3D Data Container Engineering Report

Publication Date: YYYY-MM-DD

Approval Date: YYYY-MM-DD

Submission Date: 2020-07-15

Reference number of this document: OGC 20-029

Reference URL for this document: http://www.opengis.net/doc/PER/20-029

Category: OGC Public Engineering Report

Editors: Timothy Miller, Gil Trenum, Josh Lieberman

Title: 3D Data Container Engineering Report

OGC Public Engineering Report

Table of Contents

Chapter 1. Subject

This OGC Engineering Report documents the goals, activities, experiences, and outcomes of the 3D Data Container and Tiles API Pilot. Participants in the Pilot cooperatively defined a GeoVolume (3D Geospatial Volume) resource and developed a GeoVolumes API based on the concept to provide access to different 2D and 3D geospatial dataset distributions organized by region of interest. Multiple client and server implementations of the GeoVolumes API successfully carried out technology interchange experiments that demonstrated the value of the API for improving interoperability between 3D geospatial data formats.

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

Chapter 2. Executive Summary

The goal of the OGC 3D Data Container and Tiles API Pilot was to:

- Explore an integrated suite of draft specifications for 3D geospatial resources compatible with existing OGC 3D delivery standards in order to support smooth transitions between 2D and 3D environments;
- Allow applications working with 2D tile resources to get 3D tiled resources; and,
- Enable 3D bounding volumes to support multiple data models, datasets, and distributions.

To achieve these goals, the Pilot participants developed a draft GeoVolume resource (originally the 3D Container resource) and GeoVolumes API providing browse and query access to 3D geospatial data for streamed data delivery by means of nested geospatial volume container resources. The 3D data resources supported by this API include feature geometries, feature attribute values, elevation models, texture data, and so forth. The API provides both link-follow and bbox query methods of access to 2D and 3D resources in a manner independent of the structure of the underlying data store, supporting multiple standard geospatial distribution formats 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].

A number of data server and client implementations were developed in the course of the Pilot in order to test interoperable data delivery via the draft API. The GeoVolumes API developed by the Pilot is described using the OpenAPI 3.0 [https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.2.md] 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 such as landing page, API definition, conformance declaration, and collections information. The Pilot developed and tested the GeoVolumes API in order to advance open standard-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.

The following requirements were identified in the Call for Participation (CFP) of the 3D Data Container and Tiles API Pilot:

1. Provide API access to tiled and un-tiled 3D resources.
2. Allow exchange of content organized according to 3D Container resources.
3. Align with existing and emerging OGC APIs, standards, and candidate standards (e.g. OGC API - Features, OGC API - Common, 3D Tiles Community Standard, I3S Community Standard).

The present draft GeoVolumes API specification includes the six basic and extended forms of functionality implemented and tested during the Pilot. The experiences developed in the course of the Pilot suggest that further work is needed to extend the API for additional data types and nested volume schemes. Future work should also address the integration of the GeoVolumes API with other OGC API building blocks to provide seamless interaction from navigating volumes of interest to accessing specific dataset distributions.

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
Josh Lieberman	OGC	Editor
Terry Idol	OGC	Contributor
Rob Jones	Helyx	Contributor
Matthew Knight	Helyx	Contributor
Anneley Hadland	Helyx	Contributor
Tamrat Belayneh	Esri	Contributor
Jérôme Jacovella-St-Louis	Ecere	Contributor
Aaron Brinton	Cognitics	Contributor
Michala Hill	Cognitics	Contributor
Ignacio Correas	Skymantics	Contributor
Preston Rodrigues	Steinbeis	Contributor
Kevin Ring	Cesium	Contributor
Volker Coors	Steinbeis	Contributor
Thunyathep Santhanavanich	Steinbeis	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. References

The following normative documents are referenced in this document.

- OGC: OGC 18-053r2, 3D Tiles Community Standard 1.0 [https://www.ogc.org/standards/3DTiles]
- OGC: OGC 17-014r5, OGC Indexed 3d Scene Layer (I3S) and Scene Layer Package Format Specification [http://www.ogc.org/standards/i3s]
- OGC: OGC 17-069r3, OGC API - Features - Part 1: Core Standard [http://docs.opengeospatial.org/is/17-069r3/17-069r3.html]

Chapter 4. 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.

● *3D-Container*

The most general definition of a 3D-Container (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 3D-Containers (children) whose bounding volumes are fully contained by the parent container's bounding volume.

● *coordinate reference system*

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

● *portrayal*

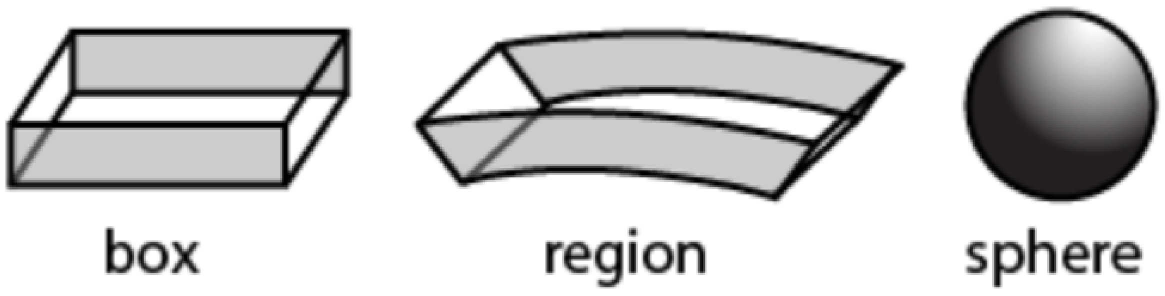
Presentation of information to humans (source: ISO 19117)

● *LiDAR*

Light Detection and Ranging — a common method for acquiring point clouds through aerial, terrestrial, and mobile acquisition methods.

● *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.

4.1. Abbreviated terms

- 3DC 3D-Container

- 3DGV 3D GeoVolume
- 3DPS 3D Portrayal Service
- API Application Programming Interface
- B3DM Batched 3D Model
- BIM Building Information Model
- BVH Bounding Volume Hierarchy
- CDB Common Database
- CRS Coordinate Reference System
- EPSG European Petroleum Survey Group
- glTF GL Transmission Format
- I3DM Instanced 3D Model
- I3S Indexed 3D Scene Layer
- IDL Interface Definition Language
- JSON JavaScript Object Notation
- LOD Level of Detail
- MBS Minimum Bounding Sphere
- MBV Minimum Bounding Volume
- OBB Oriented Bounding Box
- RS Regular Subdivision
- TIEs Technology Integration Experiments
- TMS Tile Map Services
- WFS Web Feature Service
- WMS Web Map Service
- WMTS Web Map Tile Service
- YAML A recursive acronym for "YAML Ain’t Markup Language"

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

Chapter 5. Overview

Section 6 discusses the existing standards and previous work relevant to the Pilot.

Section 7 presents the server and data architecture developed in this Pilot.

Section 8 presents multiple use cases: (1) API user, client, and (2) server use case and I3S To 3D-Tiles converter use case.

Section 9 presents the Pilot participants client and server 3D Container implementations.

Section 10 presents the Technology Integration Experiments (TIEs) results for the Pilot.

Section 11 discusses Pilot recommendations going forward.

Section 12 provides a summary of the main findings and discusses server and data architecture developed in this Pilot.

Chapter 6. Existing and Relevant Standards

6.1. Background

This OGC Pilot builds on previous work from other OGC Innovation Program initiatives as well as OGC Standards Program efforts. The following documents serve as foundational references for this Pilot activity. Some items listed below are not yet released to the public. Draft versions of these documents have been made available. The final versions of these documents may change from the currently provided versions. Participants are advised to check the OGC website for updates on these documents.

6.2. Existing Standards/Previous Work

- 3D Tiles Community Standard 1.0 (18-053r2) - The OGC 3D Tiles Community Standard specifies a model and encoding for streaming and rendering massive 3D geospatial content such as Photogrammetry, 3D Buildings, BIM/CAD, Instanced Features, and Point Clouds. The community standard defines a hierarchical data structure and a set of tile formats which deliver content that can be rendered. Of specific interest to this initiative is glTF [https://www.khronos.org/gltf/]. The standard document also describes 3D Tile Styles, a declarative styling specification which may be applied to tilesets.

- OGC Indexed 3D Scene Layer (I3S) and Scene Layer Package Format Specification (OGC 17-014r7) - An OGC Community standard that specifies a model and encoding format for the transmission of 3D content as well as a persistence model for Scene Layers. A Scene Layer is a container for arbitrarily large amounts of heterogeneously distributed 3D geographic content, supporting coordinate reference systems and height models in conjunction with a rich set of layer types.

- OGC Testbed 13: 3D Tiles and I3S Interoperability and Performance Engineering Report (OGC 17-046) - The report captures the lessons learned from prototyping interoperability of 3D Tiles and I3S using: a 3D Portrayal Service, performance studies of 3D tiling algorithms, and a proof-of-concept of the use of 3D Tiles and I3S as data delivery formats for the OGC 3D Portrayal Service interface standard.

- OGC 3D Portrayal Service 1.0 Standard (OGC 15-001r4) - The OGC standard specifies how geospatial 3D content is described, selected, and delivered. The standard provides a framework to determine whether 3D content is interoperable at the content representation level.

- OpenAPI (v3.0) - OpenAPI is a freely-available API description framework that provides a developer with programmatic access to a software application or service. These APIs use sets of technologies that enable websites and/or client applications to interact with each other by using REST, SOAP, JavaScript, and other web technologies. These APIs have allowed web communities to create an open architecture for sharing content and data between communities and applications. A very typical application for an OpenAPI implementation is to access data such as: Tweets, geolocation(s), maps, stock quotes, weather sensors, and so forth. Since 2016 OGC members and staff have actively been investigating OpenAPI (and its commercial equivalent, Swagger). OGC Members and staff recognized that the existing OGC Web Service Standards (OWS) were in effect web APIs, but that modernizing how they provide content via the web required a fundamental change in underlying design. Two documents further advanced the idea: 1) the OGC Open Geospatial APIs White Paper and 2) the Spatial Data on the Web Best Practices, jointly developed by the OGC and the W3C memberships. These documents highlighted how geospatial data should be more native to the architecture of the web. Further, OGC staff worked on "implementer friendly" views of OGC standards and experimented with an OpenAPI definition for the Web Map Tile Service (WMTS) and became aware of work by the United Kingdom Hydrographic Office (UKHO) to publish OpenAPI definitions for the OGC Web Map Service (WMS) of Electronic Navigational Charts.

- OGC API - Features - Part 1: Core standard (OGC 17-069r3) - This OGC standard provides API building blocks to create, modify and query features on the Web. This standard specifies the fundamental API building blocks for interacting with features. The spatial data community uses the term 'feature' for things in the real world that are of interest. The standard is part of the OGC API family of standards. OGC API standards define modular API building blocks to spatially enable Web APIs in a consistent way. Most recently, the OGC Web Feature Service (WFS) and Filter Encoding Service (FES) Standards Working Group rebuilt the WFS standard to the new OGC API - Features standard with an integrated OpenAPI definition as core to describe how to build against the standard. The patterns used to define the OGC API - Features standard are being used to advance other OGC standards as OpenAPI definitions. This work can be reviewed on the OGC API - Features standard webpage.

- OGC Testbed-15: Maps and Tiles API Engineering Report (OGC 19-069) - This Engineering Report was developed as part of the Testbed-15 initiative [1]. It is not a standard, but reflects the latest work on Web APIs to access and manage maps and 2D tiled data.

- OGC Testbed-15: Styles API Engineering Report (OGC 19-010r2) - This Engineering Report was developed as part of the Testbed-15 initiative. It is not a standard, but reflects the latest work done in OGC on styles for 2D data. The Styles API is a Web API that enables map servers and clients as well as visual style editors to manage and fetch styles [2]. The API is consistent with the emerging OGC API family of standards. The API complements the Features, Maps and Tiles APIs and builds on the conceptual model for styles developed in OGC Testbed-15. This report specifies the API using OpenAPI, specifies how the same styles should be represented in a GeoPackage and documents the lessons learned during the implementation.

- OGC® Routing Pilot ER (OGC 19-041r3) - This Engineering Report was produced by the OGC Routing Pilot. The Pilot assessed an abstract baseline suite of functions, capabilities, and encodings to address a common standard interface for network routing functionality [3]. This included guidance for extending the Routing API to account for various routing data models and support for network routing engine configuration via the Routing API. The engineering report is not an OGC standard, but it was delivered to the OGC Standards Program for further consideration.

Chapter 7. Architecture

This section describes the architectural context of the Pilot as well as relevant architectural perspectives on the software designed and implemented to demonstrate solutions to Pilot requirements.

7.1. Enterprise architecture

The high-level architecture of the Pilot scenario is based on the needs of various types of entities and/or systems ('A'), ('B'), and ('C') in a distributed enterprise as illustrated in Figure 1. The three systems differ in the volume of data that can be stored and processed, the available bandwidth for data transport between entities, the multiplicity of supported analytics, and the offered data products. Despite these differences, the goal of this Pilot was to develop a model that allows offering, discovering, requesting, and processing data at each entity using a common API on top of a single organizational model of 3D geospatial data in space. At the same time, this common API should leverage 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 that users can work with 3D geospatial datasets by means of the optimal distribution format and interaction method for their specific application task.

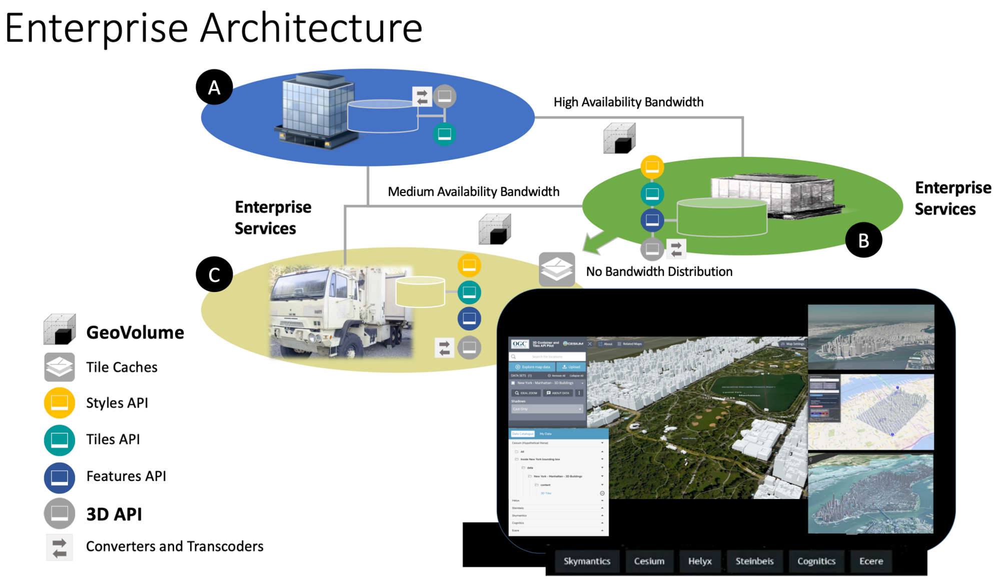


Figure 1. Enterprise architecture for exchange of 3D datasets

High capacity data centers ('A') have 3D geodata available for broad regions or on a global scale. Data can be made available in multiple formats 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 GeoVolumes API as developed in this Pilot. 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 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 capacity data centers make use of the specific space-centric indexing scheme that is the fundamental idea of a GeoVolume/3D Container resource and the corresponding GeoVolume API offered by ('A'). The indexing scheme is illustrated further below. The scheme allows organizing geodata according to the human conceptual model of space. That is, the scheme makes data available that corresponds to cognitive entities such as a country with its cities and cities with its buildings, streets, or underground infrastructure. Following the same conceptual model and distribution options as the high capacity data centers, medium capacity data centers in turn can make their products available via the GeoVolume API.

The high-level architecture defines a third enterprise entity, low capacity field operations ('C'). These are connected at various bandwidths including sporadically or completely 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 geodata distributions 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 variable-resolution scene graphs delivered to client applications and accompanying fine-grained element access.

7.2. Service architecture

The architecture of server and client components implemented in the Pilot is shown in Figure 2. Each 3D / Globe client component is able to access multiple 3D model dataset distributions in multiple formats by means of their characteristic API components through multiple 3D servers each implementing the GeoVolumes API as an overall access interface.

