3D Data Container and Tiles API Pilot Summary Engineering Report

Publication Date: YYYY-MM-DD

Approval Date: YYYY-MM-DD

Submission Date: YYYY-MM-DD

Reference number of this document: OGC 20-031

Reference URL for this document: http://www.opengis.net/doc/PER/20-031

Category: OGC Public Engineering Report

Editor: Tim Miller, Gil Trenum, Ingo Simonis

Title: 3D Data Container and Tiles API Pilot Summary Engineering Report

---

OGC Public Engineering Report

WARNING

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Public Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

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.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for

complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Table of Contents

Chapter 1. Subject

This Engineering Report summarizes the purpose and key results of the 3D Data Container and Tiles API Pilot [https://www.ogc.org/projects/initiatives/3dt], an OGC Innovation Program [https://www.ogc.org/ogc/programs/ip] initiative conducted between October 2019 and July 2020. In the context of both existing and emerging 3D and 2D standards, the focus of the Pilot was on the exchange and visualization of 3D data using open standards.

[1] Source: **Tiles-3D Community Standard** [https://www.ogc.org/standards/3DTiles] 3D Tiles Community Standard 1.0 (18-053r2)

Chapter 2. Executive Summary

A variety of solutions and standards co-exist to access and transfer 3D geospatial content over the internet (e.g. 3D Tiles [https://www.ogc.org/standards/3DTiles], I3S [https://www.ogc.org/standards/i3s], glTF [https://www.khronos.org/gltf/], CDB [https://www.ogc.org/standards/cdb], CityGML [https://www.ogc.org/standards/citygml]). These solutions were developed for various technical and commercial reasons. They use different distribution mechanisms and are optimized for particular user requirements and bandwidth situations (e.g. image stream, scenes, or raw vector data delivery). As each of these co-existing solutions binds the user to a particular approach, it is challenging to access a variety of 3D content from different providers. The 3D Data Container and Tiles API Pilot addressed this challenge. The Pilot achieved its goal to develop a resource model and corresponding Application Programming Interface (API) to integrate various approaches into a single, open standards based solution. The developed GeoVolumes API and 3D Container resource model allow efficient discovery and access of 3D content based on a space-centric perspective.

The goal of the 3D Data Container and Tiles API Pilot was not to replace existing APIs and distribution models for 3D data, but to develop an integration concept for existing OGC 3D delivery standards to:

- support smooth transitions between 2D and 3D environments;
- allow applications to get 2D, 2.5D, and 3D resources; and
- enable 3D bounding volumes to support multiple types of 3D content.

To achieve these goals, the Pilot developed a 3D geospatial data container and a corresponding draft OGC API - GeoVolumes providing browse and query access to 3D geospatial content. GeoVolumes (aka 3D Containers) can be hierarchically defined with nesting support. The 3D data resources supported by the GeoVolumes API include feature geometries, feature attribute values, elevation models, texture data, and other resource types. The API provides both link-follow and bounding-box query methods of access to 2D and 3D content in a manner independent of the underlying data store. Multiple standard geospatial distribution formats such as 3D Tiles, I3S, CDB, and CityGML are supported for streamed data delivery by means of the GeoVolumes API.

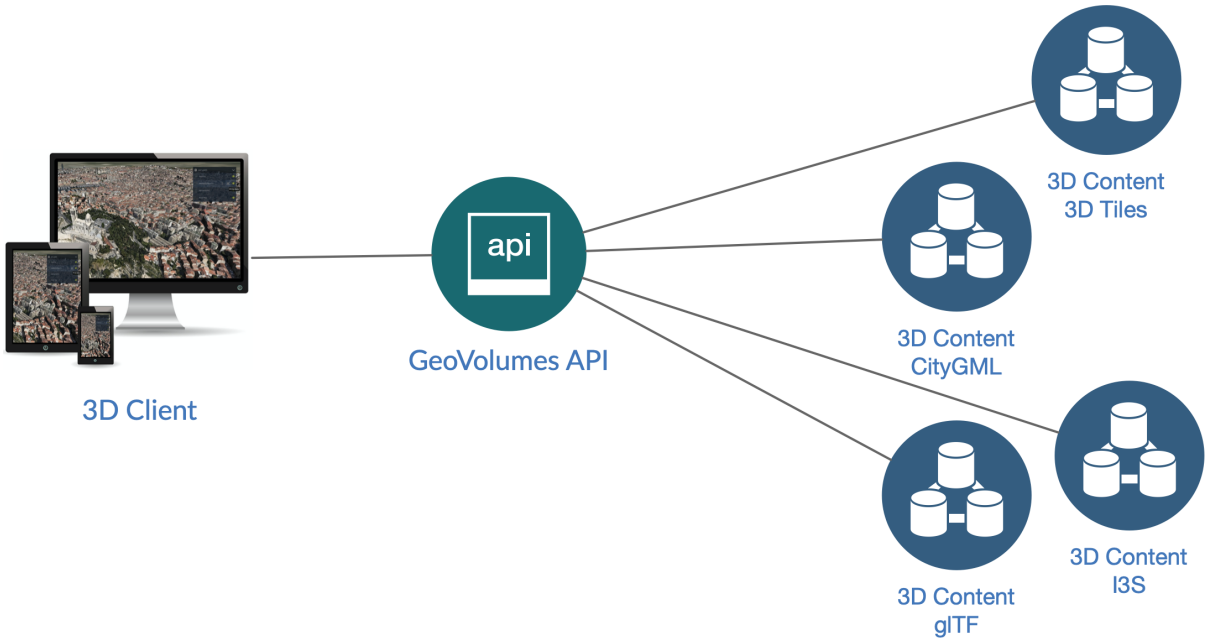


Figure 1. Client connecting to various offerings in multiple formats through single API

Multiple data server and client implementations were developed during the Pilot in order to test interoperable 3D content delivery via the GeoVolumes API. The API was defined using the OpenAPI 3.0 definition language and conforms to the building blocks of the draft OGC API - Common – Part 1: Core [http://docs.opengeospatial.org/DRAFTS/19-072.html] specification. Thus, the Pilot developed and tested the GeoVolumes API in order to advance open standards-based and unified approaches for delivering 3D content using state of the art API practices that work across different data formats, streaming protocols, and model types.

This summary first outlines a high-level architecture with different distribution levels, bandwidth options, and required capabilities. This approach helps to better understand where the GeoVolume (also known as 3D Container) and its corresponding API actually exist and integrate with other established or emerging standards. The summary then briefly introduces differences between existing solutions to further illustrate the challenge described above.

Once the context is set, the summary outlines the technical challenges that had to be addressed, defines the GeoVolume/3D Container (3DC), describes its capabilities and constraints, and introduces the API that uses the GeoVolume as its resource model. The summary describes how wide-spread tools have been used to document the building blocks of the API and provides further insights into taken design decisions using statements from the Pilot participants on initial challenges, tested approaches, and final solutions. The summary briefly illustrates the developed prototypes and ends with an assessment of the commercial potential of the proposed solutions, their future path in the standardization process, and required next steps towards a fully interoperable, space-centric 3D data exchange and exploration environment.

From a standardization perspective, the Pilot developed a draft API and resource model that have been proven mature enough to be introduced to OGC's Standards Program. The next steps now include generating a new OGC Standards Working Group (SWG), which requires defining the SWG charter. Once the charter is approved by the OGC Technical and Planning Committees (TC & PC), the SWG starts with the consensus based standard development process and eventually

recommends to the Technical Committee the release of the final Standard to the public.

