{z1},x2,y2,{z2}. Validate that the response has a status code 200. Validate the contents of the returned document against the schema in: /collections/{3DContainerId}, item E. Verify that each container id #/collections/{i} (where {i} is the index of the style in the array) is unique. Verify that each collection has at least one link with rel=self. Verify that for each link with rel=self that the href value links to a resource at the path /collections/{3DContainerId} If each collection has content greater than one, verify each content contains rel=original or rel=alternate

B

ANNEX B (INFORMATIVE) ANNEX B: WEB API (NORMATIVE)

B

ANNEX B (INFORMATIVE) ANNEX B: WEB API (NORMATIVE)

B.1. HTTP 1.1

REQUIREMENT B.1	
LABEL	/req/core/http
A	The server SHALL conform to HTTP 1.1.
B	If the server supports HTTPS, the server SHALL also conform to HTTP over TLS.

This includes the correct use of status codes, headers, etc.

RECOMMENDATION B.1	
LABEL	/rec/core/head
A	The server SHOULD support the HTTP 1.1 method HEAD for all resources that support the method GET.

Supporting the method HEAD in addition to GET can be useful for clients and is simple to implement.

Servers implementing CORS will implement the method OPTIONS, too.

B.2. HTTP status codes

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

status codeThe **Status Codes** listed in Table 3 are of particular relevance to implementers 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 3 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 B.1 — Typical HTTP status codes

Typical HTTP status codes		
STATUS CODE	Mandatory / Optional	Description
200	**Mandatory**	A successful request.
304	Optional	An entity tag was provided in the request and the resource has not been changed since the previous request.
400	**Mandatory**	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	Optional	The request requires user authentication. The response includes a WWW-Authenticate header field containing a challenge applicable to the requested resource.
403	Optional	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 authorized to perform the requested operation on the resource.
404	**Mandatory**	The requested resource does not exist on the server. For example, a path parameter had an incorrect value.
405	Optional	The request method is not supported. For example, a POST request was submitted, but the resource only supports GET requests.
406	Optional	The Accept header submitted in the request did not support any of the media types supported by the server for the requested resource.
500	Optional	An internal error occurred in the server.

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

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

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.

B.3. Unknown or invalid query parameters

REQUIREMENT B.2	
LABEL	/req/core/query-param-unknown
A	The server SHALL respond with a response with the status code 400, if the request URI includes a query parameter that is not specified in the API definition.

If a server wants to support vendor specific parameters, these must be explicitly declared in the API definition.

If OpenAPI is used to represent the API definition, a capability exists to allow additional parameters without explicitly declaring them. That is, parameters that have not been explicitly specified in the API definition for the operation will be ignored.

OpenAPI schema for additional “free-form” query parameters

```
in: query
name: vendorSpecificParameters
schema:
  type: object
  additionalProperties: true
style: form
```

Note that the name of the parameter does not matter as the actual query parameters are the names of the object properties. For example, assume that the value of vendorSpecificParameters is this object:

```
{
  "my_first_parameter": "some value",
  "my_other_parameter": 42
}
```

In the request URI this would be expressed

as `&my_first_parameter=some%20value&my_other_parameter=42`.

REQUIREMENT B.3	
LABEL	/req/core/query-param-invalid
A	The server SHALL respond with a response with the status code 400, if the request URI includes a query parameter that has an invalid value.

This is a general rule that applies to all parameters, whether they are specified in this document or in additional parts. A value is invalid, if it violates the API definition or any other constraint for that parameter stated in a requirement.

B.4. Web caching

Entity tags are a mechanism for web cache validation and for supporting conditional requests to reduce network traffic. Entity tags are specified by [HTTP/1.1 (RFC 2616)].

RECOMMENDATION B.2	
LABEL	/rec/core/etag
A	The service SHOULD support entity tags and the associated headers as specified by HTTP/1.1.