Service Architecture

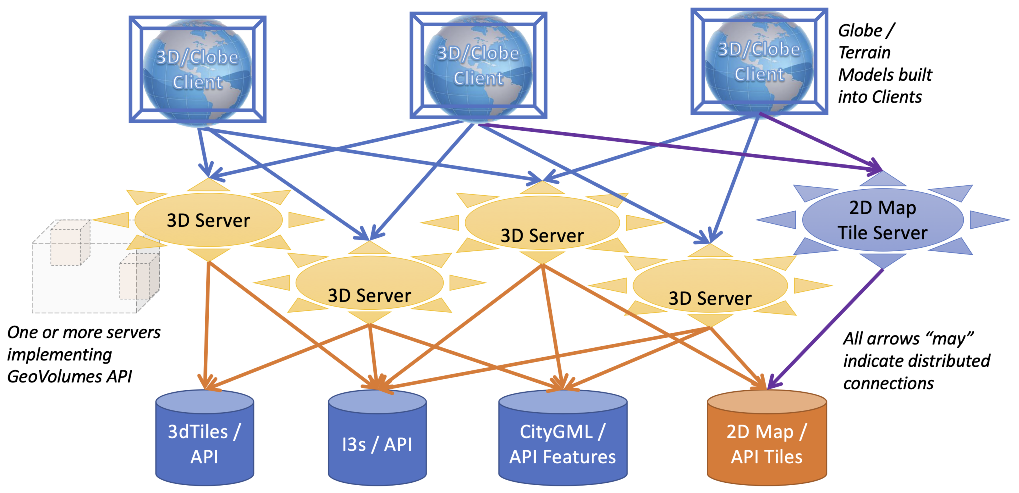


Figure 2. Service architecture implemented during the pilot

For the purposes of the Pilot, clients then visualized the returned geodata in the context of a 3D globe rendering either built into the client or assembled using 2D tiles fetched separately from the GeoVolumes API, using for example a 2D Map Tile Server. 2D tiles could also be accessed through the GeoVolumes API. However, this capability was not implemented in the Pilot.

7.3. Resource architecture

The organization of geospatial volumes in the GeoVolumes API is extremely flexible and can be arranged according to the needs of each service provider, as shown in Figure 3. GeoVolumes can be organized into a flat list of volumes having no particular relation with each other, but arguably are more representative of geographic perception when arranged into a GeoVolume hierarchy that organizes datasets according to theme and resolution.

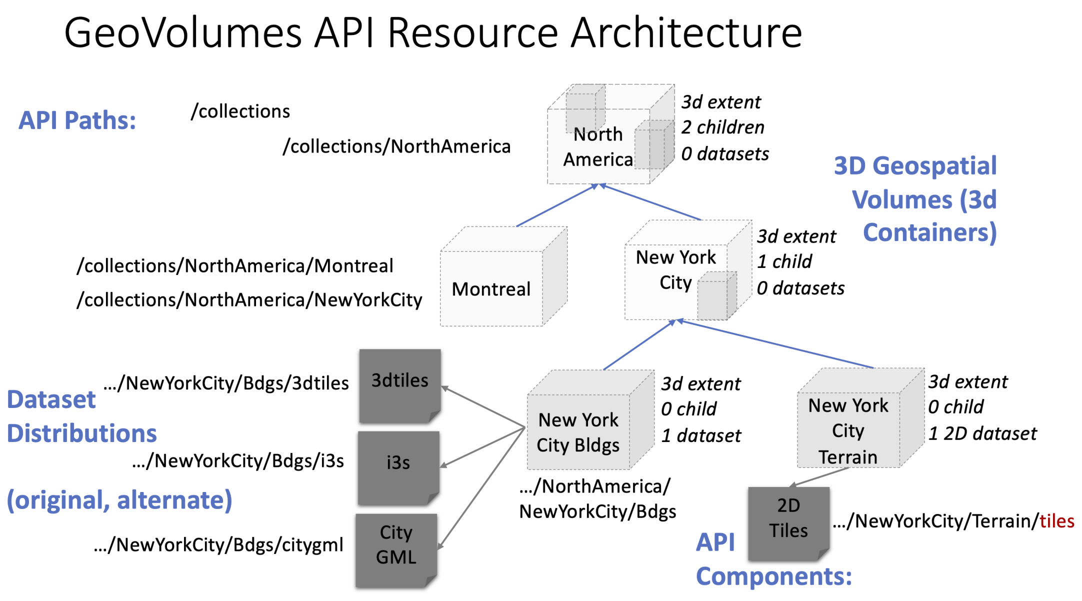


Figure 3. Resource architecture for a 3dgeovolumes hierarchy (example)

In this sort of scheme, each GeoVolume may have one or more children volumes whose extents may themselves overlap but that in aggregate are completely contained in the parent volume extent, a property known as "spatial coherence". Each GeoVolume also "contains" (references and describes the extent of) at most one dataset as its "contents", but may provide links to multiple distributions of that dataset in different formats, e.g. 3D Tiles or I3S.

Figure 3 also illustrates that a GeoVolume can contain not just an explicitly 3D dataset but also a 2D tiles or features dataset as long as the elevation at which such data should be situated (e.g. ground level) falls within the elevation bounds of the GeoVolume extent

7.4. API Design

The development and design of the architecture for the Pilot utilized the integrated API development platform of SwaggerHub [https://swagger.io/], providing the tools to develop and document the OpenAPI definitions for the resources and API implemented in this pilot. The summary documentation in this section was generated largely from the YAML definition file created in SwaggerHub. A complete draft specification for the GeoVolume API may be found in the accompanying OGC API 3D Tiles Engineering Report. Within this section, the API resources and applicable HTTP methods are described in the Server Architecture section and the associated schema representation is described in Data Architecture section.

NOTE

While the API design and implementations developed during the Pilot used the term *3D*

Container, future work will use the equivalent term *GeoVolume*.

7.4.1. Landing Page "/"

The API entry point is the Landing Page (path `/`).

Description

The Landing Page provides links to the API/Definition (path `/api`), the Conformance declaration (path `/conformance`), and the Collections (path `/collections`). The collection information resource is extended from the draft OGC API - Common and OGC API - Features definition to form 3D Containers (`/collections/{3DContainerID}`).

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. 3D Container - overview of resources and applicable HTTP methods

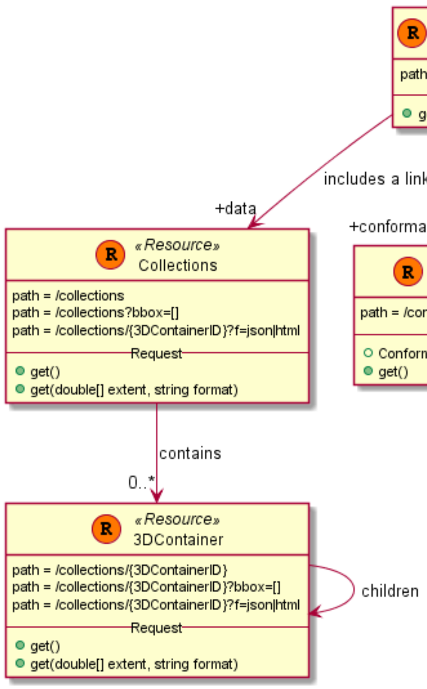


Figure 4. 3D Container - of resources and applicable HTTP methods

7.4.2. API Definition "/api"

`GET /api`

Description

Provides the API definition that describes the capabilities of the server. This definition can be used by developers to understand the API, by software clients to connect to the server, or by development tools to support the implementation of servers and clients. The "api" resource requires no parameters. The HTTP `/api` GET response returns the list of URIs of API definitions in JSON.

7.4.3. Conformance "/conformance"

`GET /conformance`

Description

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.

7.4.4. Collections "/collections"

`GET /collections`

Description

Collections provides the information and access to the collection of 3D containers. The collection resource accepts 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 `/collections` GET response returns JSON containing two properties, links (link: URI, type, relationship) and 3D-Container.

7.4.5. 3D-Container "/collections/{3d-containerID}"

`GET /collections/{3d-containerID}`

Description

The collection resource supports 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/{3DContainerID}` GET response returns JSON representing the 3D-Container.

7.5. Data Architecture

The data architecture can be separated into three parts:

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

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 model 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.

Table 2. 3D Container schema overview.

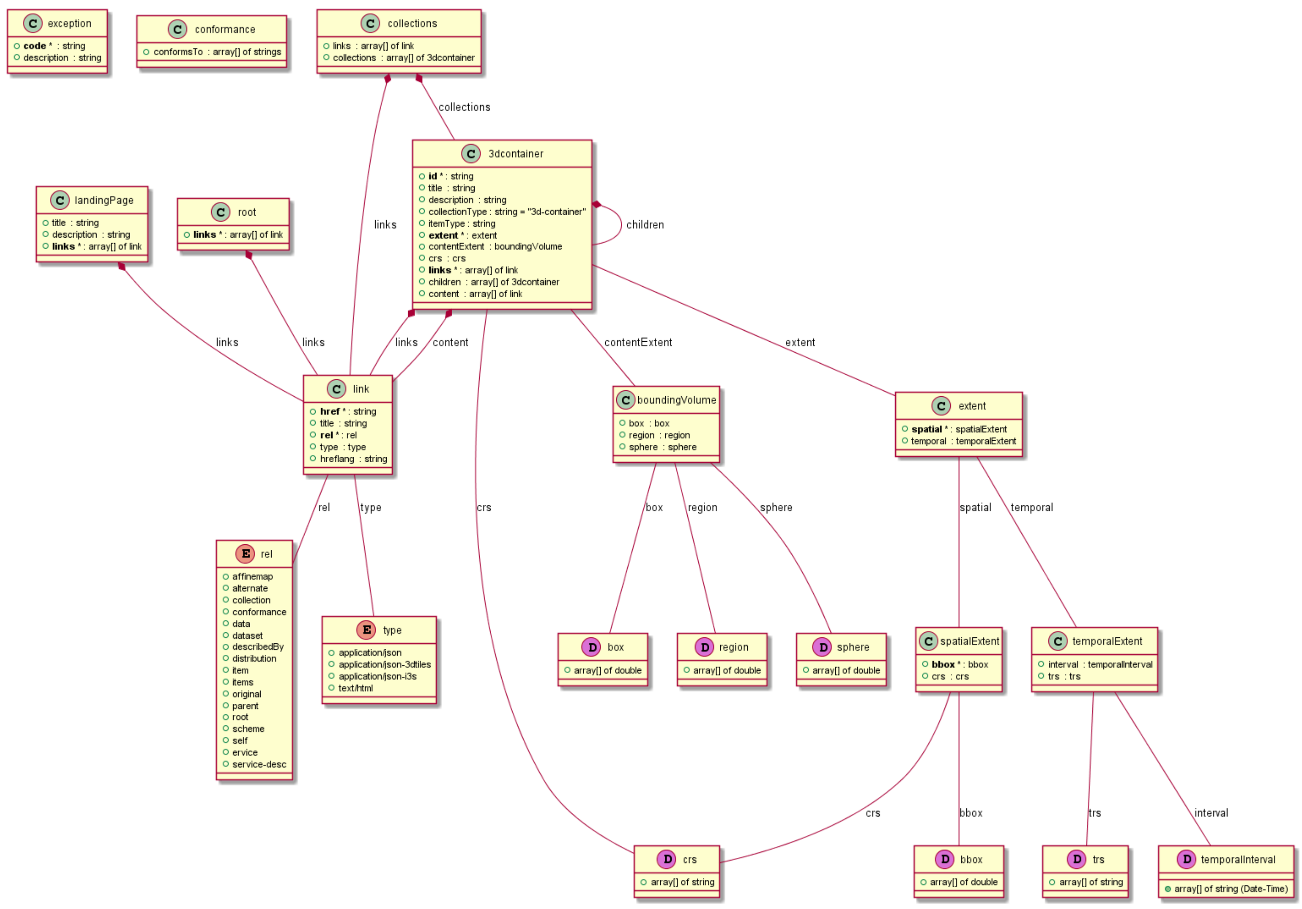


Figure 5. 3D-Container Model

7.5.1. 3D-Container

The most general definition of a 3D-Container (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 3D-Containers (children) whose bounding volumes are fully contained by the parent container's bounding volume. The default representations of a 3D-Container 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 consensus of the Pilot participants was to extend the collection resource and corresponding collection information document schema 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 3D-Containers 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.

What a content reference links to is dependent on content type and TBD for some types:

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

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
collectionType		String	Property with value "3d-container" is required.
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

Field Name	Required	Type	Description
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 a 3D-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" property 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 • CityGML: Collection document and/or logical space feature (CityModel) • CDB: Root folder • 2D features: link to collection information document

Table 3. 3D-Container

7.5.2. 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#31]

Field Name	Required	Type	Description
box		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 4. Bounding Volume

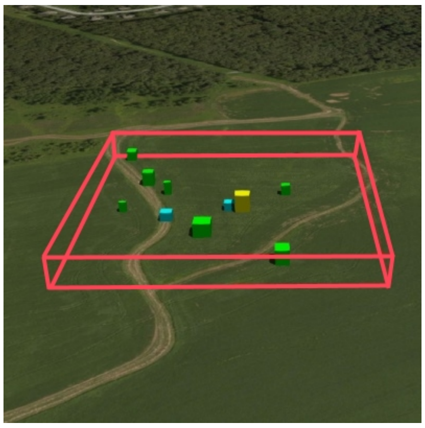


Figure 6. Bounding box

Bounding Box YAML:

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

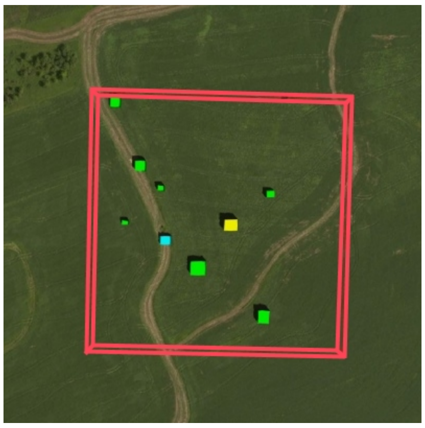


Figure 7. Region

Bounding Region YAML:

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

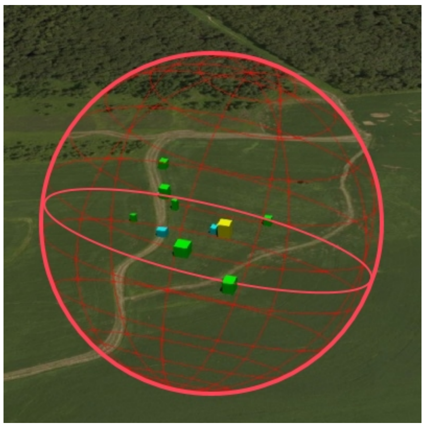


Figure 8. Sphere

Bounding Sphere YAML:

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

Source: 3D-Tiles Bounding volumes [http://docs.opengeospatial.org/cs/18-053r2/18-053r2.html#31]

7.5.3. Coordinate Reference System (CRS)

This is the CRS of the coordinates in the spatial extent (property 'bbox'). The default reference system is WGS 84 longitude/latitude [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.

7.5.4. 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.

Field Name	Required	Type	Description
spatial		spatialExtent	
temporal		temporalExtent	

Table 5. Extent

7.5.5. Link

Link to content.

Field Name	Required	Type	Description
href	X	String	The URI of the link
title		String	
rel	X	rel	Link Relationship
type		type	
hreflang		String	

Table 6. Link

7.5.6. 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#_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.

7.5.7. Spatial Extent

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

Field Name	Required	Type	Description
bbox	X	List of [array]	One or more bounding boxes that describe the spatial extent of the dataset. 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.

Table 7. Spatial Extent

7.5.8. Temporal Extent

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

Field Name	Required	Type	Description
interval		List of [array] [Date-Time]	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		TRS	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 enum values.

Table 8. Temporal Extent

7.5.9. 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"

7.5.10. Content Type

Content type enumeration examples:

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

Chapter 8. API Use Cases

The following section describes the actions for two use cases:

1. Typical user/client/server sequence for accessing a 3D Container.
2. I3S To 3D-Tiles Converter for accessing 3D content.