It is important that additional integration tests and real-world deployment demonstrations further explore the potential of the GeoVolumes API, resolve any outstanding interoperability issues, and explore best practices for the organization of GeoVolumes within different domains and disciplines. Several of these tests are scheduled to begin in September 2020.

2.1. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Contacts

Name	Organization	Role
Ryan Gauthier	US Army Geospatial Center	Sponsor/Contributor
Jeff Harrison	US Army Geospatial Center	Sponsor/Contributor
Tom Myers	US Army Geospatial Center	Sponsor/Contributor
Tim Miller	Leidos	Editor
Gil Trenum	Leidos	Editor
Ingo Simonis	OGC	Editor
Josh Lieberman	OGC	Contributor
Rob Jones	Helyx	Contributor
Matthew Knight	Helyx	Contributor
Anneley Hadland	Helyx	Contributor
Jerome Jacovella-St-Louis	Ecere	Contributor
Michala Hill	Cognitics	Contributor
Nacho Correas	Skymantics	Contributor
Volker Coors	Steinbeis	Contributor
Thunyathep Santhanavanich	Steinbeis	Contributor
Kevin Ring	Cesium	Contributor

2.2. Foreword

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.

Chapter 3. High-Level Architecture

The high-level architecture is built around the needs of the various entities ('A'), ('B'), and ('C') in an enterprise as illustrated in the Figure 2. The three entities differ in the amount of data that can be stored and processed, the available bandwidth for data transport, the multiplicity of supported analytics, and the offered data products. Despite these differences, it was the goal of this Pilot to develop a model that allows offering, discovering, requesting, and processing data at each entity using a common single API on top of a single organizational model of 3D data in space. At the same time, this common single API should acknowledge available 3D data format 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], or CDB [https://www.ogc.org/standards/cdb] to ensure that customers can retrieve 3D data in the optimal format for their respective tasks.

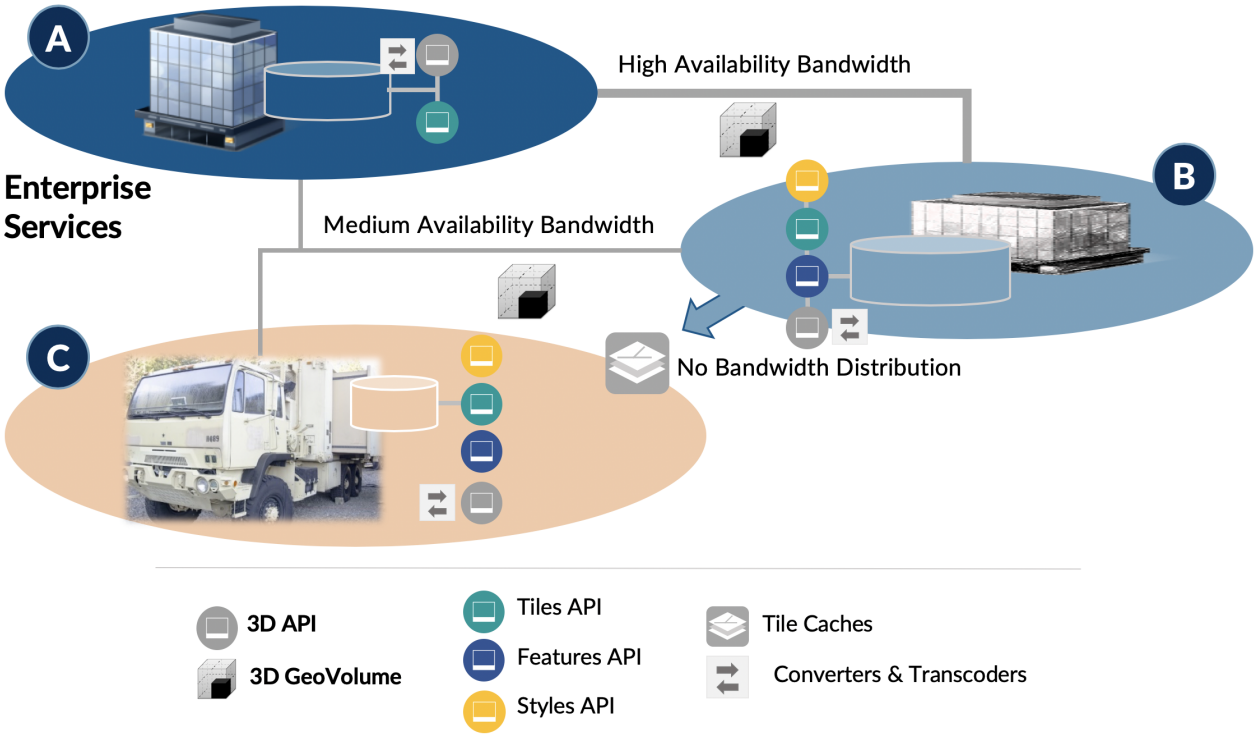


Figure 2. High level architecture with high capacity data center ('A'), medium capacity data center ('B'), and low capacity field operations ('C').

High capacity data centers ('A') have 3D data available for broad regions or on a global scale. Data can be made available in multiple formats and distributions based on a set of converters and transcoders. Access to the data is offered to other entities in the enterprise by means of the GeoVolumes API as developed in this Pilot. Depending on users' needs, data can be made available at alternative APIs in 2D or in raw format, such as e.g. OGC API Tiles or OGC API Features.

Medium capacity data centers ('B') do not require all data that is available at the data center ('A'), but are more selective. 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 capacity data centers make use of the specific space-centric indexing scheme that is the fundamental idea of GeoVolumes/3D Container and the

corresponding GeoVolumes API offered by ('A'). The indexing scheme is illustrated further below. It allows organizing data according to the human conceptual model of space. That is, it makes data available that corresponds to cognitive entities like a country with its cities and cities with its buildings, streets, or underground infrastructure. Medium capacity data centers again make their products available at the GeoVolume API; following the same conceptual model and distribution options as the high capacity data centers.

The high-level architecture defines a third enterprise entity, the low capacity field operations ('C'). These are connected at various bandwidth down to complete offline situations. In these cases, offline data packaging mechanisms and data volume optimized data selection and transmission processes are essential. Customers at this level want to go back and forth between 3D data optimized for visualization via low bandwidth connections and attribute-loaded data that provides detailed information about selected elements in a given view. That is, server-side rendering pipelines need to be integrated with scene graphs delivered to client applications.

Chapter 4. Existing Space-Centric 3D Standards

Several 3D data related standards are currently available or in development. In the context of OGC, these are CityGML and CDB on the data model and storage format end, and the 3D Portrayal Service (3DPS) as a content transmission format agnostic service for 3D scene navigation and retrieval of feature information. The community standards 3D Tiles and I3S exist for data delivery. On the content transmission side, additional standards such as X3D [https://www.web3d.org/standards/number/19775-1] or glTF [https://www.khronos.org/gltf/] have been explored in the OGC context several times. All these standards either serve different roles or have different strengths and weaknesses within 3D data visualization and analysis pipelines and will co-exist in future. A detailed comparison between 3D transmission formats 3D Tiles and I3S is available in the 3D Container Engineering Report [https://portal.ogc.org/files/?artifact_id=94028]; their usage and performance have been a subject of the OGC Testbed-13: 3D Tiles and I3S Interoperability and Performance ER [https://docs.ogc.org/per/17-046.html].