B.5. Support for cross-origin requests

Access to data from a HTML page is by default prohibited for security reasons if the data is located on another host than the webpage ("same-origin policy"). A typical example is a web-application accessing feature data from multiple distributed datasets.

RECOMMENDATION B.3	
LABEL	/rec/core/cross-origin
A	If the server is intended to be accessed from the browser, cross-origin requests SHOULD be supported. Note that support can also be added in a proxy layer on top of the server.

Two common mechanisms to support cross-origin requests are:

- Cross-origin resource sharing (CORS)

- JSONP (JSON with padding)

C

ANNEX C (INFORMATIVE) ANNEX C: DATA ARCHITECTURE (INFORMATIVE)

C

ANNEX C (INFORMATIVE) ANNEX C: DATA ARCHITECTURE (INFORMATIVE)

The data architecture can be separated into three parts:

1. Common: Landing Page, Conformance Declaration, API Definition
2. GeoVolumes content: Collections, 3D Container
3. Supporting metadata components

Table C.1

SCHEMAS	OVERVIEW
Landing Page	OGC API — Common Landing Page
Conformance	OGC API — Common Conformance Declaration
API Definition	OGC API — Common API Definition
Collections	In this specification, 'collection' is used as a synonym for '3d-container collection'.
3D Container	Spatial information resource that has a distinct bounding volume containing 3D content.
Bounding Volume	A closed volume completely containing the union of a set of geometric objects.
CRS	Coordinate reference system describing coordinates of the extents.
Exception	An exception describes an event, which occurs during execution that disrupts the normal flow of the program's instructions.
Extent	The extent of the collection.
Link	Link to content.
Spatial Extent	The spatial extent of the element in the collection.

Temporal Extent	The temporal extent of the element in the collection.
TRS	Coordinate reference system of the coordinates in the temporal extent.
Content Type	JSON media type used for content.

C.1. GeoVolumes (3D-Container)

The most general definition of a GeoVolume is a spatial information resource with a distinct bounding volume, a (required) enclosing bounding box (2D / 3D), containing at most one 3D model dataset which is relevant to that volume (items, content) and represented by references to one or more distributions, and includes or references other GeoVolumes (children) whose bounding volumes are fully contained by the parent container's bounding volume. The default representations of a GeoVolume are JSON / HTML information documents which define the bounding box, link to an implicit tileset scheme if applicable, and provide the described sections and links.

This resource is not the same as the collection resource defined in the present draft OGC API — Common but could be considered a specialized type of that collection resource. The collection resource and corresponding collection information document schema were extended in ways that in turn echo the 3D Tiles root node resource (tileset.json). As the definitions of *collections* and *collection* evolve with development of OGC API — Common modules, this recommendation may be superseded.

Choices can be made whether to provide a flat list of all GeoVolumes supported by the API instance or only 1-2 top-level or "root" 3D-Container collections which then include child 3D-Containers. The collection information in the collections array may also consist only of the required elements, leaving other elements to be included in the document at the individual collection path.

Table C.2 — GeoVolume (3D-Container)

FIELD NAME	REQUIRED	TYPE	DESCRIPTION
Id	X	String	Identifier (name) of a specific 3D Container.
Title	+	String	Human readable title of a specific 3D Container.
description	+	String	Detailed description of a specific 3D Container.
collection Type	+	String	Property with value "3d-container" is required.