8.1. API user/client/server Use Case

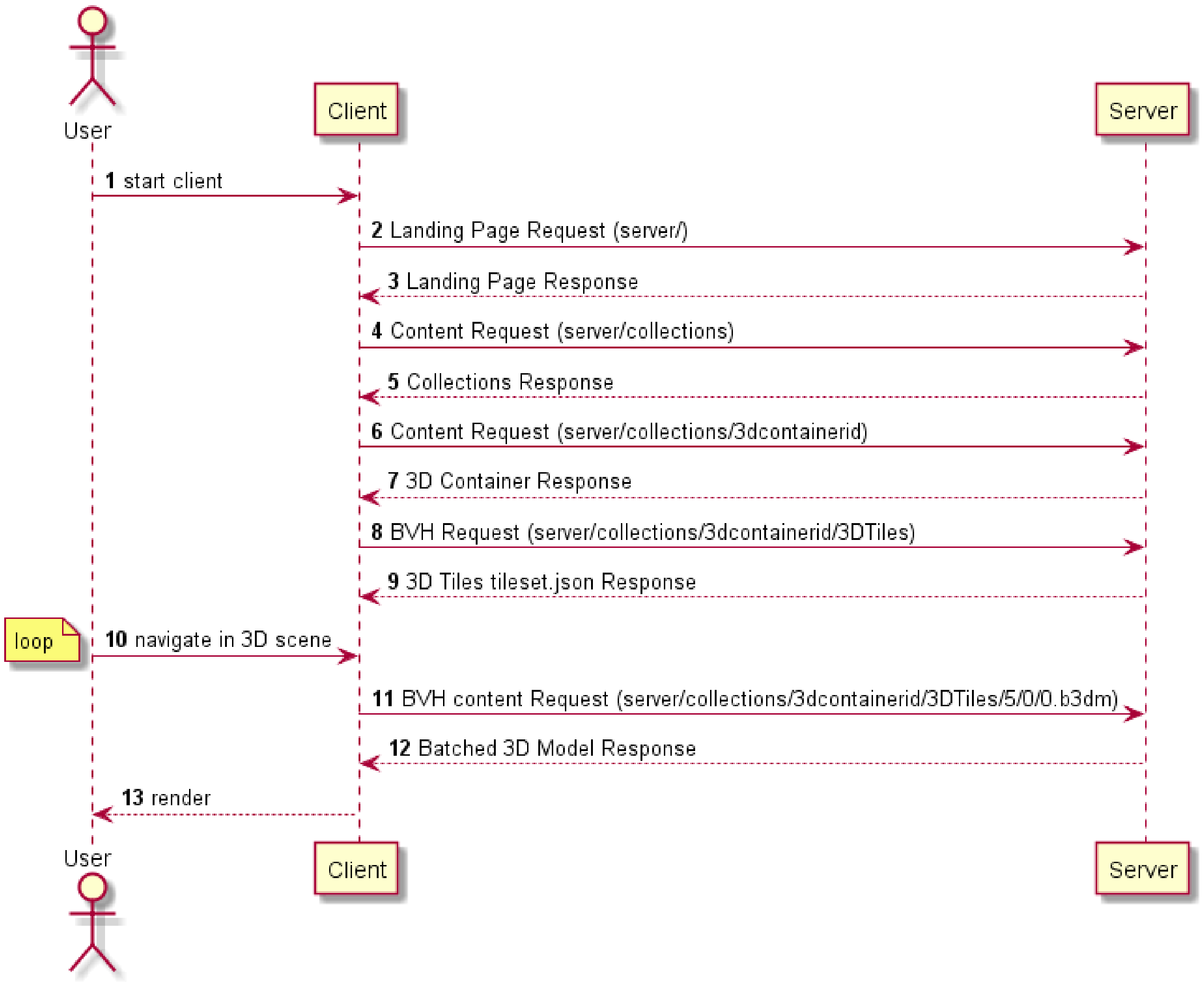
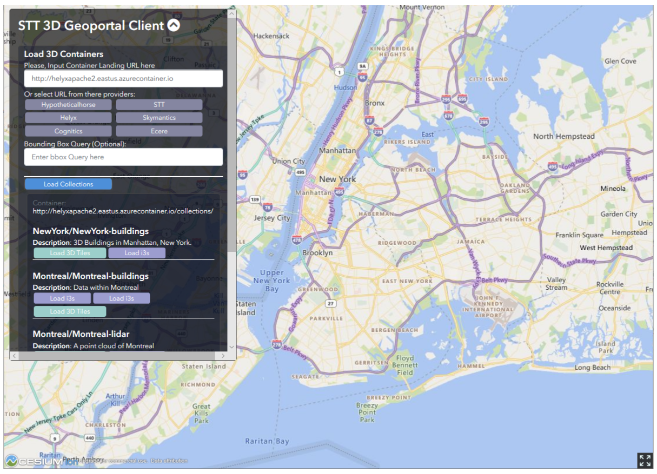

User
Client
Server
1 start client
2 Landing Page Request (server/)
3 Landing Page Response
4 Content Request (server/collections)
5 Collections Response
6 Content Request (server/collections/3dcontainerid)
7 3D Container Response
8 BVH Request (server/collections/3dcontainerid/3DTiles)
9 3D Tiles tileset.json Response
loop
10 navigate in 3D scene
11 BVH content Request (server/collections/3dcontainerid/3DTiles/5/0/0.b3dm)
12 Batched 3D Model Response
13 render
User
Client
Server

Figure 9. API Sequence Diagram

The following is the user/client/server sequence for accessing a 3D Container:

1. User starts client, selects provider, and loads collections.


STT 3D Geoportal Client

Load 3D Containers

Please, Input Container Landing URL here

http://helyxapache2.eastus.azurecontainer.io

Or select URL from there providers:

Bounding Box Query (Optional):

Enter bbox Query here

Container:
http://helyxapache2.eastus.azurecontainer.io/collections/

NewYork/NewYork-buildings

Description: 3D Buildings in Manhattan, New York.

Montreal/Montreal-buildings

Description: Data within Montreal

Montreal/Montreal-lidar

Description: A point cloud of Montreal

Figure 10. User loads collections

2. Client Landing Page Request. Client requests the landing page from server. Client landing page request: http://helyxapache2.eastus.azurecontainer.io/

3. Server Landing Page Response. The server provides the following landing page response:

```
{
    "title": "Pilot 3D Container API",
    "description": "A pilot of an API for 3D containers and tiles.",
    "links": [
        {service-desc...},
        {conformance...},
        {
            "type": "application/json",
            "title": "Collections",
            "href":
"http://helyxapache2.eastus.azurecontainer.io/collections/",
            "rel": "data"
        }
    ]
}
```

4. Client Content Request. Client parses the landing page for data link relationship and requests the collections from the server. The client can optionally provide a bounding box query string to filter the contents from the server and format type. Client collections requests: http://helyxapache2.eastus.azurecontainer.io/collections http://helyxapache2.eastus.azurecontainer.io/collections?bbox=-75,40.,-12,-73,41,600&f=json

5. Server Collections Response. The server processes the request returning a collections object, containing an array of links and array of 3D containers. The response may contain multiple containers/container IDs in a flat or hierarchical organization.

6. Client Content Request. The client parses the collections response within the 3d container. Links to the 3D Tiles content maybe located in (1) links, (2) content, or (3) children properties. From the 3D container information, the client can process either a flat list or hierarchical list of 3D containers. The organization depends on the use of the children object within the 3D container. Client container request: http://helyxapache2.eastus.azurecontainer.io/collections/NewYork/NewYork-buildings/

7. Server 3D Container Response. The server responds with only one 3D Container 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"
  },
  "links": [
    {
      "rel": "self",
      "href":
"http://helyxapache2.eastus.azurecontainer.io/collections/NewYork/NewYork-
buildings/",
      "type": "application/json",
      "title": "NYC - 3D Buildings Manhattan"
    },
    {
      "rel": "items",
      "href":
"http://helyxapache2.eastus.azurecontainer.io/collections/NewYork/NewYork-
buildings/i3s/",
      "type": "application/json+i3s",
      "title": "NYC - 3D Buildings Manhattan: i3s"
    },
    {
      "rel": "items",
      "href":
"http://helyxapache2.eastus.azurecontainer.io/collections/NewYork/NewYork-
buildings/3dTiles/",
      "type": "application/json+3dtiles",
      "title": "NYC - 3D Buildings Manhattan: 3D Tiles"
    }
  ],
  "content": [
    {
      "title": "NYC - 3D Buildings Manhattan: i3s",
      "rel": "original",
      "href":
"http://helyxapache2.eastus.azurecontainer.io/collections/NewYork/NewYork-
buildings/i3s/",
      "type": "application/json+i3s",
      "collectionType": "3d-container"
    },
    {
      "title": "NYC - 3D Buildings Manhattan: 3D Tiles",
      "rel": "original",
      "href":
"http://helyxapache2.eastus.azurecontainer.io/collections/NewYork/NewYork-
buildings/3dTiles/",
      "type": "application/json+3dtiles",
      "collectionType": "3d-container"
    }
  ]
}
```

8. Client Bounding Volume Hierarchy (BVH) Request. The client parses the 3D container for 3D Tiles link. Client 3D Tiles/tileset.json request : http://helyxapache2.eastus.azurecontainer.io/collections/NewYork/NewYork-buildings/3dTiles/

9. Server 3D tileset.json Response. The server response with tileset.json BVH.