Chapter 5. Technical Challenges and Interoperability Issues

This Pilot addressed a number of technical challenges and interoperability issues such as:

1. Web-ready conceptual data containers, access and management of web API optimized for rendering and streaming
2. Compression for mesh-data exchanged in payloads and efficient handling of both attribute streams and geometry buffers
3. Integration of hierarchical level-of-detail approaches with 2D tile concepts and support for multiple Coordinate Reference Systems (CRS)
4. General alignment between 3D spaces and 2D tiled geospatial resources solutions
5. Integration of existing and emerging standards
6. Support for various visualization and processing pipelines, such as server- or client-based filtering, mapping, and rendering as well as analytical requirements

Chapter 6. 3D Data Container

The 3D (Data) Container (3DC), is a geo-volume information resource described by an enclosing bounding volume (2D/3D). The 3DC has an enclosing bounding box and contains at most one 3D model dataset that is relevant to that volume. The 3D model dataset is represented by reference to one or more distributions. This summary uses the terms 3DC, 3D Container, and 3D Data Container interchangeably. The reason is that the original term “3D Container” is used in the GeoVolume API, but was questioned towards the end of the Pilot due to the potential risk of term overloading.

A bounding volume is a closed volume containing the union of a set of geometric objects. The following figure illustrates typical bounding volumes (red) with enclosed objects (yellow, green, and blue cuboids).

Figure 3. Bounding volumes: Box (left), region (middle), sphere (right)

GeoVolumes follow one conceptual organization of space applied by humans, which is a nested collection of spaces where every space contains either a number of sub-spaces or a set of objects. As an example, the GeoVolume “Earth” contains a set of child GeoVolumes, one for each continent. Each continent then may have a set of child GeoVolumes for the various countries, or, if countries are irrelevant in that scenario, a number of datasets that represent the topography, rivers, and human settlements.

Thus, 3DC may describe a collection of spatially disjoint GeoVolumes or a nested, hierarchical collections of GeoVolumes. This concept is illustrated in the following figure. Here, the GeoVolume “North America” contains the two child GeoVolumes “Montreal” and “New York City”. Both are spatially disjoint. The GeoVolume “New York City” again contains a single 3D model dataset representing buildings. These buildings are available in distribution formats CityGML, 3D Tiles, and I3S.

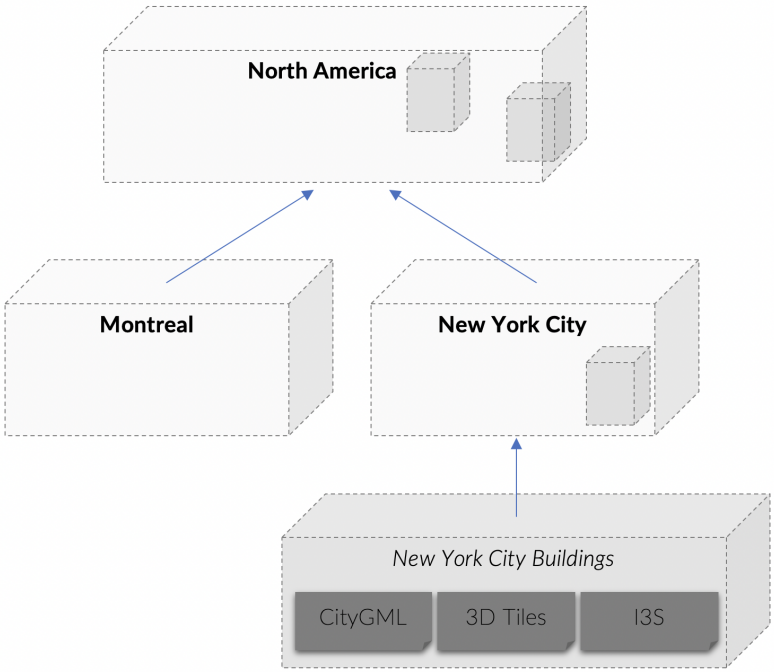


Figure 4. Hierarchical organization 3DC with varying distribution formats

The default representations of a 3D-Container are json/html information documents that define the bounding box, link to an implicit tileset scheme if applicable, and provide links to the actual content. The complete definition of the 3D Container is available in the 3D Container Engineering Report [https://portal.ogc.org/files/?artifact_id=94028]. The following figure illustrates the main elements of the container. It does not provide all details to improve readability. 3D Containers are organized in collections as described above.

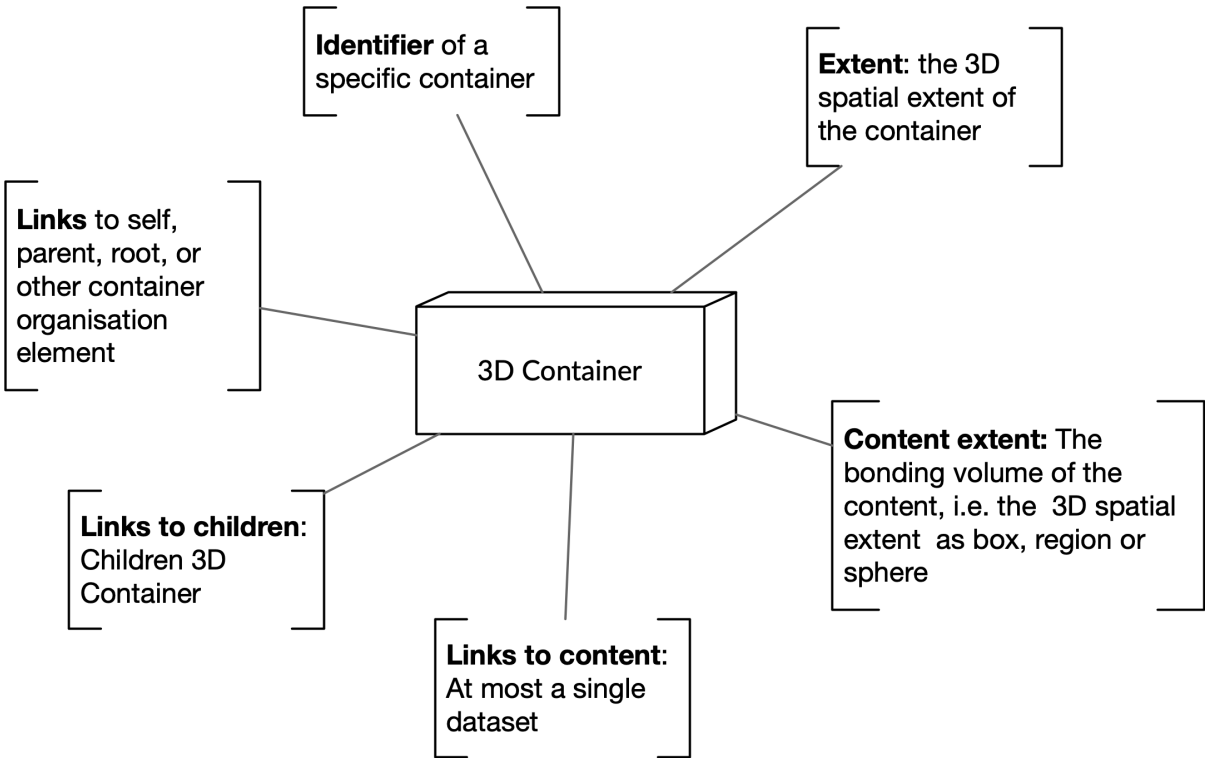


Figure 5. High level model of 3D Container (some details omitted)

What a content reference links to is dependent on content type and currently defined for the

following types:

- 3DTiles: tileset.json
- I3S: NodeIndexDocument
- CityGML: Collection document and/or logical space feature (CityModel)
- CDB: Root folder
- 2D features: Link to collection information document

Chapter 7. GeoVolumes API

The following section describes the key elements of the GeoVolumes API, an OpenAPI 3.0 REST API developed for this Pilot. The GeoVolumes API conforms to the OGC API-Common foundation resources: landing page, API definition, conformance declaration, and collections (spatial resources). It provides access to 3D resources and allows the exchange of content compliant with the 3D Container also developed in this Pilot. Table 1 describes the resources and applicable HTTP methods for the GeoVolumes API. Those resources and methods are illustrated graphically in Figure 6.