FIELD NAME	REQUIRED	TYPE	DESCRIPTION
itemType	+	String	Indicator about the type of the items in the collection (the default value is 'unknown').
Extent	X	extent	The 3D spatial extent of the container is required.
contentExtent	+	boundingVolume	Optional 3D spatial extent as box, region, or sphere.
Crs	+	crs	Coordinate Reference System
Links	X	List of [link]	Array members in the "link" property must include "self", and may include "items" (for compatibility with a common collection type), "parent" (link to enclosing 3D-Container), "root" (link to top-level 3D-Container), "scheme" link to the definition of any implicit tile scheme which sets the 3D-Container organization, extent and/or addressing, "affinemap" (link from a 3D-Container with 2D content to a 3D-Container with 2.5D content (surface, point cloud) to which the 2D content should S be texture-mapped).
Children	+	List of [3dcontainer]	A "children" property with an array of zero or more "child" 3D-Containers is required (could be just the required properties: id, self-link,extent).
content	+	List of [link]	A property with an array of zero or more content references is required. The content of a3D-Container is at most a single dataset (e.g. it may have no content and only children). Each link in the "content" array shall be to a specific distribution of that dataset. The "rel" of each reference indicates its relation to the dataset, such as "original" (the distribution representing the most original version of the dataset, e.g. a CityGML model), or "alternate"(other dataset distributions in different encodings or for different platforms).What a content reference links to is dependent on content type and TBD for some types: * 3DTiles: tileset.json * I3S: NodeIndexDocument * City GML: Collection document and/or logical space feature (CityModel) * CDB: Root folder * 2D features: link to collection information document

C.2. Bounding Volume

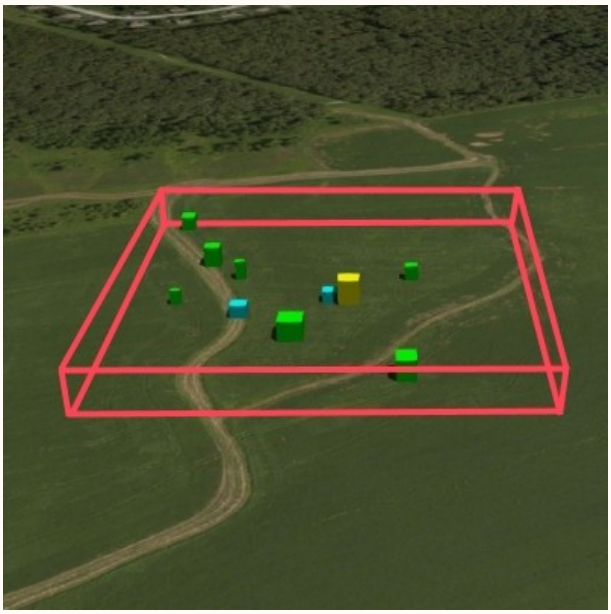
A bounding volume for a set of objects is a closed volume that completely contains the union of the objects in the set.

Exactly one box, region, or sphere property is required. See 3D-Tiles Bounding volumes [http://docs.opengeospatial.org/cs/18-053r2/18-053r2.html%2331]

Table C.3 — Bounding Volume

FIELD NAME	REQUIRED	TYPE	DESCRIPTION
Box	X	List of [double]	An array of 12 numbers that define an oriented bounding box. The first three elements define the x, y, and z values for the center of the box. The next three elements (with indices 3, 4, and 5) define the x-axis direction and half-length. The next three elements (indices 6, 7, and 8) define the y-axis direction and half-length. The last three elements (indices 9, 10, and 11) define the z-axis direction and half-length.
region	+	List of [double]	An array of six numbers that define a bounding geographic region in EPSG:4979 coordinates with the order [west, south, east, north, minimum height, maximum height]. Longitudes and latitudes are in radians, and heights are in meters above (or below) the WGS84 ellipsoid.
sphere	+	List of [double]	An array of four numbers that define a bounding sphere. The first three elements define the x, y, and z values for the center of the sphere. The last element (with index 3) defines the radius in meters.