```
{
  "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": [...]
  }
}
```

10. User navigates through the 3D scene

11. Client BVH Content Request. The client requests 3D Tiles batched 3D model from the server. Client B3DM request: http://helyxapache2.eastus.azurecontainer.io/collections/NewYork/

NewYork-buildings/3dTiles/5/0/0.b3dm

12. Server Batched 3D Model Response. Server provides batched 3D model content to the client.

13. Client renders 3D model content to the user.

8.2. I3S To 3D-Tiles Converter Use Case

Please note that the i3s-to-3dtiles service is owned and provided by the Terria [https://terria.io] team at CSIRO's Data61 [https://data61.csiro.au/]. This service is fine to use (lightly) for testing, but production or higher-traffic usage *must* be discussed with the Terria team first.

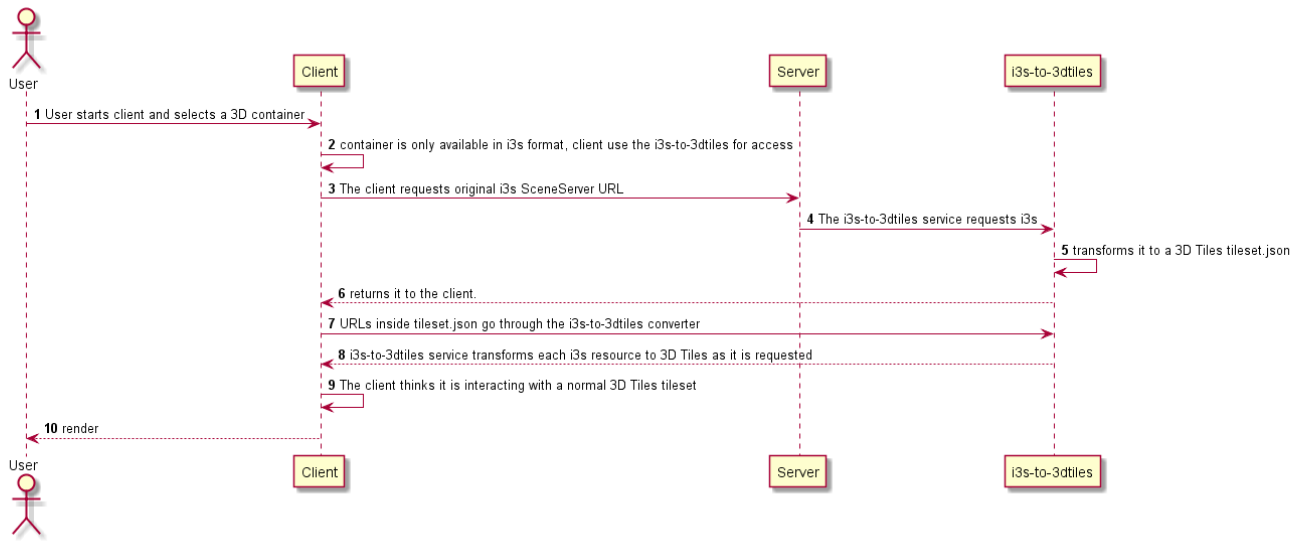

User
Client
Server
i3s-to-3dtiles
1 User starts client and selects a 3D container
2 container is only available in i3s format, client use the i3s-to-3dtiles for access
3 The client requests original i3s SceneServer URL
4 The i3s-to-3dtiles service requests i3s
5 transforms it to a 3D Tiles tileset.json
6 returns it to the client.
7 URLs inside tileset.json go through the i3s-to-3dtiles converter
8 i3s-to-3dtiles service transforms each i3s resource to 3D Tiles as it is requested
9 The client thinks it is interacting with a normal 3D Tiles tileset
10 render
User
Client
Server
i3s-to-3dtiles

Figure 11. I3S to 3D Tiles

The following is the user/client/server sequence for accessing/converting I3S to 3D Tiles:

1. User starts client and selects a 3D container to view.

2. The selected container is only available in I3S format and not as 3D Tiles. The client chooses to use the i3s-to-3dtiles service to access the I3S content.

3. The client requests the original I3S SceneServer URL [https://nsw.digitaltwin.terria.io/i3s-to-3dtiles/]. For example: https://nsw.digitaltwin.terria.io/i3s-to-3dtiles/https://portal.spatial.nsw.gov.au/i3s/rest/services/Hosted/ESP/WaratahStation/SceneServer

4. The i3s-to-3dtiles service requests the original I3S SceneServer URL (https://portal.spatial.nsw.gov.au/i3s/rest/services/Hosted/ESP/WaratahStation/SceneServer).

5. Transforms it to a 3D Tiles tileset.json.

6. Returns it to the client.

7. Relative URLs inside the converted tileset.json, such as `SceneServer/layers/0/nodes/2/geometries/0` go through the i3s-to-3dtiles converter as well when they are resolved to absolute paths, such as https://nsw.digitaltwin.terria.io/i3s-

to-3dtiles/https://portal.spatial.nsw.gov.au/i3s/rest/services/Hosted/ESP/WaratahStation/SceneServer/layers/0/nodes/2/geometries/0.

8. The i3s-to-3dtiles service transforms each I3S resource to 3D Tiles as it is requested. In the case above, the I3S geometry is converted on-the-fly to a 3D Tiles B3DM.
9. The client thinks it is interacting with a normal 3D Tiles tileset and need not have any awareness of the converter at all.
10. Client renders the 3D model content to the user.

Chapter 9. Implementations

The Pilot consisted of four components: Two client applications and two server applications. The following is from the Pilot Call for Participation.

9.1. Components

- D110 - Client for 3D Data Container and Tiles API - A client that interacts with Servers via the proposed OGC API - Tiles-3D and is able to consume and process the data that follows the draft 3D Tiled Data Container specification. Participants developing clients will capture videos and screenshots of the client interfaces interacting with Tiles-3D for demonstration purposes.
- D112 - Client for 3D Data Container and Tiles API and 2D Tiles - A client with support for 2D and 3D data. The client shall interact with 3D Servers via the proposed OGC API - Tiles-3D and is able to consume and process the data that follows the 3D Tiled Data Container specification for 3D data. In order to support 2D data, the client shall interact with the OGC draft API – Maps and Tiles as developed by Testbed-15. Participants developing clients will capture videos and screenshots of the client interfaces interacting Servers for demonstration purposes.
- D100 - Tiles-3D Server - Server-side implementation of the OGC API - Tiles-3D with support for the 3D Tiled Data Container specification.
- D102 – Tiles-2D & Tiles-3D Server - Server-side implementation of the draft OGC API – Maps and Tiles. The focus of this component implementation is on extensions to emerging 2D OGC APIs needed to deliver 2D tiles and support smooth transition to 3D environments. Ideally, this server implements the OGC API - Tiles-3D with support for the 3D Tiled Data Container specification additionally.

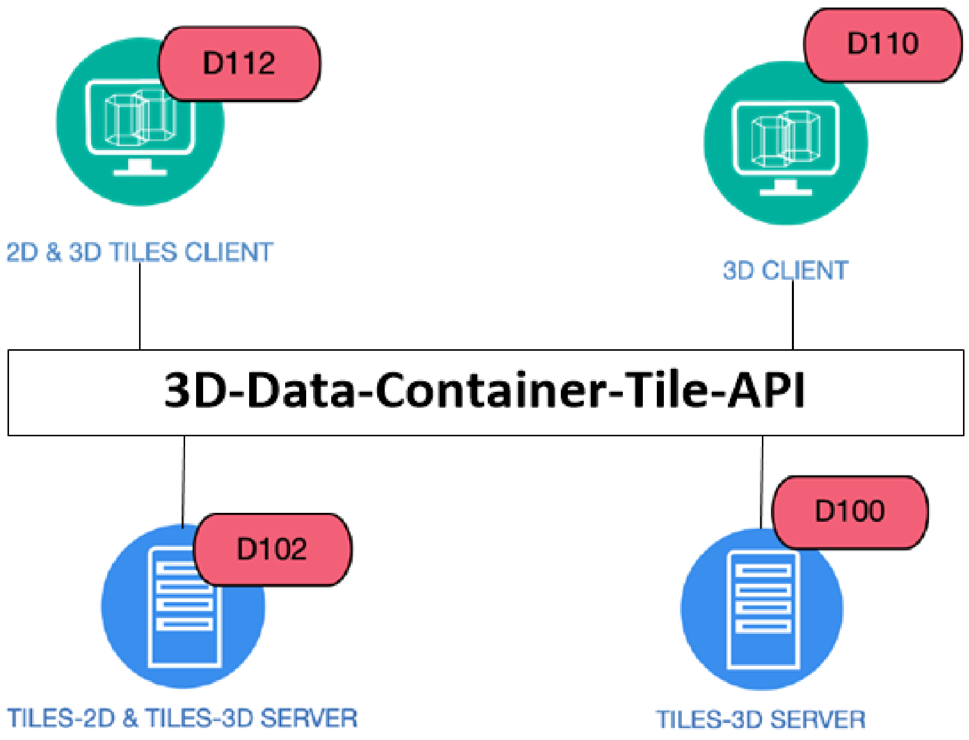


Figure 12. Components

All pilot participants successfully implemented the 3D Data Container specification. The specification supports flat and hierarchal organizations. The flat organization was the most common approach used. The Skymantics implementation used the hierarchal organization of the children property to create a Western Hemisphere container with child containers.

Service	**GeoVolume API Landing Page URL**	**Content**
Cesium D102	https://3d.hypotheticalhorse.com	• Cesium OSM Buildings: 3D Tiles • New York Buildings: 3D Tiles, I3S • Western Sydney City Deal - 3D Buildings: 3D Tiles • Waratah Station BIM: : 3D Tiles, I3S
Cognitics D102	http://cdb.cognitics.net:3000/	• New York Buildings : 3D Tiles
Ecere D102	https://maps.ecere.com/3DAPI/	• New York Buildings: 3D Tiles
Helyx D100	http://helyxapache2.eastus.azurecontainer.io/	• New York Buildings: 3D Tiles, I3S • Montreal Buildings: 3D Tiles + I3S • Montreal LiDAR

Service	GeoVolume API Landing Page URL	Content
Skymantics D102	http://13.82.99.186:5050/	• Westworld ◦ NYC: 3D Tiles, I3S ◦ Marseille: 3D Tiles • Malalison Island
Steinbeis D100	http://steinbeis-3dps.eu:8080/3DContainerTile/	• United States - New York Buildings: 3D Tiles, I3S

Table 9. Pilot Container Implementations

Pilot Participant	Client URL/Application
Cesium D112	https://map.hypotheticalhorse.com/
Cognitics D112	Unity 3D
Ecere D112	GNOSIS Cartographer [http://ecere.ca/gnosis/]
Skymantics D112	http://13.82.99.186:8072/web/3dcat/index.html
Steinbeis D110	http://steinbeis-3dps.eu/STT3DClient/

Table 10. Pilot Client Implementations

9.2. Cesium

In the course of the Pilot, 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 implementations provided by all the other Pilot participants, and the Cesium server was successfully accessed by all other participants' client applications.

9.2.1. Web-based 3D Client

The Cesium 3D Container and Tiles API client is available at https://map.hypotheticalhorse.com.

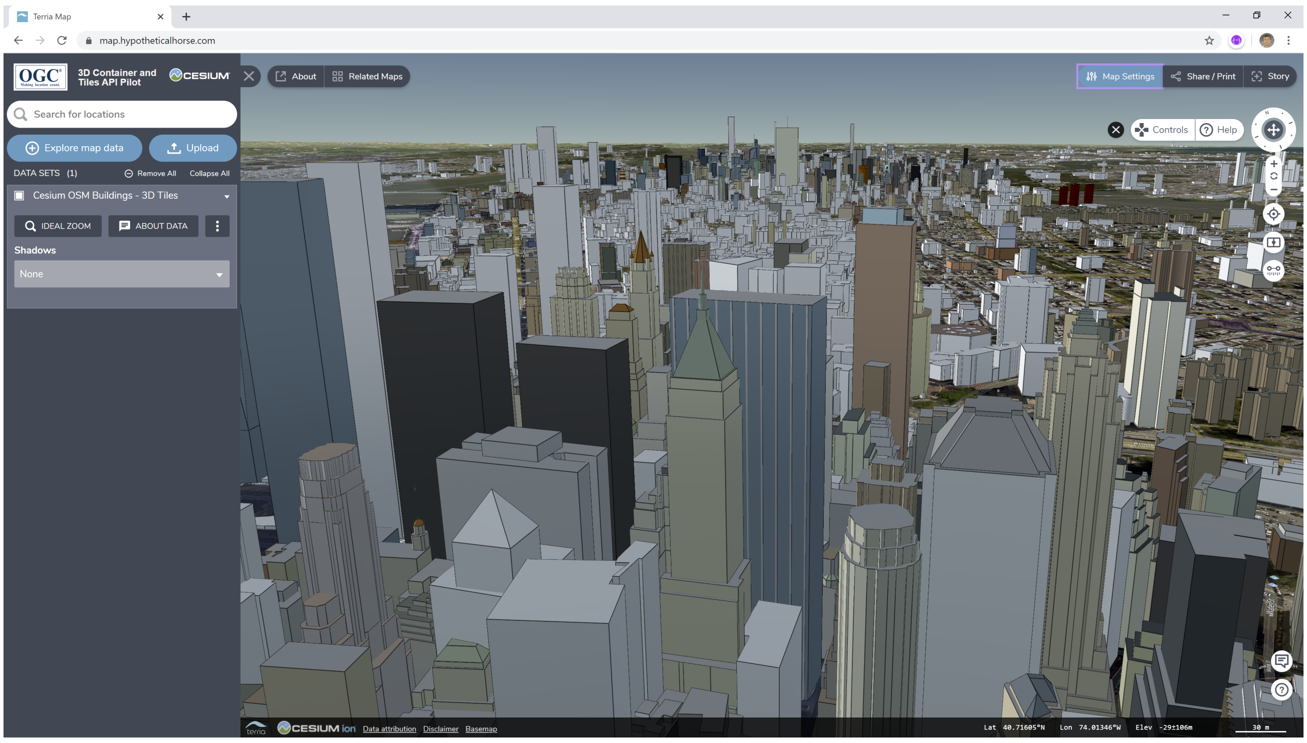


Figure 13. Cesium Web-based 3D Client

The client is built on an open-source framework called TerriaJS [https://terria.io], which in turn is built on top of the open-source CesiumJS [https://cesium.com/cesiumjs/] library. TerriaJS was originally built for Australia's NationalMap. As such, TerriaJS is a good fit for applications that enable users to interactively explore a large catalog of diverse geospatial data. Extending the application to interface with the 3D Container and Tiles API was straightforward.

The 3D Container and Tiles API is found in the TerriaJS catalog, which is accessible by clicking "Explore map data" in the top left. The API is portrayed as a series of folders with the following structure:

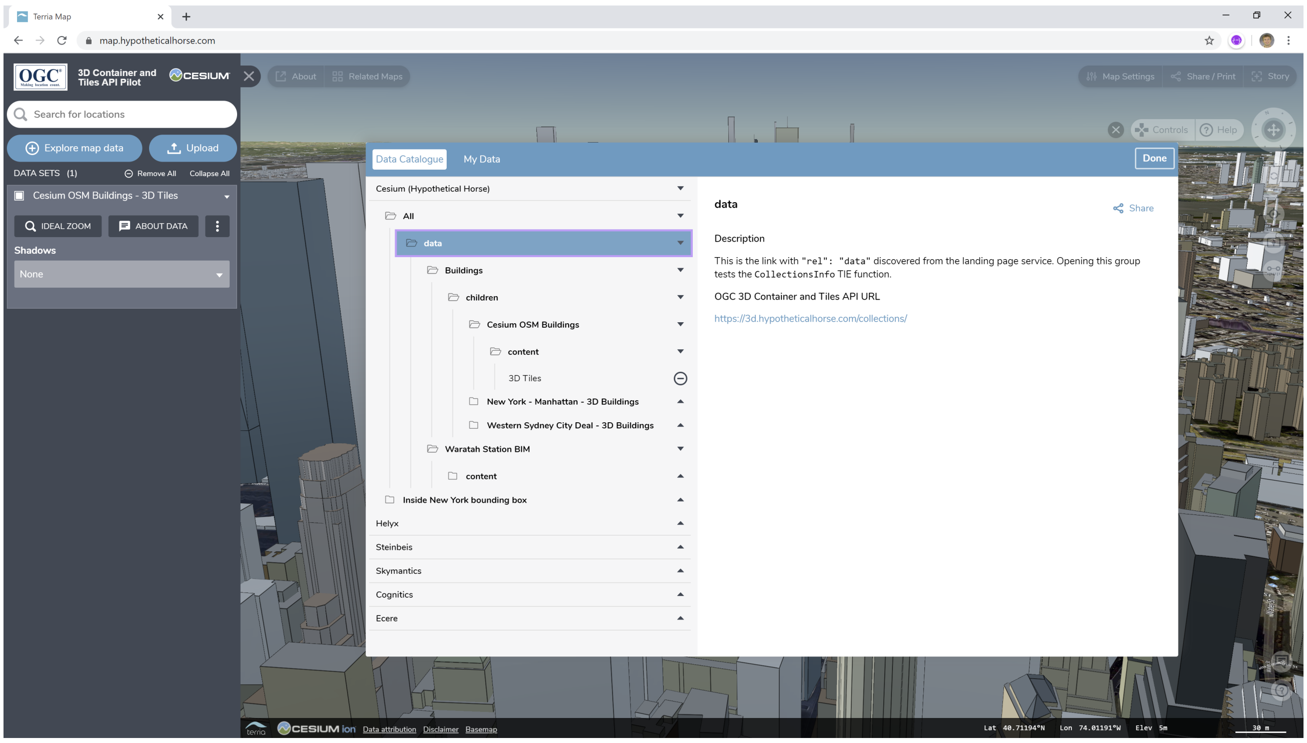
```
[Participant Name] (1)
├── All (2)
│   ├── data (3)
|   │   ├── [3D Container title] (4)
|   |   |   ├── children (5)
|   |   |   |   ├── [Child 3D Container title] (6)
|   |   |   |   |   ├── content (7)
|   |   |   |   |   |   ├── [Distribution title] (8)
|   |   ├── [Another 3D Container title] (4)
|   |   |   ├── content (7)
|   |   |   |   ├── [Distribution title] (8)
├── Inside New York bounding box (9)
│   ├── data (3)
|   │   ├── [3D Container title] (4)
|   |   |   ├── children (5)
|   |   |   |   ├── [Child 3D Container title] (6)
|   |   |   |   |   ├── content (7)
|   |   |   |   |   |   ├── [Distribution title] (8)
|   |   ├── [Another 3D Container title] (4)
|   |   |   ├── content (7)
|   |   |   |   ├── [Distribution title] (8)
```

These parts mirror the structure of the API itself:

1. `[Participant Name]` - Each participant (Cesium, Helyx, Steinbeis, Skymantics, Cognitics, and Ecere) has a top-level folder in the TerriaJS catalog.
2. `All` - A reference to the API's landing page. When this folder is opened, the client does an HTTP GET request to the landing page URL.
3. `data` - Represents the link with `"rel": "data"` discovered from the landing page resource. When the user opens this folder, the client queries the `/collections` resource (which need not have that particular URL).
4. `[3D Container title]` - Each 3D Container discovered from the `/collections` service is listed. When the user opens this folder, the client does an HTTP GET of the container's self link.
5. `children` - If the 3D Container has children, they are listed in this group.
6. `[Child 3D Container title]` - Each 3D Container discovered as a child of another container is listed. When the user opens this folder, the client does an HTTP GET of the child container's self link.
7. `content` - If the 3D container has any content items, they are listed in this group.
8. `[Distribution title]` - Each distribution item discovered in the container's content, such as a 3D Tiles, i3s, or CityGML representation of the dataset, is listed here.

9. `Inside New York bounding box` - This group is identical to `All` (2) except that it specifies a `bbox` parameter when querying `/collections` and parent containers, limiting results to only those containers that are within a bounding box around Manhattan.

In the user interface, it looks like this:



data

Description

This is the link with `"rel": "data"` discovered from the landing page service. Opening this group tests the `CollectionsInfo` TIE function.

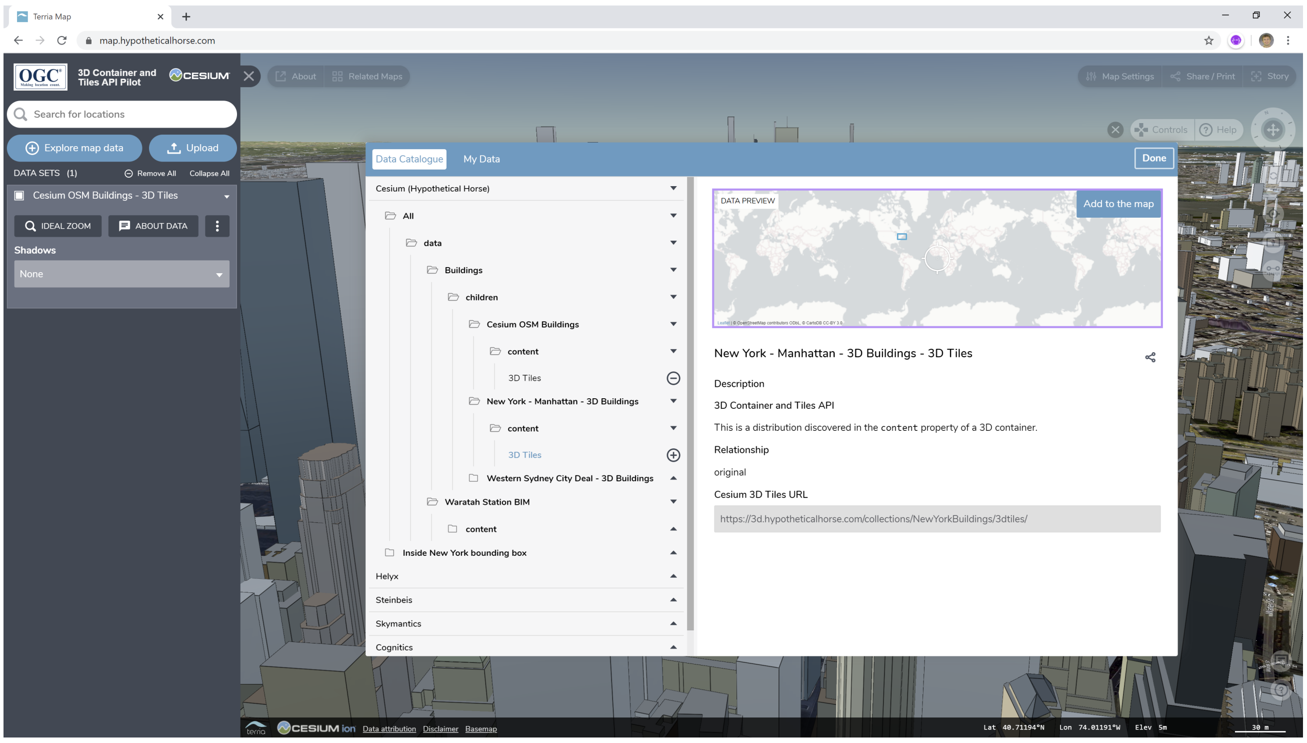
OGC 3D Container and Tiles API URL

https://3d.hypotheticalhorse.com/collections/

Figure 14. API Structure in the User Interface

Please note that the approach used in the Pilot is not the most user-friendly way to portray the API. A typical TerriaJS application would skip the intermediate `data`, `children`, and `content` folders and simply show the entire server as a folder with sub-folders for 3D containers. A more verbose approach is used here in order to reveal and test the structure of the 3D Container and Tiles API.

When the user selects a particular distribution item in the catalog, the client shows the bounding box of the dataset on the preview map.



New York - Manhattan - 3D Buildings - 3D Tiles

Description

3D Container and Tiles API

This is a distribution discovered in the `content` property of a 3D container.

Relationship

original

Cesium 3D Tiles URL

https://3d.hypotheticalhorse.com/collections/NewYorkBuildings/3dtiles/

Figure 15. Bounding Box Preview

Clicking the map zooms in closer to the bounding box:

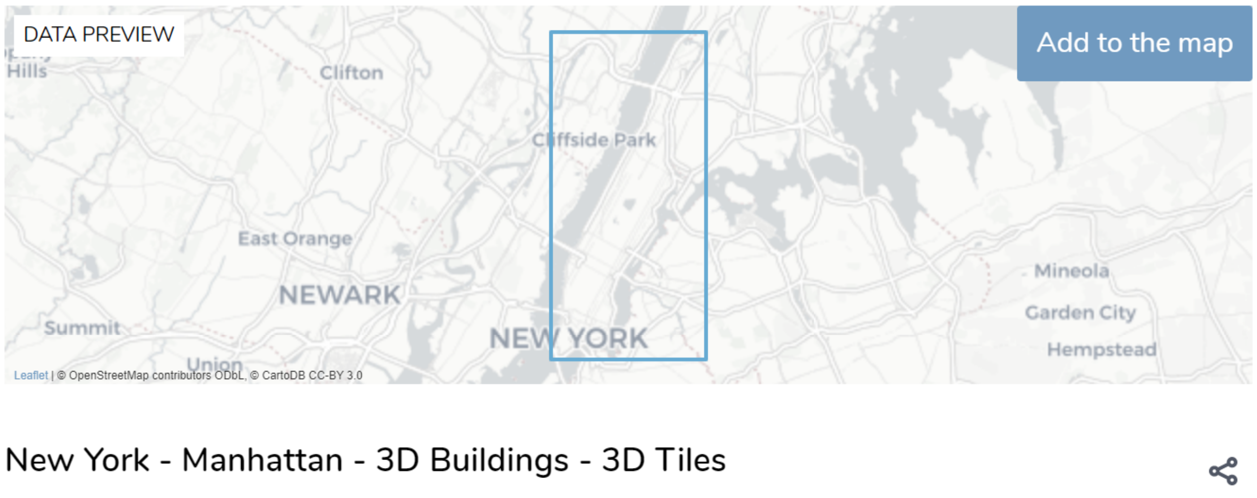
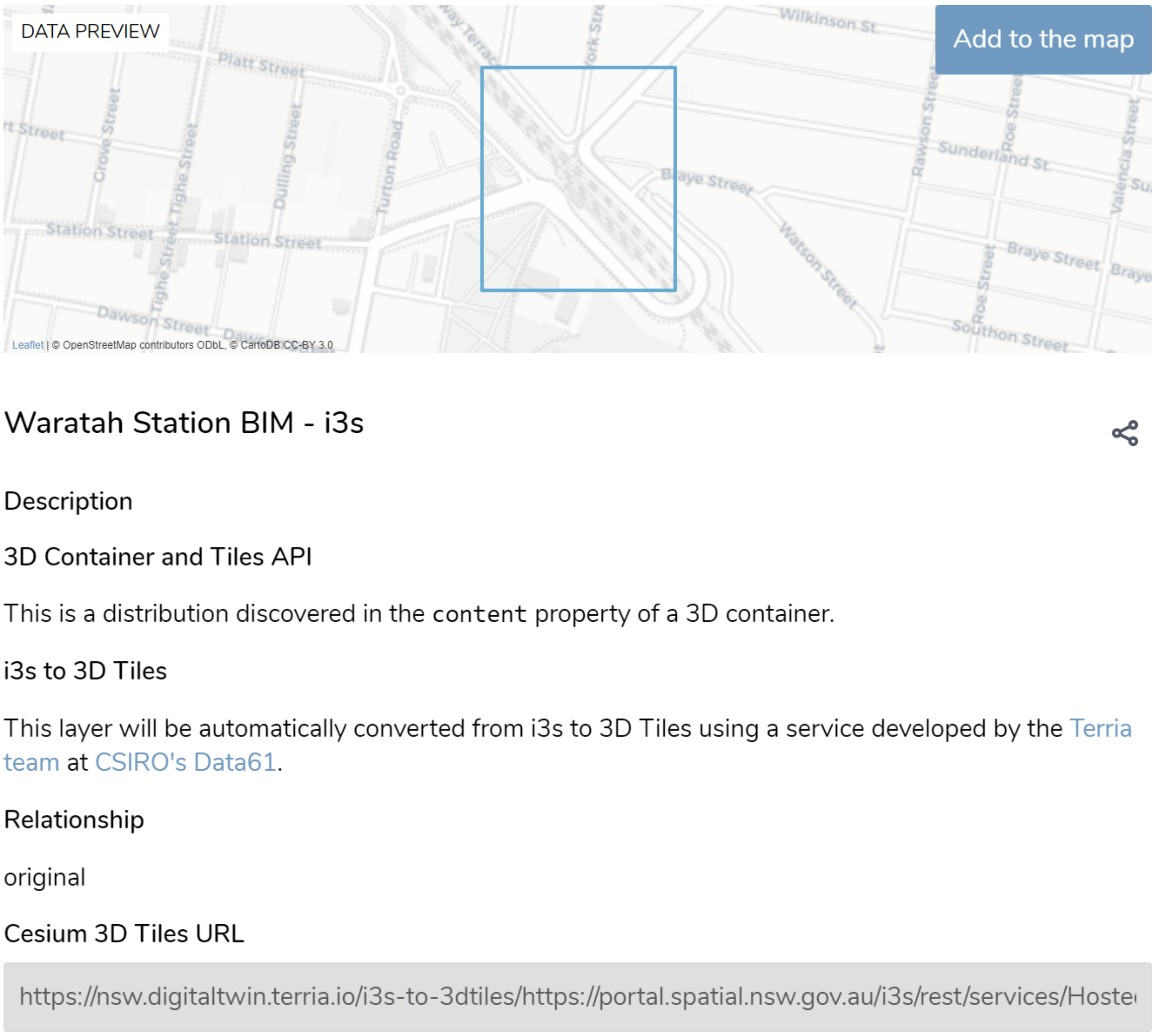


Figure 16. Data preview - New York

While Cesium-based, the client supports visualization of I3S datasets using the I3S to 3D Tiles conversion service developed, owned, and provided by the Terria team at CSIRO’s Data61. The service automatically and transparently converts I3S datasets to 3D Tiles on-the-fly. While the participants of the Pilot had permission to use the service (lightly) for testing in this Pilot, production or higher-traffic usage *must* be discussed with the Terria team first.


DATA PREVIEW
Add to the map
Leaflet | © OpenStreetMap contributors ODbL, © CartoDB CC-BY 3.0

Waratah Station BIM - i3s

Description

3D Container and Tiles API

This is a distribution discovered in the `content` property of a 3D container.

i3s to 3D Tiles

This layer will be automatically converted from i3s to 3D Tiles using a service developed by the Terria team at CSIRO's Data61.

Relationship

original

Cesium 3D Tiles URL

https://nsw.digitaltwin.terria.io/i3s-to-3dtiles/https://portal.spatial.nsw.gov.au/i3s/rest/services/Hoste

Figure 17. Data preview - Waratah Station

The complete source code for the Cesium client application is available on GitHub: https://github.com/kring/TerriaMap/tree/ogc3d

The TerriaJS catalog reference class that interfaces with the 3D Container and Tiles API is found here: https://github.com/kring/TerriaMap/blob/ogc3d/lib/Models/Ogc3dContainerReference.ts

9.2.2. 3D Container and Tiles API Server

The 3D Container and Tiles API server implemented by Cesium 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; and,

- Open source code.

The landing page for the server is found at https://3d.hypotheticalhorse.com/. The landing page follows the recommended structure in OGC API - Common, providing links to a service description, a conformance page, and the collections identified by the server.

https://3d.hypotheticalhorse.com/