Resource	Path	HTTP method	Document reference
Landing Page	`/`	GET	Landing Page
Conformance Declaration	`/conformance`	GET	Conformance
API Definition	`/api`	GET	API/Definition
Collections	`/collections`	GET	Collections
3D Container	`/collections/{3DContainerID}`	GET	3D Container

Table 1. Overview of API resources and applicable HTTP methods

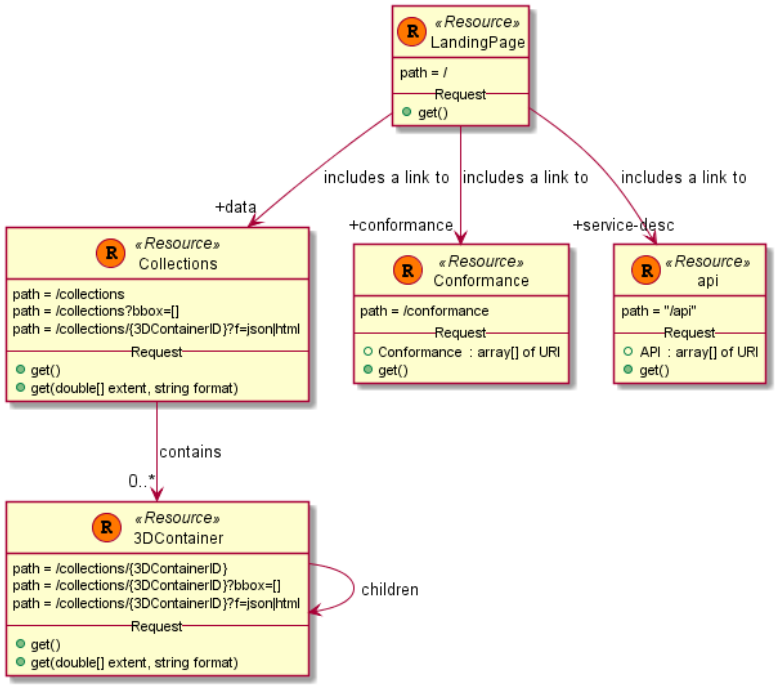


Figure 6. UML diagram of API resources

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`). The landing page requires no parameters. The HTTP `/` GET response returns landing page content in JSON.

Conformance: 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.

API/Definition: Provides the API definition that describes the capabilities of the server and which 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. The api resource requires no parameters. The HTTP `/api` GET response returns the list of URIs of API definitions in JSON.

Collections: Collections provides the information and access collection of 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 3DC that are within the area. The HTTP `/collection` GET response returns JSON containing two properties, links (link: URI, type, relationship) and 3D Data Container.

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

Chapter 8. Implementations, Experiences, and Lessons Learned by Participants

Five clients have been implemented in the course of the Pilot that interact with six implementations of the GeoVolume API and its 3D Container resources. Figure 7 shows the setup. Clients and servers have been thoroughly tested for interoperability. The connectors only illustrate the technical interoperability experiments, some are omitted to improve readability.

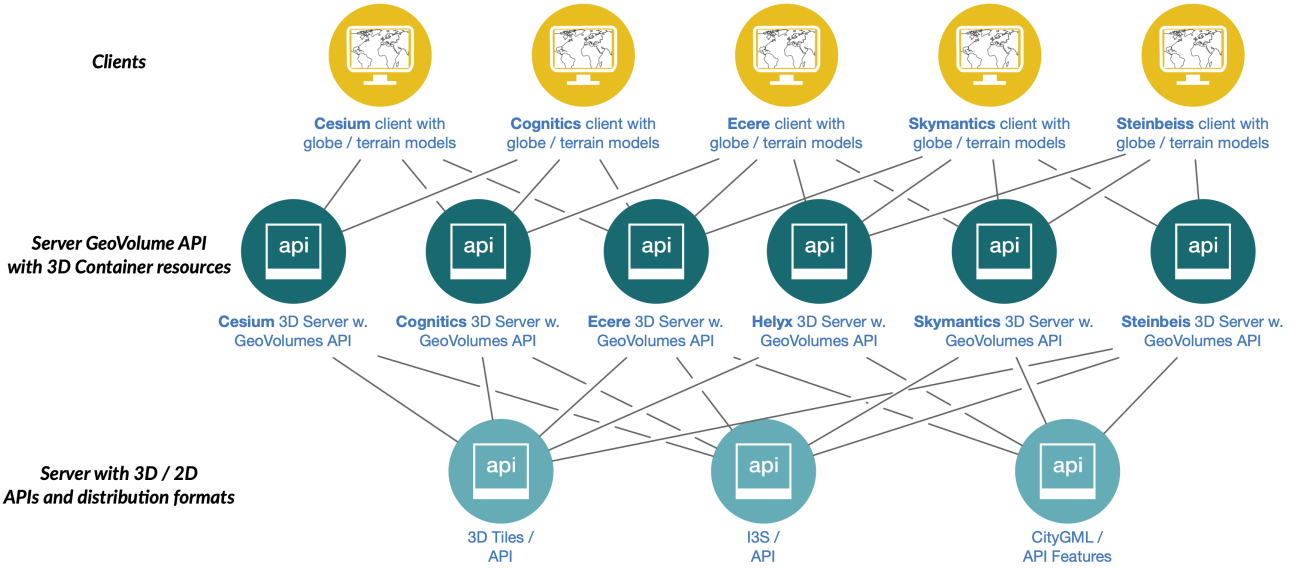


Figure 7. Client and server implementations. The connectors are exemplary only. All clients have been tested with all GeoVolume APIs.

8.1. Experiences

Participants shared the following experiences and lessons learned:

- The new API developed in this Pilot is useful for discovering a spatially-organized collection of 3D datasets, particularly where the datasets are available in multiple formats.
- It is our view that this concept is not only useful for discovering spatially-organized collections. It is also useful to be able to organize and search for datasets by theme or by other criteria. Future work could allow searches by additional criteria.
- Also, this concept is not only useful for 3D data. It's easy to imagine a dataset in GML format, for example, being made available in a wide variety of other ways as well: As an "OGC API - Features" service, as 2D vector and raster tiles using "OGC API - Tiles", and as extruded 3D volumes served in 3D Tiles and I3S. Future work could extend the API developed in this Pilot to allow all six of these formats / APIs to be discoverable via a single API, and make it clear that all are distributions of the same dataset.
- The new 3D data container format helped considering 3D Tiles and I3S as complementary representations, increasing the attractiveness for 3D apps with support for both representations. The proof of concept of the I3S-to-3DTiles tool, which allowed apps supporting only 3D Tiles representation to render I3S models, lowers the barriers for such a

move.