Table C.4



Bounding Box YAML:

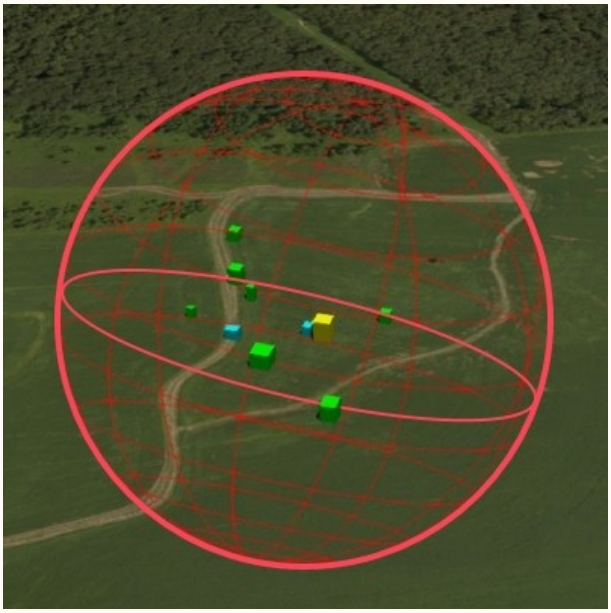
```yaml
box:
  type: array
  items:
    type: number
    format: double
  minItems: 12
  maxItems: 12
```

Figure C.1

Bounding Region YAML:

```yaml
box:
  type: array
  items:
    type: number
    format: double
  minItems: 6
  maxItems: 6
```

Figure C.2



Bounding Sphere YAML:

```yaml
box:
  type: array
  items:
    type: number
    format: double
  minItems: 4
  maxItems: 4
```

Figure C.3

C.3. Coordinate Reference System (CRS)

This is the CRS of the coordinates in the spatial extent (property 'bbox'). The default reference system is EPSG:4979 / WGS 84 longitude/latitude/height [http://www.opengis.net/def/crs/

OGC/0/CRS84h]. In the Core this is the only supported CRS. Extensions may support additional coordinate reference systems and add additional enum values.

C.4. Extent

This is the extent of the 3D GeoVolume. In the Core only spatial and temporal extents are specified. Extensions may add additional members to represent other extents, for example, thermal or pressure ranges. It is recommended that the spatial extent be expressed in CRS84 except if this is not possible.

Table C.5 — Extent

FIELD NAME	REQUIRED	TYPE	DESCRIPTION
extent	+	spatialExtent	+
temporal	+	temporalExtent	+

C.5. Link

Link to content.

Table C.6 — Link

FIELD NAME	REQUIRED	TYPE	DESCRIPTION
href	X	String	The URI of the link
title	+	String	+
rel	+	rel	Link Relationship
Type	+	type	+
Hreflang	+	String	+

C.6. Link Relation Type (Rel)

This defines the relationship between the current JSON resource representation and a related JSON resource. For more information see:

- Link Relations [http://docs.opengeospatial.org/is/17-069r3/17-069r3.html%23_link_relations]
- IANA: Link Relation Types [https://www.iana.org/assignments/link-relations/link-relations.xml]

The following enumeration is provided as an example. Other relationship types are possible.

- affinemap: Link from a 3D-Container with 2D content to a 3D-Container with 2.5D content (surface, point cloud) to which the 2D content should be texture-mapped.
- alternate: Refers to a substitute for this context.
- collections: The target points to a 3D-container resource which represents the collection resource for the context.
- conformance: Refers to a resource that identifies the specifications that the link's context conforms to.
- data: Indicates that the link's context is a distribution of a dataset that is an API and refers to the root resource of the dataset in the API.
- dataset: Indicates that the link's context is a distribution of a dataset that is an API and refers to the root resource of the dataset in the API.
- describedby: Refers to a resource providing information about the link's context.
- distribution: Indicates that the link's context is a distribution.
- item: The target IRI points to a resource that is a member of the collection represented by the context IRI.
- items: Refers to a resource that is comprised of members of the collection represented by the link's context.
- original: The distribution representing the most original version of the dataset, e.g. a CityGML model.
- parent: link to enclosing 3D-Container.
- root: link to top-level 3D-Container.
- scheme: link to the definition of any implicit tile scheme which sets the 3D-Container organization, extent and/or addressing.

- self: Conveys an identifier for the link's context.
- service: Indicates a URI that can be used to retrieve a service document.
- service-desc: Identifies service description for the context that is primarily intended for consumption by machines.