```
{
    "title": "Pilot 3D Container API",
    "description": "A pilot of an API for 3D containers and tiles.",
    "links": [
        {
            "title": "Service Description",
            "href": "https://app.swaggerhub.com/apis/timothy-miller/3D-Data-
Container/0.0.1",
            "rel": "service-desc"
        },
        {
            "title": "Conformance",
            "href": "https://3d.hypotheticalhorse.com/conformance/",
            "rel": "conformance"
        },
        {
            "title": "Collections",
            "href": "https://3d.hypotheticalhorse.com/collections/",
            "rel": "data"
        }
    ]
}
```

The collections page, linked from the landing page and found at https://3d.hypotheticalhorse.com/collections/, is the most critical part of the 3D Container and Tiles API. The collections page lists the 3D Container collections known to the server, and provides all the information the client needs to pick a compatible distribution and access it.

https://3d.hypotheticalhorse.com/collections/

```
{
    "links": [
        {
            "href": "https://3d.hypotheticalhorse.com/collections/",
            "rel": "self",
            "type": "application/json",
            "title": "this document"
        }
    ],
    "collections": [
        {
            "id": "Buildings",
            "title": "Buildings",
            "description": "3D Buildings",
            "collectionType": "3d-container",
            "extent": {
                "spatial": {
                    "bbox": [
                        [-180, -90, 0, 180, 90, 1000]
                    ],
                    "crs": "http://www.opengis.net/def/crs/OGC/0/CRS84h"
                }
            },
            "links": [
                {
                    "title": "3D Buildings",
                    "href":
"https://3d.hypotheticalhorse.com/collections/Buildings/",
                    "rel": "self",
                    "type": "application/json"
                }
            ],
            "content": [],
            "children": [
                {
                    "id": "CesiumOSMBuildings",
                    ...
                },
                {
                    "id": "NewYorkBuildings",
                    "title": "New York - Manhattan - 3D Buildings",
                    "description": "3D Buildings in Manhattan, New York.",
                    "collectionType": "3d-container",
                    "extent": {
                        "spatial": {
                            "bbox": [
                                [
                                    -74.01900887327089,
                                    40.700475291581974,
                                    0,
                                    -73.9068954348699,
                                    40.880256294183646,
                                    547.6909683533274
                                ]
                            ],
                            "crs":
"http://www.opengis.net/def/crs/OGC/0/CRS84h"
                        }
                    },
                    "links": [
                        {
                            "title": "New York - Manhattan - 3D Buildings",
                            "href":
"https://3d.hypotheticalhorse.com/collections/Buildings/NewYorkBuildings/",
                            "rel": "self",
                            "type": "application/json"
                        }
                    ],
                    "content": [
                        {
                            "title": "3D Tiles",
                            "href":
"https://3d.hypotheticalhorse.com/collections/NewYorkBuildings/3dtiles/",
                            "rel": "original",
                            "type": "application/json+3dtiles"
                        }
                    ],
                    "children": []
                },
                {
                    "id": "WesternSydneyBuildings",
                    ...
                }
            ]
        },
        {
            "id": "WaratahStation",
            ...
        }
    ]
}
```

The Cesium API offers 3D container data arranged in the following hierarchy:

```
collections/
 ├──── Buildings/
 |    ├──── Cesium OSM Buildings
|    |    ├──── 3D Tiles
 |    ├──── New York - Manhattan - 3D Buildings
|    |    ├──── 3D Tiles
|    ├──── Western Sydney City Deal - 3D Buildings
|    |    ├──── 3D Tiles
 ├──── Waratah Station/
 |    ├──── 3D Tiles
 |    ├──── i3s
```

- *Buildings* - https://3d.hypotheticalhorse.com/collections/Buildings/ - A parent collection container that includes several buildings collections:
  - *Cesium OSM Buildings* - https://3d.hypotheticalhorse.com/collections/Buildings/CesiumOSMBuildings/ - Worldwide 3D buildings served from Cesium ion in 3D Tiles format.
  - *New York - Manhattan - 3D Buildings* - https://3d.hypotheticalhorse.com/collections/Buildings/NewYorkBuildings/ - CityGML-derived buildings in Manhattan served directly from the Cesium 3D Container and Tiles API server in 3D Tiles format.
  - *Western Sydney City Deal - 3D Buildings* - https://3d.hypotheticalhorse.com/collections/Buildings/WesternSydneyBuildings/ - 3D Buildings for New South Wales, Australia's Western Sydney City Deal. The buildings are served from the New South Wales state government's Spatial Digital Twin in 3D Tiles format.
- *Waratah Station* - https://3d.hypotheticalhorse.com/collections/WaratahStation/ - A detailed model of Waratah train station in New South Wales, Australia, served from the New South Wales state government's ArcGIS Server in I3S format and also offered in 3D Tiles format via on-the-fly conversion.

The `Buildings` 3D container may be queried directly via its URL, https://3d.hypotheticalhorse.com/collections/Buildings/, in which case only child containers of that container will be returned.

Collections may be queried by bounding box. For example, to query for all collections in a bounding box around Manhattan we can GET https://3d.hypotheticalhorse.com/collections/?bbox=-74.021,40.701,-73.990,40.775. This will return the `Buildings` 3D container, but it will not return the `Waratah Station` 3D container because it is in Australia.

The user can also query a single 3D container to find all of the container's children that lie within a bounding box. For example, to query the `Buildings` container for child containers in a bounding box around Manhattan, the user or application can GET https://3d.hypotheticalhorse.com/collections/Buildings/?bbox=-74.021,40.701,-73.990,40.775.

The two queries above use a 2D bounding box, but a 3D bounding box with heights is also supported. The following query does not include the Manhattan dataset because the tallest building in Manhattan reaches a height of less than 550 meters: https://3d.hypotheticalhorse.com/collections/Buildings/?bbox=-180,-90,550,180,90,1000.

9.2.2.1. Architecture

The Cesium API server consists of a "dynamic" portion backed by a Node.js-based server running on an Amazon EC2 virtual machine, combined with a static portion backed by the Amazon Simple Storage Service (S3). Amazon CloudFront is used to make these two parts appear to clients as a single cohesive server.

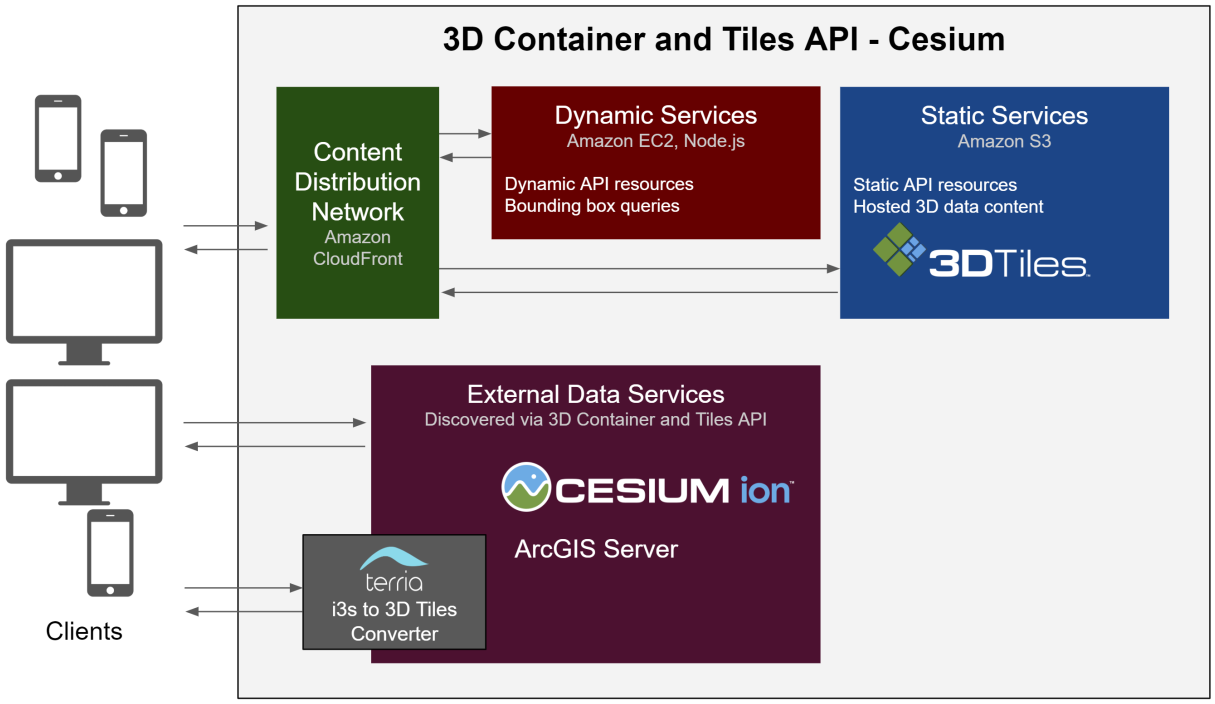


Figure 18. Cesium API Server Architecture

The open-source code for the dynamic portion may be found on GitHub: https://github.com/TerriaJS/terriajs-server/tree/ogc3d

Specifically, the OGC 3D Container and Tiles API services are implemented here: https://github.com/TerriaJS/terriajs-server/blob/ogc3d/lib/controllers/ogc3d.js

9.3. Cognitics

9.3.1. Introduction

Cognitics implemented a static 3D server providing 3DTiles data and developed a client using Unity 3D to query and display 3D data from other participant servers.

9.3.2. Data

The Pilot participants provided the data available on the Cognitics server. The data consisted of the 3D Tiles WGS84 Quadtree Level 14, the same data as the 3D Tiles, but using a uniform quadtree. All the data tiles are placed on level 14 of a double-headed quadtree where the Level 0 tiles are -180°→0° (Western Hemisphere) and 0°→180° (Eastern Hemisphere).

9.3.3. Server Architecture

The Cognitics server architecture is shown below.

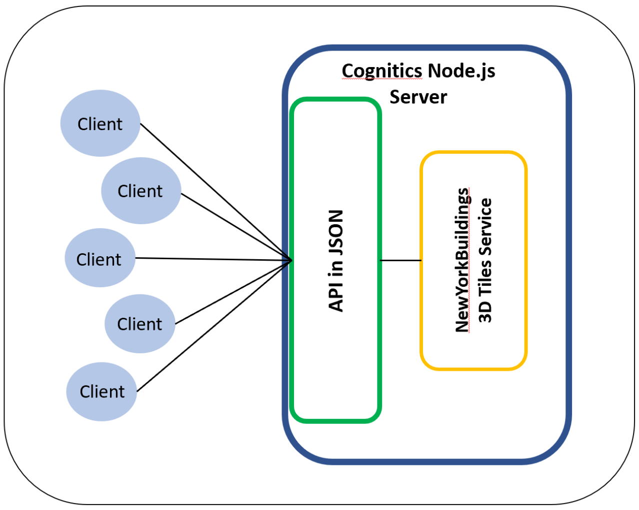


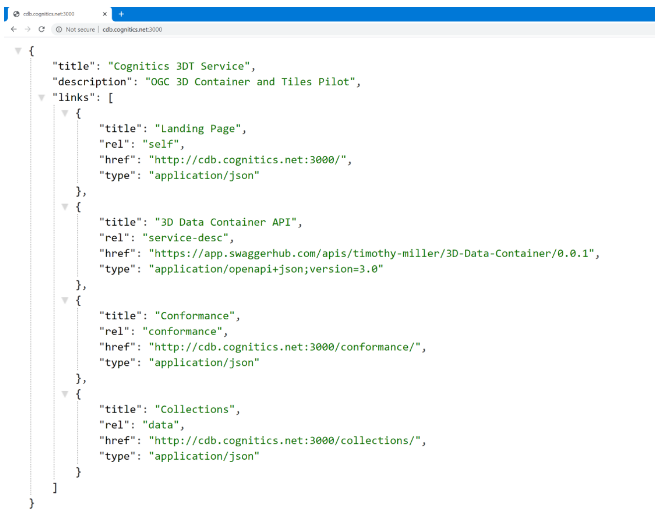
Figure 19. Cognitics Server Architecture

9.3.4. Server Functionality:

Cognitics developed a simple Node.js server that provides static content. The implementation provides glTF content as a single container based on a request via 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.

9.3.4.1. Landing Page

The Cognitics API landing page shown below is available at http://cdb.cognitics.net:3000/. The landing page displays in JSON and shows the API description, and links to the conformance classes and collections page.



cdb.cognitics.net:3000

Not secure | cdb.cognitics.net:3000

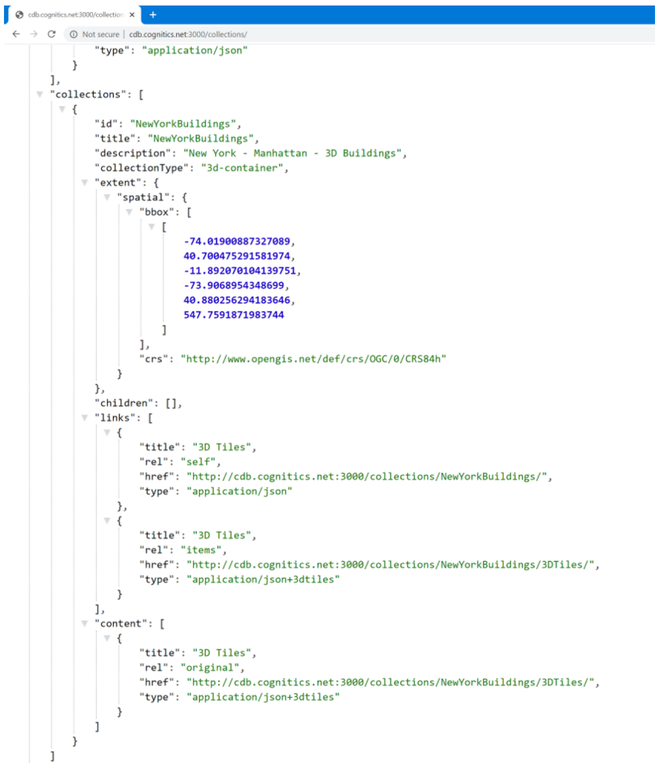
```
{
    "title": "Cognitics 3DT Service",
    "description": "OGC 3D Container and Tiles Pilot",
    "links": [
        {
            "title": "Landing Page",
            "rel": "self",
            "href": "http://cdb.cognitics.net:3000/",
            "type": "application/json"
        },
        {
            "title": "3D Data Container API",
            "rel": "service-desc",
            "href": "https://app.swaggerhub.com/apis/timothy-miller/3D-Data-Container/0.0.1",
            "type": "application/openapi+json;version=3.0"
        },
        {
            "title": "Conformance",
            "rel": "conformance",
            "href": "http://cdb.cognitics.net:3000/conformance/",
            "type": "application/json"
        },
        {
            "title": "Collections",
            "rel": "data",
            "href": "http://cdb.cognitics.net:3000/collections/",
            "type": "application/json"
        }
    ]
}
```

9.3.4.2. 3D Container

The collections page is available at http://cdb.cognitics.net:3000/collections/. Here the API returns information about the available 3D containers. The Cognitics implementation of the API offers a data hierarchy as shown below.

```
collections/
 ├──── NewYorkBuildings/
 │     ├──── 3D Tiles
```

The image below shows the Cognitics collections page, which consists of NewYorkBuildings.



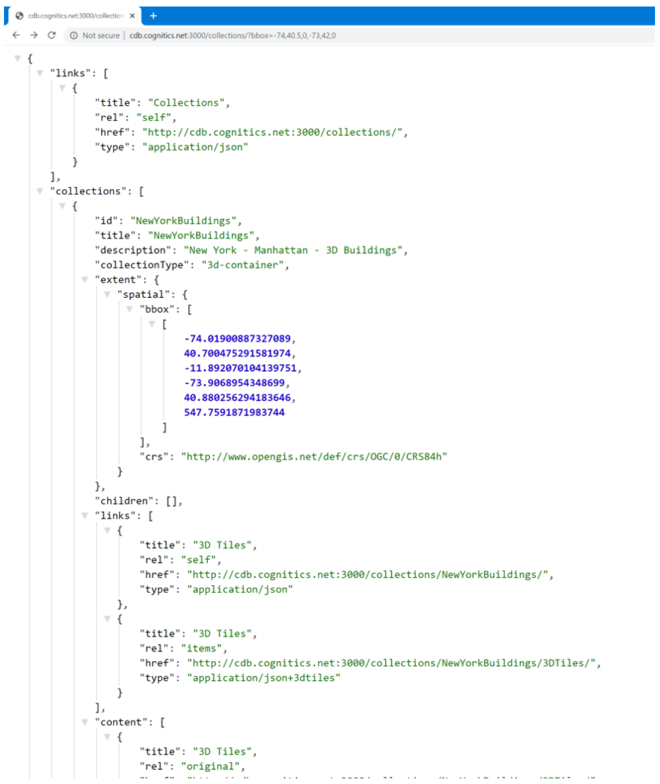
cdb.cognitics.net:3000/collection

Not secure | cdb.cognitics.net:3000/collections/

```
            "type": "application/json"
        }
    ],
    "collections": [
        {
            "id": "NewYorkBuildings",
            "title": "NewYorkBuildings",
            "description": "New York - Manhattan - 3D Buildings",
            "collectionType": "3d-container",
            "extent": {
                "spatial": {
                    "bbox": [
                        [
                            -74.01900887327089,
                            40.700475291581974,
                            -11.892070104139751,
                            -73.9068954348699,
                            40.880256294183646,
                            547.7591871983744
                        ]
                    ],
                    "crs": "http://www.opengis.net/def/crs/OGC/0/CRS84h"
                }
            },
            "children": [],
            "links": [
                {
                    "title": "3D Tiles",
                    "rel": "self",
                    "href": "http://cdb.cognitics.net:3000/collections/NewYorkBuildings/",
                    "type": "application/json"
                },
                {
                    "title": "3D Tiles",
                    "rel": "items",
                    "href": "http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/",
                    "type": "application/json+3dtiles"
                }
            ],
            "content": [
                {
                    "title": "3D Tiles",
                    "rel": "original",
                    "href": "http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/",
                    "type": "application/json+3dtiles"
                }
            ]
        }
    ]
```

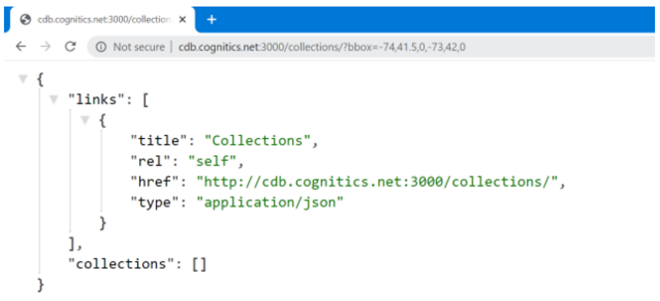
9.3.4.3. Bounding Box query

The Cognitics implementation if the API supports querying of 3D containers and collections using either a 2D or 3D bounding box, and supports both fully contains and intersection matching functions. The image below shows a bounding box query that returns the NewYorkBuildings collection.



```
cdb.cognitics.net:3000/collections/?bbox=-74,40.5,0,-73,42,0
{
  "links": [
    {
      "title": "Collections",
      "rel": "self",
      "href": "http://cdb.cognitics.net:3000/collections/",
      "type": "application/json"
    }
  ],
  "collections": [
    {
      "id": "NewYorkBuildings",
      "title": "NewYorkBuildings",
      "description": "New York - Manhattan - 3D Buildings",
      "collectionType": "3d-container",
      "extent": {
        "spatial": {
          "bbox": [
            [
              -74.01900887327089,
              40.700475291581974,
              -11.892070104139751,
              -73.9068954348699,
              40.880256294183646,
              547.7591871983744
            ]
          ],
          "crs": "http://www.opengis.net/def/crs/OGC/0/CRS84h"
        }
      },
      "children": [],
      "links": [
        {
          "title": "3D Tiles",
          "rel": "self",
          "href": "http://cdb.cognitics.net:3000/collections/NewYorkBuildings/",
          "type": "application/json"
        },
        {
          "title": "3D Tiles",
          "rel": "items",
          "href": "http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/",
          "type": "application/json+3dtiles"
        }
      ],
      "content": [
        {
          "title": "3D Tiles",
          "rel": "original",
```

To complete the TIE tests for the bounding box function, Cognitics queried the collection using -74, 41.5, 0 and -73, 42, 0 coordinates. If the participant server's implementation of the bounding box function was correct, nothing would be returned in our collections list.