- The decision to implement the 3D Tiles API separated from the original 3D Server offered great flexibility and allowed to easily deploy hierarchical 3D Container federated catalogs
- We tested two main approaches – to serve data from static files (e.g. 3D Tiles or Scene Package) and to chain / federate services from third party servers. Both worked well.
- In terms of key design decisions, we chose to represent our API using both JSON responses and HTML to provide an API that was easier to navigate.
- In terms of lessons learned, as well as API interoperability challenges, there are often subsidiary challenges such as CORS headers and server / service structure (trailing slashes) that can trip up interoperability between servers and clients.
- In terms of recommendations, we found that it would be fruitful if there were further tools available to easily convert CityGML data to I3S or 3D Tiles (particularly data with textures), and that converter services to convert between the I3S and 3D Tiles were a promising tool that warrants more development.
- Our experience with implementation of the API was relatively linear. Once we aligned our API implementation to the specifications and schemas outlined in the 3D Container Recommendation, only minor modifications were required to achieve and support successful TIEs with the other servers.
- APIs can now combine 3D Tiles and I3S. This Pilot has drafted a new API that allows for the delivery of 3D Models through the combination of two different representation formats: 3D Tiles and I3S. Other storage formats, such as CityGML and CDB, have also been tested in this Pilot by some participants. This is an important step forward, as it provides two distinct advantages:
    - APIs can be more interoperable, as they can offer two different representations for the same data. If a client only supports one representation, they can still render the model.
    - For ecosystems where clients support both representations, 3D Models can be richer and more performant, by combining 3D Tiles and I3S representations to deliver a model. This allows us to use each standard for the use cases with the best fit. However, there does not seem to be a clear recommendation as to which standard is the best for each use case. This becomes a particularly steep learning curve for newcomers, as they need to learn several standards before they can make educated decisions on which formats to use for their own case.
- The hierarchical structure for 3D Containers defined in this Pilot opens the door to offer 3D Models in a natural way. But from the service architecture point of view, it also allows to deploy 3D Container federated catalogs, that is, APIs that offer 3D Containers from different sources, combining 3D Tiles and I3S, without hosting the data. This could be a natural way to separate the responsibilities for creating and hosting local 3D data (for local and regional entities) and for finding, aggregating and serving national or global 3D data (for national or global entities).
- During the Pilot there were numerous examples of rendering issues in the various clients. Some issues could be fixed during the Pilot but most remain confirmed and open. There

seems to be the need to put development teams from different 3D app vendors in touch, for example by organizing regular 3D rendering interoperability sprints to help finding and ironing out all these interoperability issues.

8.2. Implementations

8.2.1. Cesium

Cesium developed and deployed an implementation of the 3D Container and Tiles API, as well as a web-based client application to query and render 3D datasets from servers implementing the API. The Cesium client application was able to successfully integrate with the API provided by all other participants, and the Cesium server was successfully accessed by all other participants' client applications.

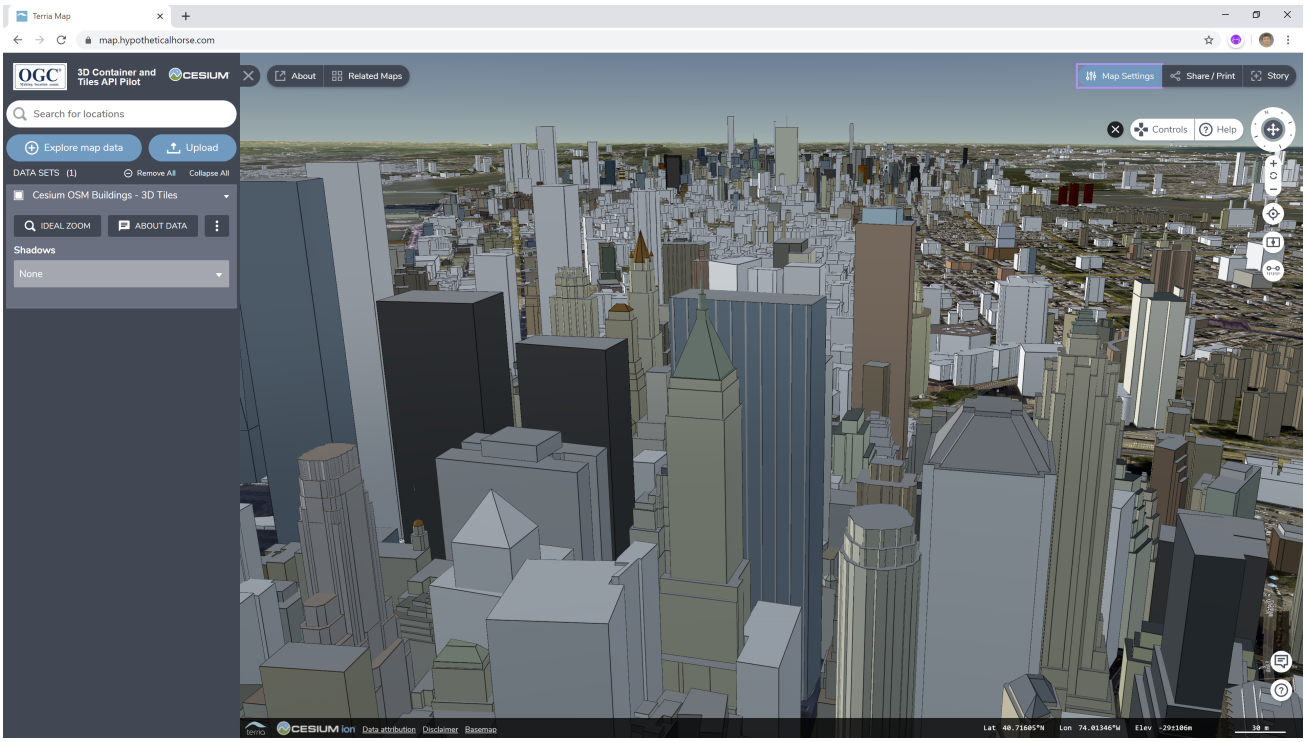


Figure 8. Screenshot of Cesium's Web-based 3D Client

The client can query GeoVolumes API endpoints and display the discovered datasets in a hierarchical catalog. The catalog interface displays the metadata for each dataset, including an inlaid map showing the data's bounding box. The user can add 3D Tiles datasets to the CesiumJS-based 3D globe. The user can also add datasets in I3S format, in which case each I3S request is passed through a service (courtesy the Terria team at CSIRO's Data61) that automatically converts each I3S resource to an equivalent 3D Tiles resource. The client can also display "groups" which use the API's bounding box search capability to restrict the view to those datasets that lie within a given bounding box.

In addition to the client, Cesium implemented a server offering the GeoVolumes API. The server supports a number of features:

- Hierarchical collections of 3D data
- Querying for 3D containers by 2D and 3D bounding boxes
- Datasets may be hosted on the API server or on a different server
- Open source code

8.2.2. Cognitics

Cognitics implemented a static Node.js 3D server providing 3DTiles and developed a client using Unity 3D to query and display 3D data from other participant servers. The implementation provides glTF content as a single container based on the request to the 3D Data Container and Tiles API. The glTF models represent the valid JSON responses through the API. The implementation fully supports embedded binary data and external binary data glTF models. Correct placement is ensured using B3DM.

The Cognitics implementation supports Unity by placing glTF models inside of a CDB environment at runtime. It also converts glTF and 3D Tiles into Unity meshes that render very quickly. The figure below shows the results of the Unity 3D client request for the "New York Buildings" collection.