C.7. Spatial Extent

This defines the spatial extent of the element in the collection.

Table C.7 — Spatial Extent

FIELD NAME	REQUIRED	TYPE	DESCRIPTION
bbox	X	List of [array]	One or more bounding boxes that describe the spatial extent of the dataset (the start and end). The value 'null' is supported and indicates an open time interval. In the Core only a single time interval is supported. In the Core only a single bounding box is supported. Extensions may support additional areas. If multiple areas are provided, the union of the bounding boxes describes the spatial extent.
crs	+	crs	Coordinate reference system of the coordinates in the spatial extent (property bbox). The default reference system is WGS 84 longitude/latitude. In the Core this is the only supported coordinate reference system. Extensions may support additional coordinate reference systems and add additional enum values.

C.8. Temporal Extent

This defines the temporal extent of the element in the collection.

Table C.8 — Temporal Extent

FIELD NAME	REQUIRED	TYPE	DESCRIPTION
interval	+	Array of Strings [date-time] Date-time EX: 2011-11-11T12:22:11Z	One, or more, time intervals that describe the temporal extent of the dataset. The value 'null' is supported and indicates an open time interval. In the Core only a single time interval is supported. Extensions may support multiple intervals. If multiple intervals are provided, the union of the intervals describes the temporal extent.
trs	+	String	Temporal coordinate reference system of the coordinates in the temporal extent (property interval). The default reference system is

FIELD NAME	REQUIRED	TYPE	DESCRIPTION
			the Gregorian calendar. In the Core this is the only supported temporal reference system. Extensions may support additional temporal reference systems and add additional enum values.

C.9. TRS

This defines the coordinate reference system of the coordinates in the temporal extent (property 'interval'). The default reference system is the Gregorian calendar. In the Core this is the only supported temporal reference system. Extensions may support additional temporal reference systems and add additional values.

- http://www.opengis.net/def/uom/ISO-8601/0/Gregorian

C.10. Content Type

Content type enumeration examples:

- application/json
- application/json+3dtiles
- application/json+i3s
- text/html

D

ANNEX D (INFORMATIVE) ANNEX D: EXAMPLE RESPONSES (INFORMATIVE)

D

ANNEX D (INFORMATIVE) ANNEX D: EXAMPLE RESPONSES (INFORMATIVE)

This section provides examples of how servers may respond to requests issued by using the GeoVolumes API.

Example 1: The server responds with only one GeoVolume from the container ID (“NewYork/NewYork-buildings”). The container can contain multiple 3D Containers, with multiple formats, and with a flat or hierarchical organization.


```
{
"id": "NewYork/NewYork-buildings",
"title": "NYC - 3D Buildings Manhattan",
"description": "3D Buildings in Manhattan, New York.",
"collectionType": "3d-container",
"extent": {
  "spatial": [
    -74.01900887327089,
    40.700475291581974,
    -11.892070104139751,
    -73.9068954348699,
    40.880256294183646,
    547.7591871983744
  ],
  "crs": " http://www.opengis.net/def/crs/OGC/0/CRS84h[http://www.opengis.net/
def/crs/OGC/0/CRS84h]"
  },
  "links": [
    {
      "rel": "self",
      "href": " http://example.org/collections/NewYork/NewYork-buildings/[http:
//example.org/collections/NewYork/NewYork-buildings/]",
      "type": "application/json",
      "title": "NYC - 3D Buildings Manhattan"
    },
    {
      "rel": "items",
      "href": "http://example.org/collections/NewYork/NewYork-buildings/i3s/",
      "type": "application/json+i3s",
      "title": "NYC - 3D Buildings Manhattan: i3s"
    },
    {
      "rel": "items",
      "href": "http://example.org/collections/NewYork/NewYorkbuildings/3dTiles/
",
      "type": "application/json+3dtiles",
      "title": "NYC - 3D Buildings Manhattan: 3D Tiles"
    }
```

```
  ],
  "content": [
    {
      "title": "NYC - 3D Buildings Manhattan: i3s",
      "rel": "original",
      "href": "http://example.org/collections/NewYork/NewYork-buildings/i3s/",
      "type": "application/json+i3s",
      "collectionType": "3d-container"
    },
    {
      "title": "NYC - 3D Buildings Manhattan: 3D Tiles",
      "rel": "original",
      "href": " http://example.org/NewYork/NewYork-buildings/3dTiles/[http://
example.org/NewYork/NewYork-buildings/3dTiles/]",
      "type": "application/json+3dtiles",
      "collectionType": "3d-container"
    }
  ]
}
```