```
cdb.cognitics.net:3000/collections/?bbox=-74,41.5,0,-73,42,0
{
  "links": [
    {
      "title": "Collections",
      "rel": "self",
      "href": "http://cdb.cognitics.net:3000/collections/",
      "type": "application/json"
    }
  ],
  "collections": []
}
```

9.3.5. Client Functionality:

9.3.5.1. Request and parse Landing Page

Cognitics used Unity 3D for the client. From the File menu, select the 3DT Link menu and a dialog box appears prompting the user to enter the participant server URL. In Unity, the user sends the bounding box parameter in the URL query string.

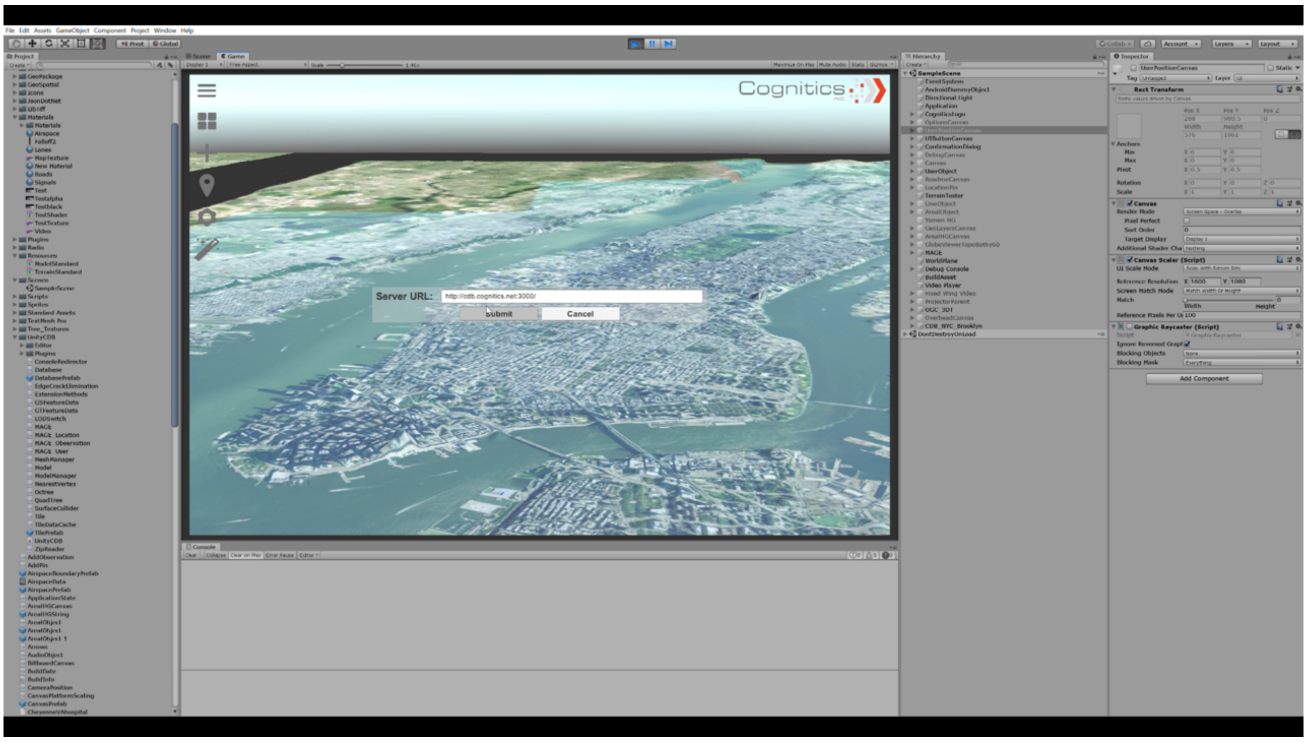


Figure 20. Request and Parse Landing Page

Unity fetches the server's landing page and parses the content for collections using the rel: data key to ensure the correct elements are returned. If a query string was used, the participant server provides a response containing only the entries that intersect with the bounding box passed in the query string.

9.3.5.2. Request and parse 3D container

Once all available collections have been parsed, 3D Tiles datasets are returned as a list in a dialog box and the user chooses which collection to be displayed.

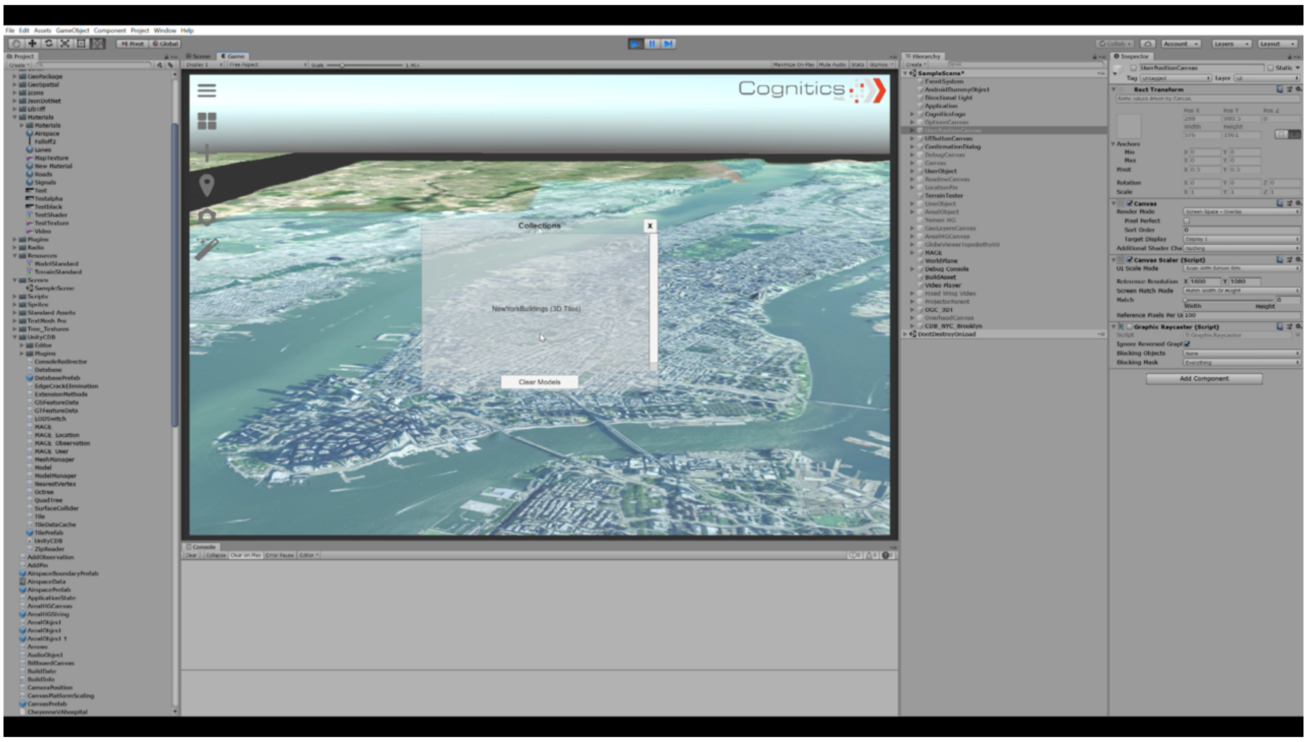
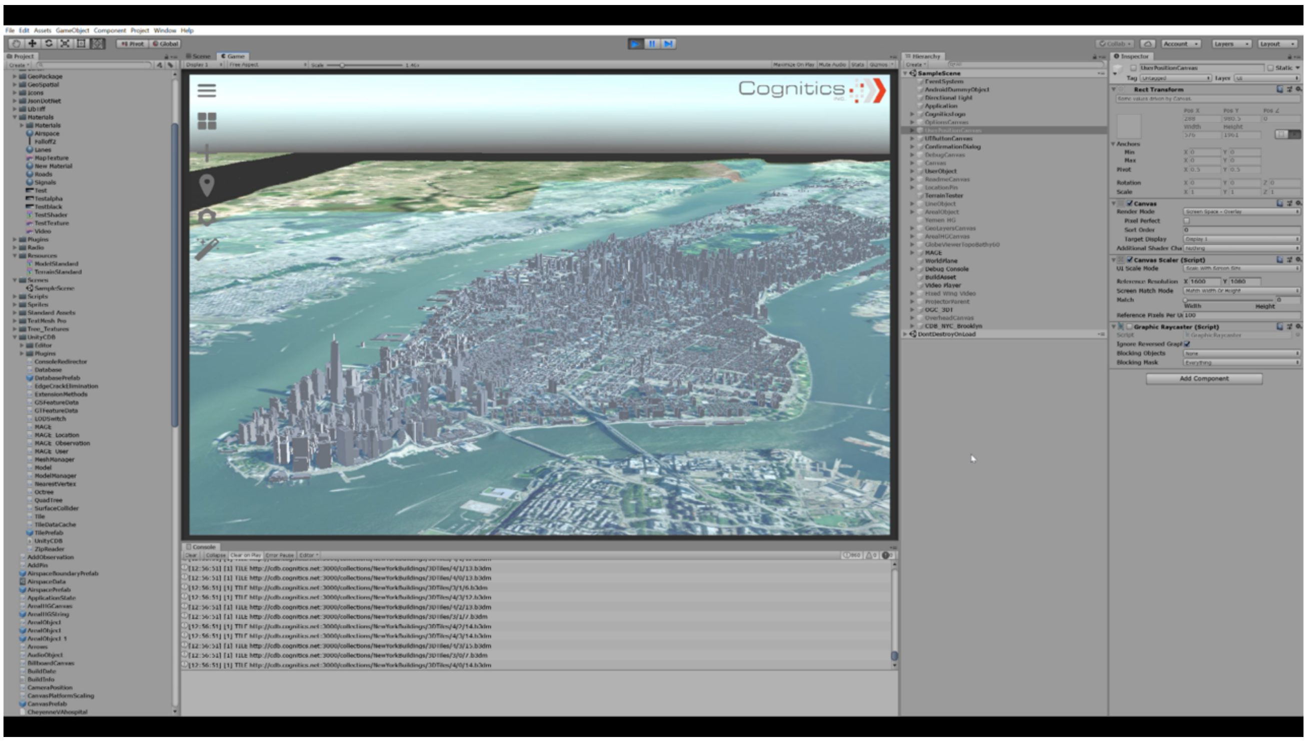


Figure 21. Request and Parse 3D Container

In Unity the bounding box is set based on the CDB bounds, rather than drawing or typing in a set of bounding box coordinates. The image below shows the results of the client request for the NewYorkBuildings collection.