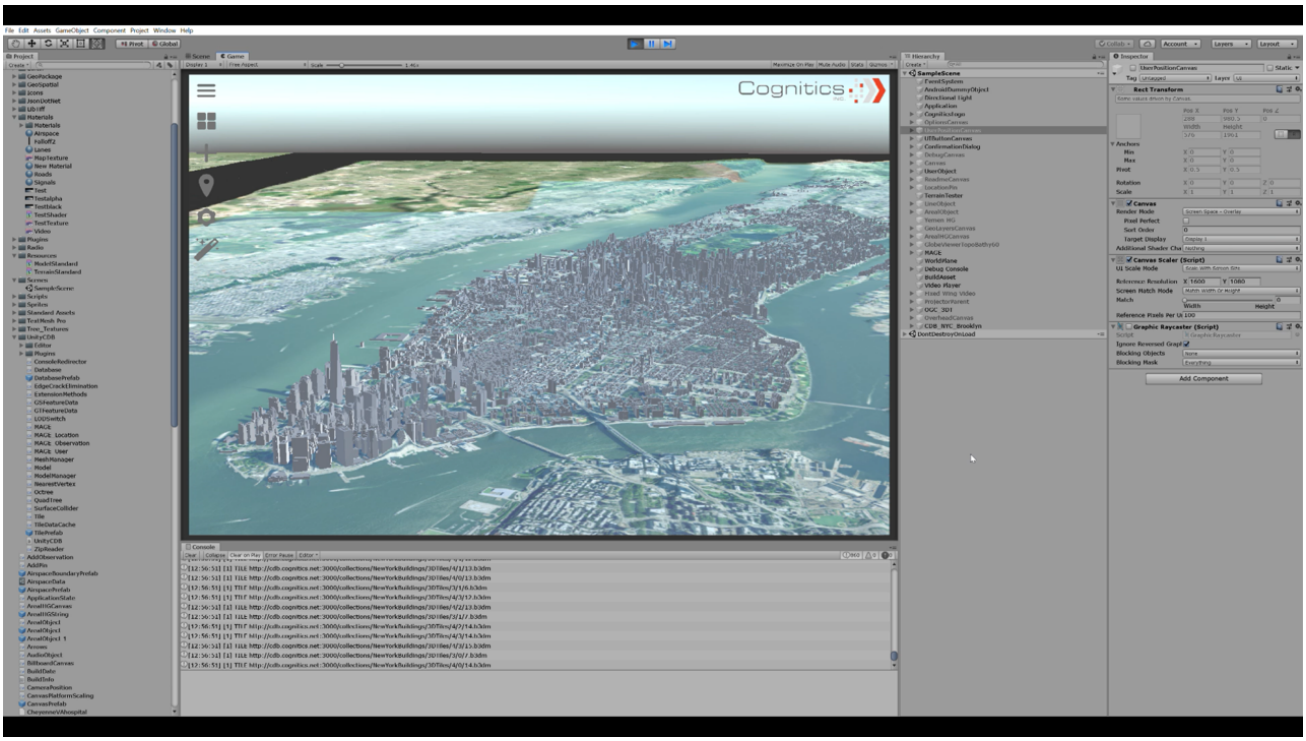


Figure 9. Screenshot of Cognitics' client application

8.2.3. Ecere

Ecere provided both server and client components based on its GNOSIS technology, integrating multiple ways to deliver and access 3D data within these OGC API components, including the Tiles API and 3D Tiles.

Ecere improved its dynamic GNOSIS Map Server with capabilities to distribute 3D data using both the Tiles API and 3D Tiles. Multi-resolution tiles for 3D data are distributed as either a 3D mesh covering a whole tile, or as tiles of vector points referencing individual 3D Models. Tiles of imagery, elevation or vector data are also supported. The tiles follow tiling schemes described by the TileMatrixSet standard. The Tiles API could also easily be extended with vertical sub-division to support highly detailed multi-stories buildings, organizing the data in a geographic space octree.

Support for distributing the data as a 3D Tiles tileset linking to batched 3D Models following a fixed tiling scheme was also developed. This capability was tested using the CesiumJS client. Future work will focus on also providing access to pre-tessellated versions of the terrain as quantized terrain mesh, as well as on the ability to load 3D datasets from CityGML sources onto the GNOSIS Map Server.

Ecere also implemented a simple static server, based on Apache, used in TIEs. The static server distributed the New York City buildings served by all participants as 3D Tiles.

Figure 10. CesiumJS client accessing Camp Pendleton data from GNOSIS Map Server as 3D Tiles generated on-the-fly

Ecere's client component, GNOSIS Cartographer, re-used existing support for common OGC API capabilities such as accessing landing pages and data layers. The client supported retrieving 3D data using both a Tiles API approach as well as 3D Tiles Bounding Volume Hierarchy tilesets.

The multi-resolution tiles accessed using the Tiles API included imagery, terrain elevation, as well as points referencing 3D Models with position and orientation information, and were all integrated within a 3D view. The GNOSIS engine tessellates a 3D mesh of the terrain on-the-fly from the elevation gridded coverage tiles.

Figure 11. Camp Pendleton CDB dataset accessed from GNOSIS Map Server using the Tiles API

Support for 3D Tiles in the latest version of Ecere's GNOSIS software was implemented during the Pilot. The Ecere client could access and visualize all data from other services being provided as batched 3D Models (.b3dm) 3D Tiles.

8.2.4. Helyx

For this Pilot Helyx provided a static 3D server built on Apache, and deployed on Microsoft Azure as a Docker image. The server was loaded with an array of different data sources for clients to test. Helyx chose a static server implementation to demonstrate and test the type of data that could be served even with a simple web server. The API was built in both JSON and HTML representations for ease of both machine and human navigation. The server demonstrated a couple of concepts, detailed below:

- **Serving a 3D Container from the local server**:
  - 3D Tiles: 3D Tiles data of New York were sourced from the Pilot partners - The quadtree version of the data was chosen, so all the data tiles were placed on level 14 of a double-headed quadtree where the Level 0 tiles are -180°→0° (Western Hemisphere) and 0°→180° (Eastern Hemisphere).
  - I3S: Building data for Montreal was sourced from the city's data portal (donnees.ville.montreal.qc.ca). An I3S scene layer package was created using Esri's CityGML Import tools, and associated 3D City Information Model. This was then exported as a Scene Layer Package, which was unzipped and served on the server. Both the buildings and the textures were served.
- **Incorporating 3rd party 3D services into the Pilot's 3D container**: As well as serving data directly from the server, the Helyx server also showed 3D data being served from 3rd party

servers. In the case of i3s, this used Esri's ArcGIS Online. In the case of 3D Tiles, Cesium's Ion server was used. These were passed through as URLs in the API content and links section.

8.2.5. Skymantics

Skymantics participation in this Pilot consisted of two different implementations: A server that offered 3D data through the 3D Tiles API and a client that queried the 3D Tiles API and rendered visually the 3D data.

Skymantics chose Hexagon's Luciad Fusion as the main 3D data server, but decided to implement the 3D Tiles API independently. This allowed for greater flexibility in the deployment, as the 3D data server was transparent for the clients. Additional servers and repositories could be added to the API as if they were in the same server. Even other 3D Tiles APIs could be connected following the same hierarchical concept that was included in the 3D Containers, much like in a federated catalog. In the Skymantics implementation, the Marseille 3D Container was hosted at Hexagon's services, and the I3S representation of the NYC buildings was borrowed from Steinbeis servers through their 3D Tiles API. The Luciad server was deployed on a virtual Linux server on Azure, and so was the 3D Tiles API.

Skymantics 3D Tiles API offered a landing page, following the structure specified in OGC API - Common. It also provided detailed information on all the 3D Containers offered by the API, including a human-readable description, spatial extent and links to the 3D Tiles data, as well as information on children 3D Containers and links to parent 3D Containers when applicable. It could offer alternate representations (I3S) if available. It was also possible to query the 3D Containers available within a specific bounding box, or even filter by available representation (3DTiles or I3S).