Example 2: The client parses the 3D Container for 3D Tiles link. Client 3D Tiles/tileset. Response. The server response with tileset.json.


```
{
  "asset": {
    "version": "1.0",
    "extras": {
      "ion": {
        "georeferenced": true,
        "movable": false
      }
    }
  },
  "properties": {
    "Height": {
      "maximum": 547.7591871983744,
      "minimum": -11.892070104139751
    },
    "Latitude": {
      "maximum": 40.880256294183646,
      "minimum": 40.700475291581974
    },
    "Longitude": {
      "maximum": -73.9068954348699,
      "minimum": -74.01900887327089
    }
  },
  "geometricError": 740.0197559011849,
  "root": {
    "boundingVolume": {
      "region": [
        -1.29187544264487,
        0.7103573144863446,
        -1.289919109210917,
        0.7134950819190251,
        0,
        547.6909683533274
      ]
    },
    "geometricError": 740.0197559011849,
    "refine": "ADD",
    "children": []
  }
```

```
}
```

E

ANNEX E (INFORMATIVE) ANNEX E: ACCESSING 3D CONTENT BY TILE COORDINATES (INFORMATIVE)

E

ANNEX E (INFORMATIVE) ANNEX E: ACCESSING 3D CONTENT BY TILE COORDINATES (INFORMATIVE)

This informative annex proposes approaches for extending TileMatrixSets and TileSet metadata and using Implicit Tiling for indexing and accessing 3D content as tiles and using a web API is used.

EXPAND

F

ANNEX F (INFORMATIVE) REVISION HISTORY

ANNEX F (INFORMATIVE) REVISION HISTORY

Table F.1 — Revision History

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2022-02-03	0.0	G.Buehler	all	Conversion to metanorma asciidoc

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Timothy Miller, Gil Trenum, Josh Lieberman: OGC 20-029, *3D Data Container Engineering Report*. Open Geospatial Consortium (2020). https://docs.ogc.org/per/20-029.html.

[2] Tim Miller, Gil Trenum, Ingo Simonis: OGC 20-031, *3D Data Container and Tiles API Pilot Summary Engineering Report*. Open Geospatial Consortium (2020). https://docs.ogc.org/per/20-031.html.

[3] Timothy Miller and Gil Trenum: OGC 20-030, *OGC API — Tiles — 3D (GeoVolumes) Engineering Report*. Open Geospatial Consortium (2020). https://docs.ogc.org/per/20-030.html.

[4] Volker Coors: OGC 17-046, *OGC Testbed-13: 3D Tiles and I3S Interoperability and Performance Engineering Report*. Open Geospatial Consortium (2018). https://docs.ogc.org/per/17-046.html.

[5] Clemens Portele: OGC 19-010r2, *OGC Testbed-15: Styles API Engineering Report*. Open Geospatial Consortium (2019). https://docs.ogc.org/per/19-010r2.html.

[6] Sam Meek, Theo Brown, Clemens Portele: OGC 19-041r3, *OGC® Routing Pilot ER*. Open Geospatial Consortium (2020). https://docs.ogc.org/per/19-041r3.html.

[7] Cesium: 3D Tiles Next (2022). https://github.com/CesiumGS/3d-tiles/tree/main/next

[8] *OGC Testbed-15: Maps and Tiles API Engineering Report* http://www.opengis.net/doc/PER/t15-D014