Skymantics developed a visual, mouse-governed client, based on Luciad RIA, to test the Pilot's 3D Tiles API implementations. The figure below shows the visualization of the Marseille 3D container in Skymantics' client application.

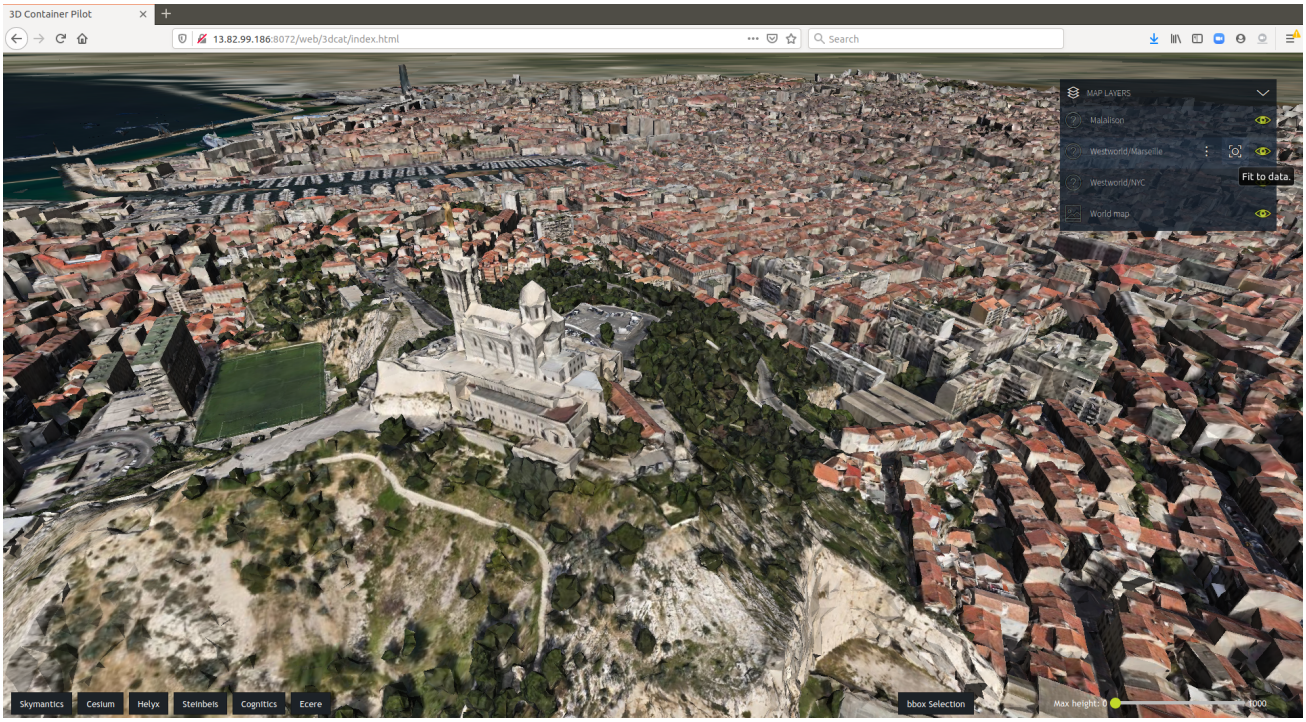


Figure 12. Screenshot of Skymantics' client application

The bottom-left corner provided direct access to all the Pilot participants' APIs. Clicking each button triggered the process to query the API landing page and explore through the 3D containers finding the available data. The top-right corner showed the map layers of the app, consisting of all the visible 3D containers found in the API. The bottom-right controller allowed to filter by bounding box.

8.2.6. Steinbeis

Steinbeis provided an API server to deliver 3D content supporting New York City buildings. The API was implemented as per the draft OGC API – Common - Part 1: Core [http://docs.opengeospatial.org/DRAFTS/19-072.html] specification, with general information on the API as well as links to the API definition, conformance classes and the 3D containers.

The Steinbeis 3D web client is implemented to interact and visualize the 3D geospatial contents from the OGC API -Tiles-3D servers. The client is developed with the CesiumJS library and ArcGIS for JavaScript library. In addition, the client can interact with the OGC 3D Portrayal Service. The figure below shows the client in action. Users can request 3D contents to the 3DPS by drawing a bounding area on the 3D map in the example area of New York City, USA, and the server will respond with the 3D contents in the requested area.

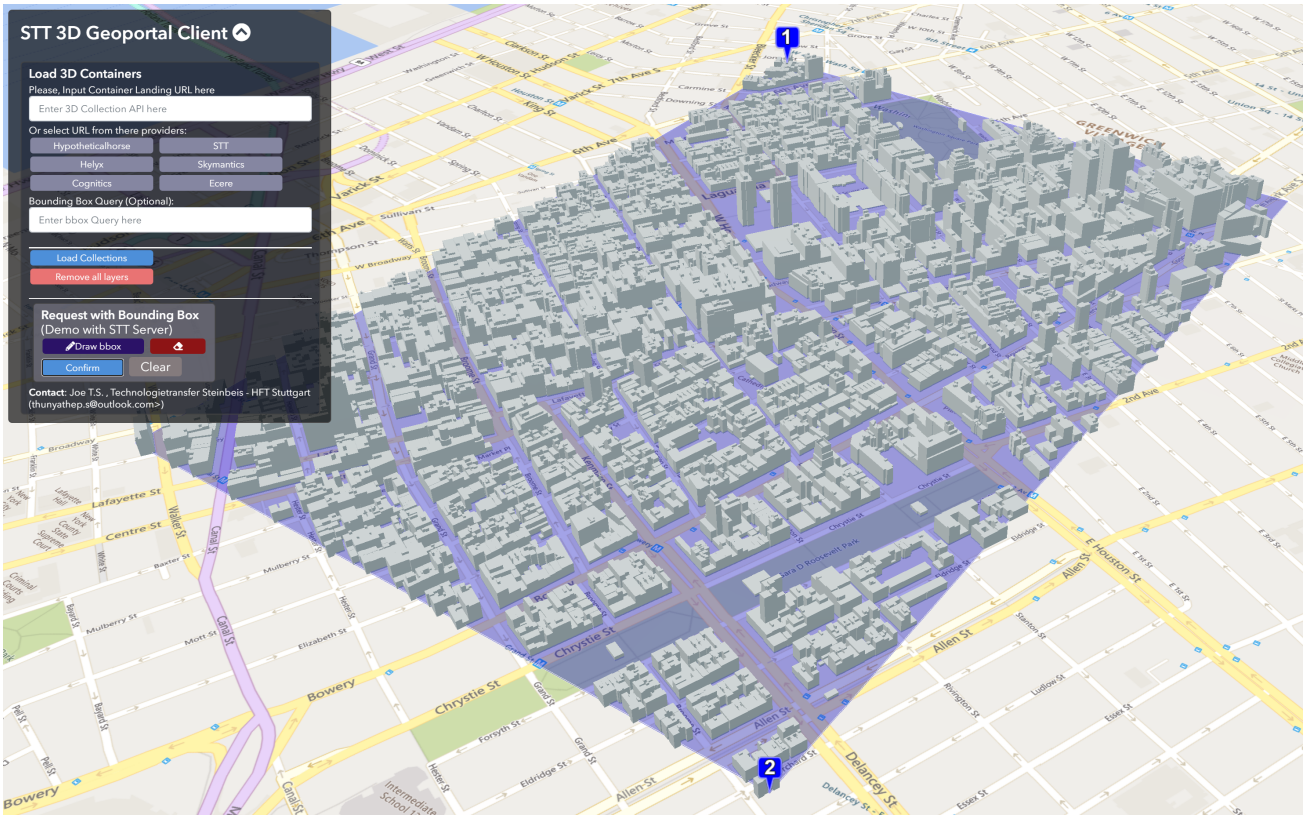


Figure 13. Screenshot of Steinbeis' client application

In this Pilot the geoportal client has been extended with support for OGC API - Features, which provide methods to query and export geospatial contents through HTTP GET requests.

Chapter 9. Commercial Potential

There has been significant development in recent years at the intersection of geospatial, 3D graphics, and modeling and simulation communities. A number of specifications, standards, and frameworks have emerged to store, visualize, and analyze 3D geospatial content. The Sponsors of the Pilot identified a need for a unified access interface and consistent conceptual understanding of tiled 3D resources to make use of diverse 3D data in various formats. Pilot participants developed a resource model - the 3DC - and a corresponding API that provides access to a variety of 3D content in multiple formats. Participants also demonstrated an implementation that allows applications working with tiled 2D geospatial resources to retrieve and position glTF 3D Models via an extension to OGC API – Features.

The draft specifications and other artifacts developed in the Pilot, as well as implementation demonstrations using various server and client configurations, provide a strong foundation for future work. The intent of this evolving body of work is to simplify access to 3D geospatial resources, to promote consistency through an open standards based approach, and ultimately to reduce implementation barriers for organizations interested in building 3D geospatial capabilities. Conducting this work through the OGC, in coordination with complimentary standards organizations like the Khronos Group, creates opportunities to tightly integrate 3D geospatial resources into the existing geospatial architecture for the web.

Chapter 10. Way Forward and Standardization

A follow-on activity, the OGC Interoperable Simulation and Gaming Sprint [https://www.ogc.org/projects/initiatives/isg-sprint], will continue to advance the draft specifications developed in the Pilot through practical exercise and testing. The Sprint event is scheduled to take place September 14-18, 2020.

Opportunities for future work include:

1. Explore performance in denied, degraded, intermittent, or low bandwidth (DDIL) environments.
    a. What mechanisms are available to deliver updated 3D content (delta updates) - and not an entire dataset - to users in the field?
    b. There are use cases in which users need to download a subset of a larger dataset to the client for offline use. Pilot participants explored the idea of a package format to contain 3D content of various types, but ultimately decided that the proposal wasn't feasible within the Pilot period. Consideration was given to GeoPackage, Scene Layer Package (SLPK), and Cesium's internal SQLite transport format (3D Tiles) as potential foundations for this notional package format.
2. Investigate best practices for organizing data within a 3DC. The OGC Interoperable Simulation and Gaming Sprint Call for Participation [https://portal.ogc.org/files/?artifact_id=94059] challenges participants to explore this area while acknowledging that solutions will vary by specific use case:
    a. Is there one bounding volume hierarchy per county, region, city, or some other geo-political boundaries?
    b. How are features (buildings, vegetation, transportation networks, etc.) structured in the data store? Are they layers in geo-political sets, or are geo-political data layers in feature sets?
3. Demonstrate integration with the OGC Sensor Things API.
4. Continue to advance integration with the modeling and simulation community. In particular, explore integration with game engines and other applications that require attribution at the individual polygon level (as opposed to feature-level attribution that is typical within 2D geospatial applications).
5. Explore methods of cataloging / traversing catalogs of 3D geospatial resources

In terms of standardization, the Pilot developed a draft API and resource model that have been proven mature enough to be introduced to OGC's Standards Program. The next steps now include the generation of a new OGC Standards Working Group (SWG), which requires the definition of the SWG charter. Once the charter is formally approved by the OGC Technical and Program Committees (TC & PC), the SWG starts with the consensus based standard development

process and eventually recommends to the Technical Committee the release of the final Standard to the public.

Appendix A: References

The following two OGC Innovation Program Engineering Reports contain the detailed experiences, lessons learned, implementation descriptions, and the draft GeoVolumes API:

- 3D Container Engineering Report [https://portal.ogc.org/files/?artifact_id=94028]
- 3D Container API Engineering Report [https://portal.ogc.org/files/?artifact_id=94029]

Appendix B: Terms and definitions

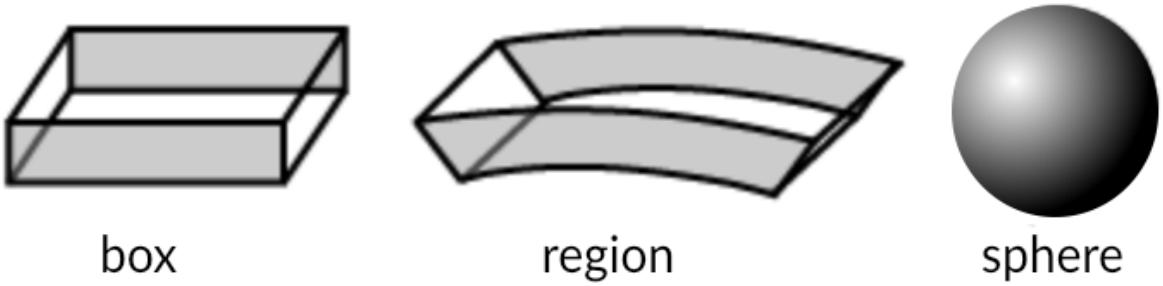
For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- *coordinate reference system*

  Coordinate system that is related to the real world by a datum term name (source: ISO 19111)

- *Bounding Volume*

  Typically a simple shape like a sphere, rectangular box, or convex hull that can simply be tested for intersection or overlap. [2]



- *YAML*

  YAML is a human friendly data serialization standard for all programming languages.

Abbreviated terms

NOTE: The abbreviated terms clause gives a list of the abbreviated terms and the symbols necessary for understanding this document. All symbols should be listed in alphabetical order. Some more frequently used abbreviated terms are provided below as examples.

- API Application Programming Interface
- CDB Common Database
- COM Component Object Model
- CORBA Common Object Request Broker Architecture
- COTS Commercial Off The Shelf
- CRS Coordinate Reference Systems
- B3DM Batched 3D Model
- BIM Building Information Model
- BVH Bounding Volume Hierarchy
- DCE Distributed Computing Environment

- DCOM Distributed Component Object Model
- glTF GL Transmission Format
- I3DM Instanced 3D Model
- IDL Interface Definition Language
- JSON JavaScript Object Notation
- MBV Minimum Bounding Volume
- MBS Minimum Bounding Sphere
- LOD Level of Detail
- OBB Oriented Bounding Box
- RS Regular Subdivision
- TMS Tile Map Services
- YAML A recursive acronym for "YAML Ain’t Markup Language"
- 3DC 3D-Container (3DC)

[2] Source: **Tiles-3D Community Standard** [https://www.ogc.org/standards/3DTiles] 3D Tiles Community Standard 1.0 (18-053r2)

Appendix C: Revision History

NOTE

Example History (Delete this note).

replace below entries as needed

Date	Editor	Release	Primary clauses modified	Descriptions
July 03, 2020	I. Simonis	0.1	all	initial version
July 14,2020	I. Simonis	0.2	all	first draft
July 15,2020	I. Simonis	1.0	all	final version
July 15,2020	I. Simonis	1.1	all	final version
July 16,2020	I. Simonis	1.2	8.1	editorial changes
September 30,2020	G. Hobona	1.3	multiple	final staff review

Table 2. Revision History