```
[12:56:51] [1] TILE http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/4/1/13.b3dm
[12:56:51] [1] TILE http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/4/0/13.b3dm
[12:56:51] [1] TILE http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/3/1/6.b3dm
[12:56:51] [1] TILE http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/4/3/12.b3dm
[12:56:51] [1] TILE http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/4/2/13.b3dm
[12:56:51] [1] TILE http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/3/1/7.b3dm
[12:56:51] [1] TILE http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/4/2/14.b3dm
[12:56:51] [1] TILE http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/4/3/14.b3dm
[12:56:51] [1] TILE http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/4/3/15.b3dm
[12:56:51] [1] TILE http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/3/0/7.b3dm
[12:56:51] [1] TILE http://cdb.cognitics.net:3000/collections/NewYorkBuildings/3DTiles/4/0/14.b3dm
```

Figure 22. Client Request Results

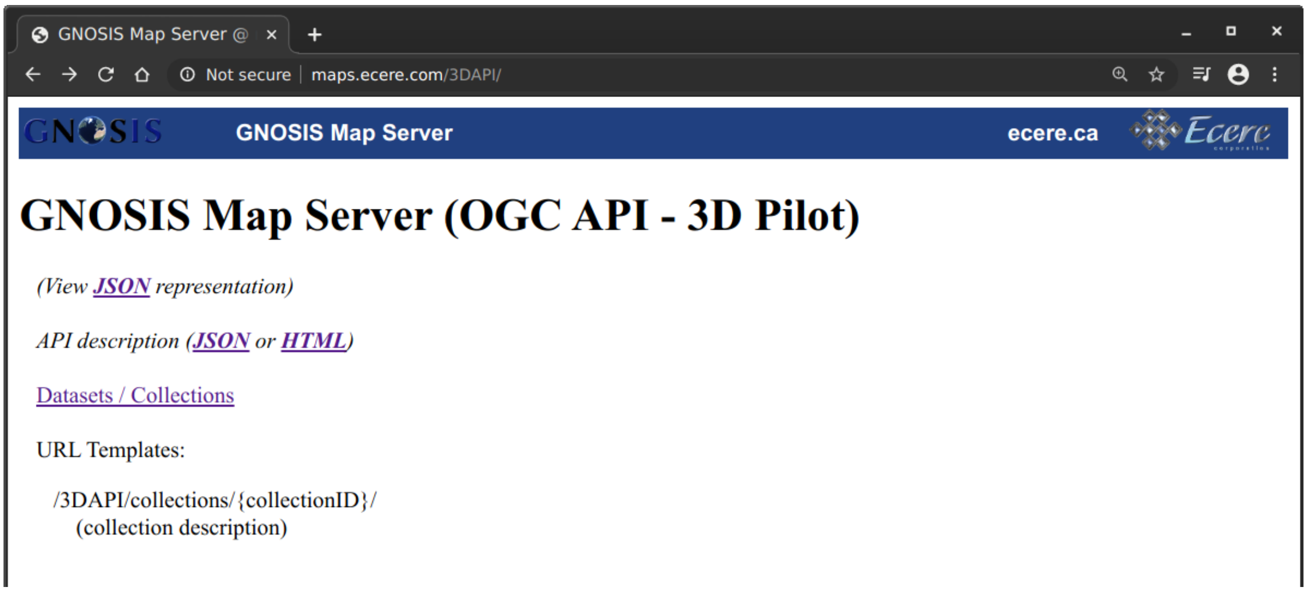
9.4. Ecere

For the Pilot, Ecere provided both server and client components based on the Ecere GNOSIS cross-platform geospatial software products. The Pilot allowed prototyping and experimenting with the various OGC API components with the integration of multiple ways to deliver and access 3D data, including the Features API, Tiles API as well as the 3D Tiles and I3S Community standards.

9.4.1. Server

9.4.1.1. Static server

The main server component provided by Ecere used in the TIEs with other participants was a very simple implementation as static files hosted using Apache. The implementation provided a landing page and a list of available 3D data layers consisting of the same New York City buildings served by all participants and distributed as 3D tiles. Both JSON and HTML representations of the landing page and data layers resources were made available. This experiment demonstrated that it is possible to implement this API by simply setting up a file structure and serving it using a readily available web server.


GNOSIS Map Server (OGC API - 3D Pilot)

(View JSON representation)

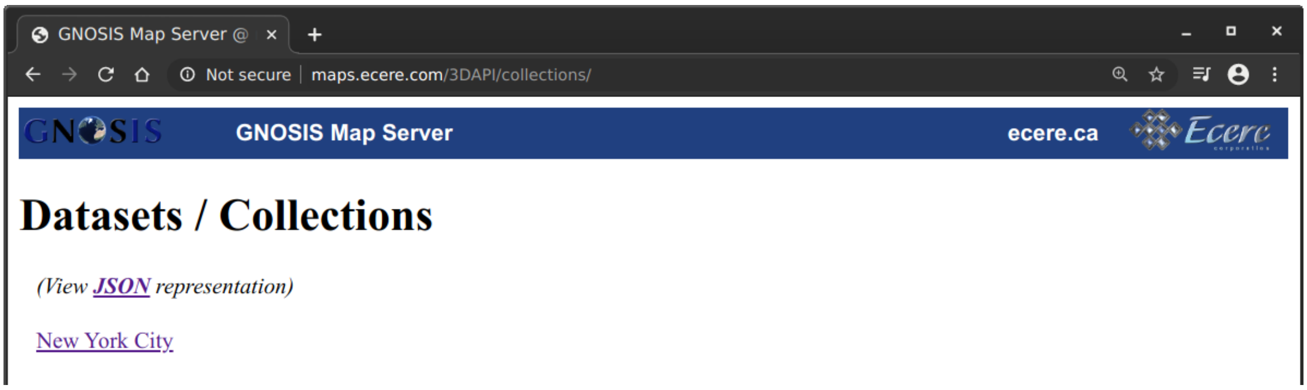
API description (JSON or HTML)

Datasets / Collections

URL Templates:

/3DAPI/collections/{collectionID}/
(collection description)

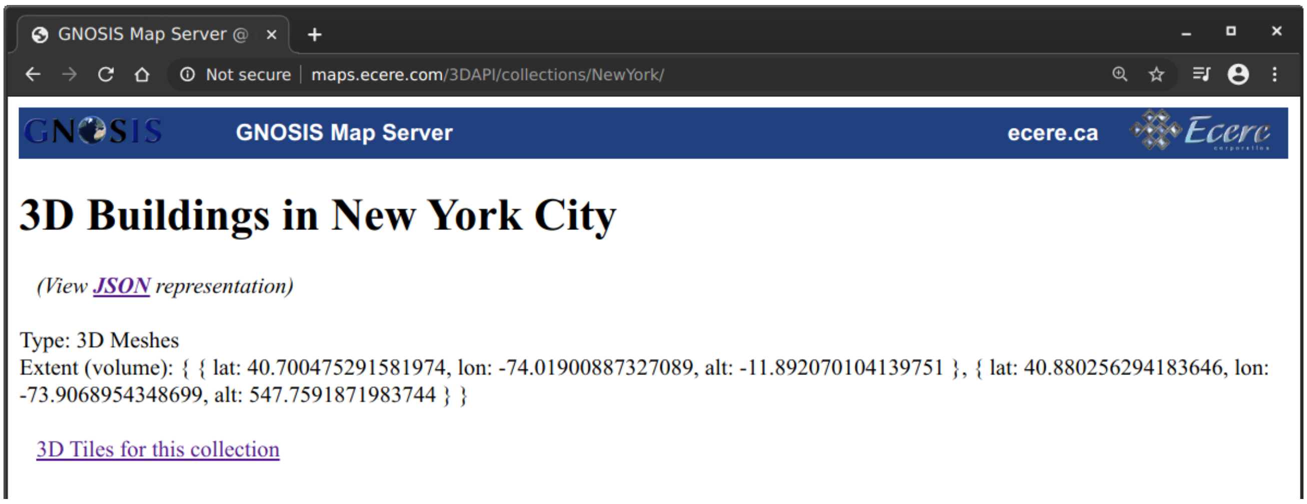
Figure 23. Landing page of static 3D API server


Datasets / Collections

(View JSON representation)

New York City

Figure 24. Listing of data layers for static 3D API server



3D Buildings in New York City

(View JSON representation)

Type: 3D Meshes
Extent (volume): { { lat: 40.700475291581974, lon: -74.01900887327089, alt: -11.892070104139751 }, { lat: 40.880256294183646, lon: -73.9068954348699, alt: 547.7591871983744 } }

3D Tiles for this collection

Figure 25. 3D Buildings in New York City description page from static 3D API server

9.4.1.2. Dynamic server - Tiles API

A dynamic web server based on Ecere's GNOSIS Map Server using the Tiles API to distribute the 3D data was also used in the Pilot. Multi-resolution tiles of imagery, elevation and vector data were made available, following a grid described by the OGC Two-Dimensional Tile Matrix Set standard. Models were referenced as points. The server supports multiple tiling schemes, but the GNOSIS Global Grid [https://maps.ecere.com/geoapi/tileMatrixSets/GNOSISGlobalGrid] was used for the experiments conducted in the Pilot. This approach was also previously demonstrated in OGC Testbed 14 CityGML & Augmented Reality (see Engineering Report [http://docs.opengeospatial.org/per/18-025.html]).

The Buildings and Tree 3D models tiles can be accessed as either:

- Tiled vector points references (as GNOSIS Map Tiles, Mapbox Vector Tile, or GeoJSON);
- E3D model;
- glTF model;
- Batched 3D model embedding glTF (linked by 3D Tiles tileset in approach described below).

Individual tiles are accessible at:

…/collections/{collectionId}/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}

Selection of an output format is done via any of HTTP Accept-content headers, f= query parameter or .{format}.

The individual features can also be accessed (e.g. as GeoJSON or HTML representation) at:

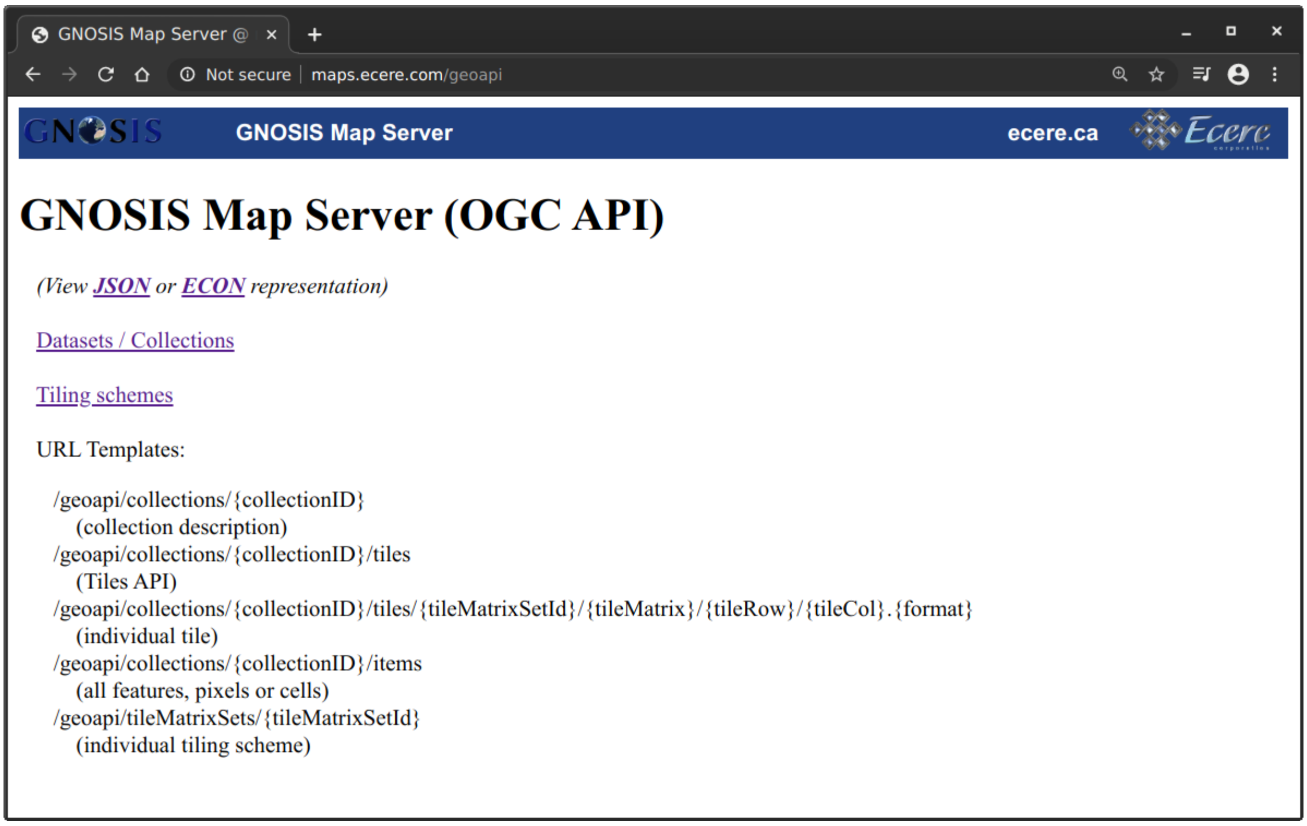
…/collections/{collectionId}/items/{itemID}

When points tiles are used, they reference models accessible at:

…/collections/{collectionId}/models/{modelID}

and models can reference textures at:

…/collections/{collectionId}/textures/{textureID}



GNOSIS Map Server (OGC API)

(View JSON or ECON representation)

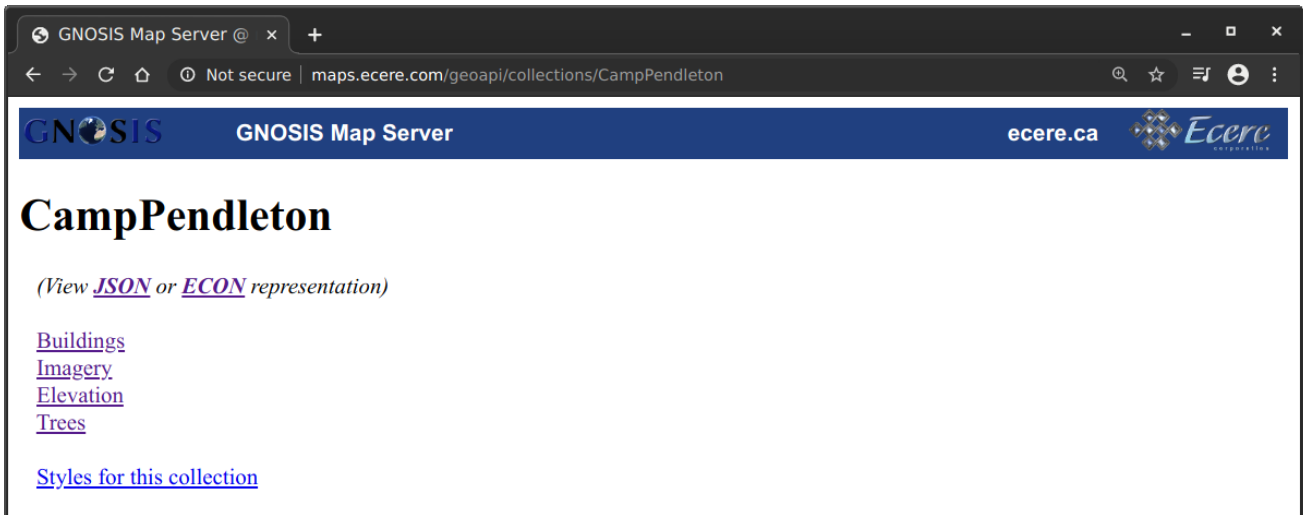
Datasets / Collections

Tiling schemes

URL Templates:

/geoapi/collections/{collectionID}
(collection description)
/geoapi/collections/{collectionID}/tiles
(Tiles API)
/geoapi/collections/{collectionID}/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}.{format}
(individual tile)
/geoapi/collections/{collectionID}/items
(all features, pixels or cells)
/geoapi/tileMatrixSets/{tileMatrixSetId}
(individual tiling scheme)

Figure 26. Landing page of dynamic 3D API server


GNOSIS Map Server @
Not secure | maps.ecere.com/geoapi/collections/CampPendleton
GNOSIS
GNOSIS Map Server
ecere.ca
Ecere

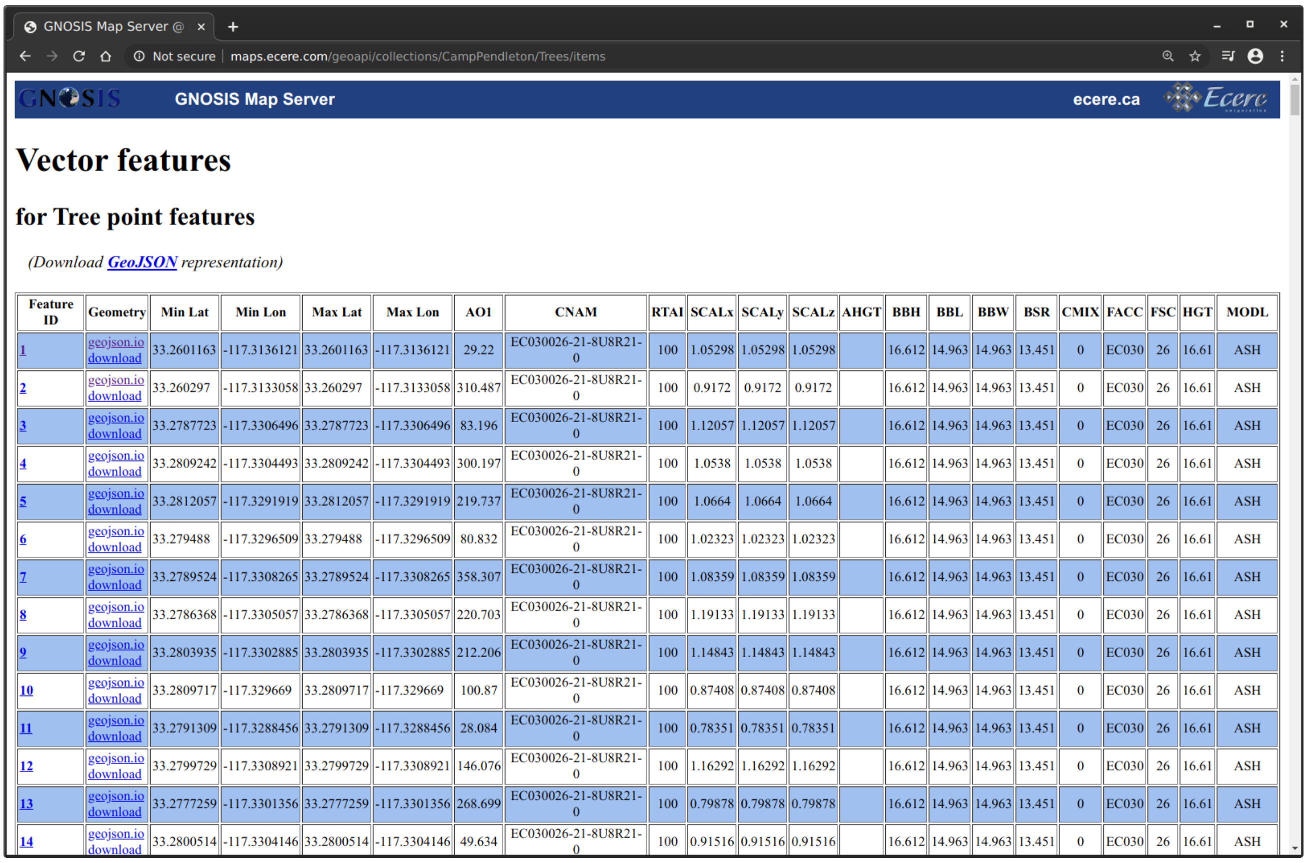
CampPendleton

(View JSON or ECON representation)

Buildings
Imagery
Elevation
Trees

Styles for this collection

Figure 27. Listing of data layers for Camp Pendleton from dynamic server


GNOSIS Map Server @

Not secure | maps.ecere.com/geoapi/collections/CampPendleton/Trees/items

GNOSIS Map Server ecere.ca

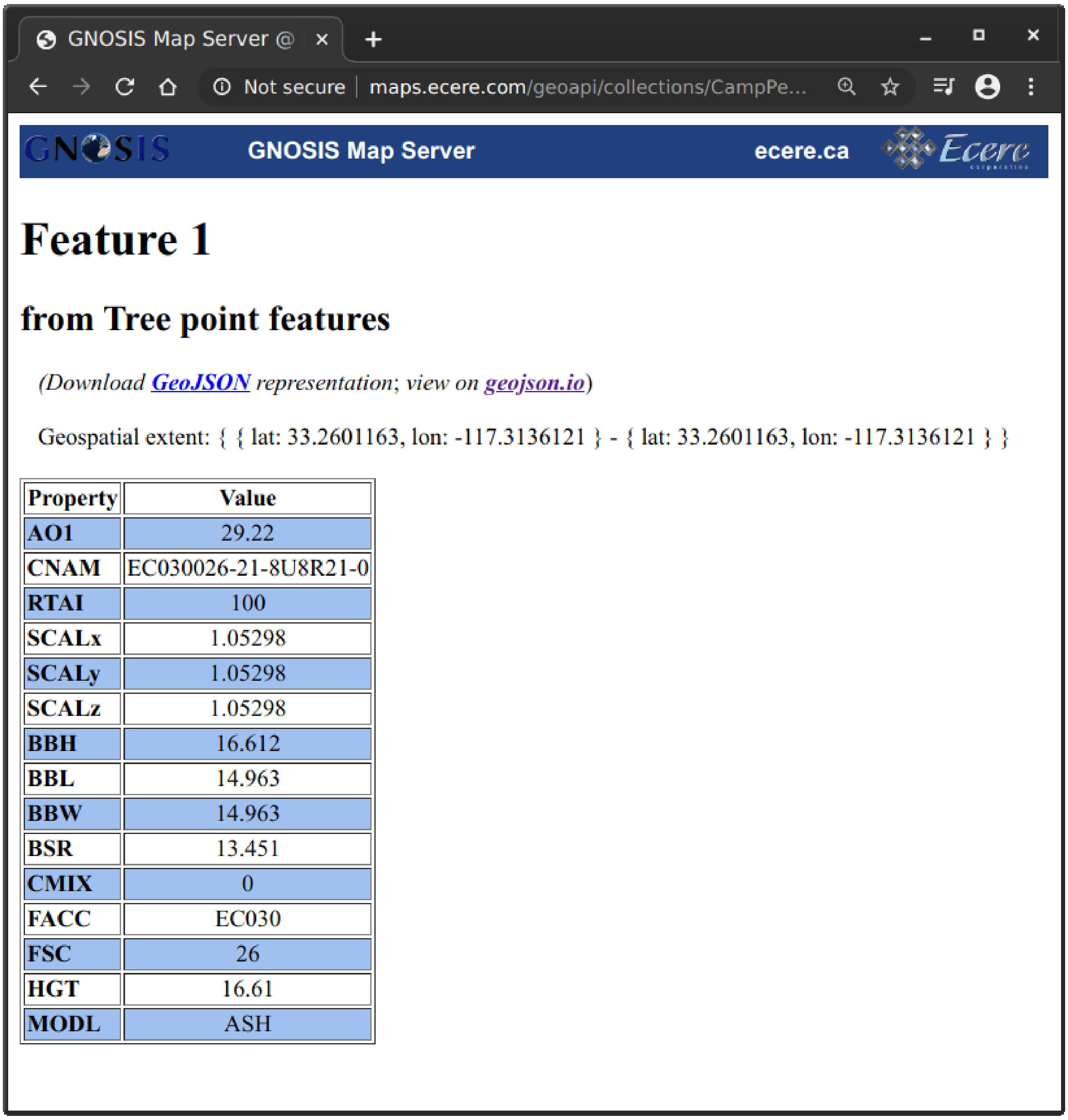
Vector features

for Tree point features

*(Download **GeoJSON** representation)*

Feature ID	Geometry	Min Lat	Min Lon	Max Lat	Max Lon	AO1	CNAM	RTAI	SCALx	SCALy	SCALz	AHGT	BBH	BBL	BBW	BSR	CMIX	FACC	FSC	HGT	MODL
1	geojson.io download	33.2601163	-117.3136121	33.2601163	-117.3136121	29.22	EC030026-21-8U8R21-0	100	1.05298	1.05298	1.05298		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH
2	geojson.io download	33.260297	-117.3133058	33.260297	-117.3133058	310.487	EC030026-21-8U8R21-0	100	0.9172	0.9172	0.9172		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH
3	geojson.io download	33.2787723	-117.3306496	33.2787723	-117.3306496	83.196	EC030026-21-8U8R21-0	100	1.12057	1.12057	1.12057		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH
4	geojson.io download	33.2809242	-117.3304493	33.2809242	-117.3304493	300.197	EC030026-21-8U8R21-0	100	1.0538	1.0538	1.0538		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH
5	geojson.io download	33.2812057	-117.3291919	33.2812057	-117.3291919	219.737	EC030026-21-8U8R21-0	100	1.0664	1.0664	1.0664		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH
6	geojson.io download	33.279488	-117.3296509	33.279488	-117.3296509	80.832	EC030026-21-8U8R21-0	100	1.02323	1.02323	1.02323		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH
7	geojson.io download	33.2789524	-117.3308265	33.2789524	-117.3308265	358.307	EC030026-21-8U8R21-0	100	1.08359	1.08359	1.08359		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH
8	geojson.io download	33.2786368	-117.3305057	33.2786368	-117.3305057	220.703	EC030026-21-8U8R21-0	100	1.19133	1.19133	1.19133		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH
9	geojson.io download	33.2803935	-117.3302885	33.2803935	-117.3302885	212.206	EC030026-21-8U8R21-0	100	1.14843	1.14843	1.14843		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH
10	geojson.io download	33.2809717	-117.329669	33.2809717	-117.329669	100.87	EC030026-21-8U8R21-0	100	0.87408	0.87408	0.87408		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH
11	geojson.io download	33.2791309	-117.3288456	33.2791309	-117.3288456	28.084	EC030026-21-8U8R21-0	100	0.78351	0.78351	0.78351		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH
12	geojson.io download	33.2799729	-117.3308921	33.2799729	-117.3308921	146.076	EC030026-21-8U8R21-0	100	1.16292	1.16292	1.16292		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH
13	geojson.io download	33.2777259	-117.3301356	33.2777259	-117.3301356	268.699	EC030026-21-8U8R21-0	100	0.79878	0.79878	0.79878		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH
14	geojson.io download	33.2800514	-117.3304146	33.2800514	-117.3304146	49.634	EC030026-21-8U8R21-0	100	0.91516	0.91516	0.91516		16.612	14.963	14.963	13.451	0	EC030	26	16.61	ASH

Figure 28. Features API for Camp Pendleton trees from dynamic server



Feature 1

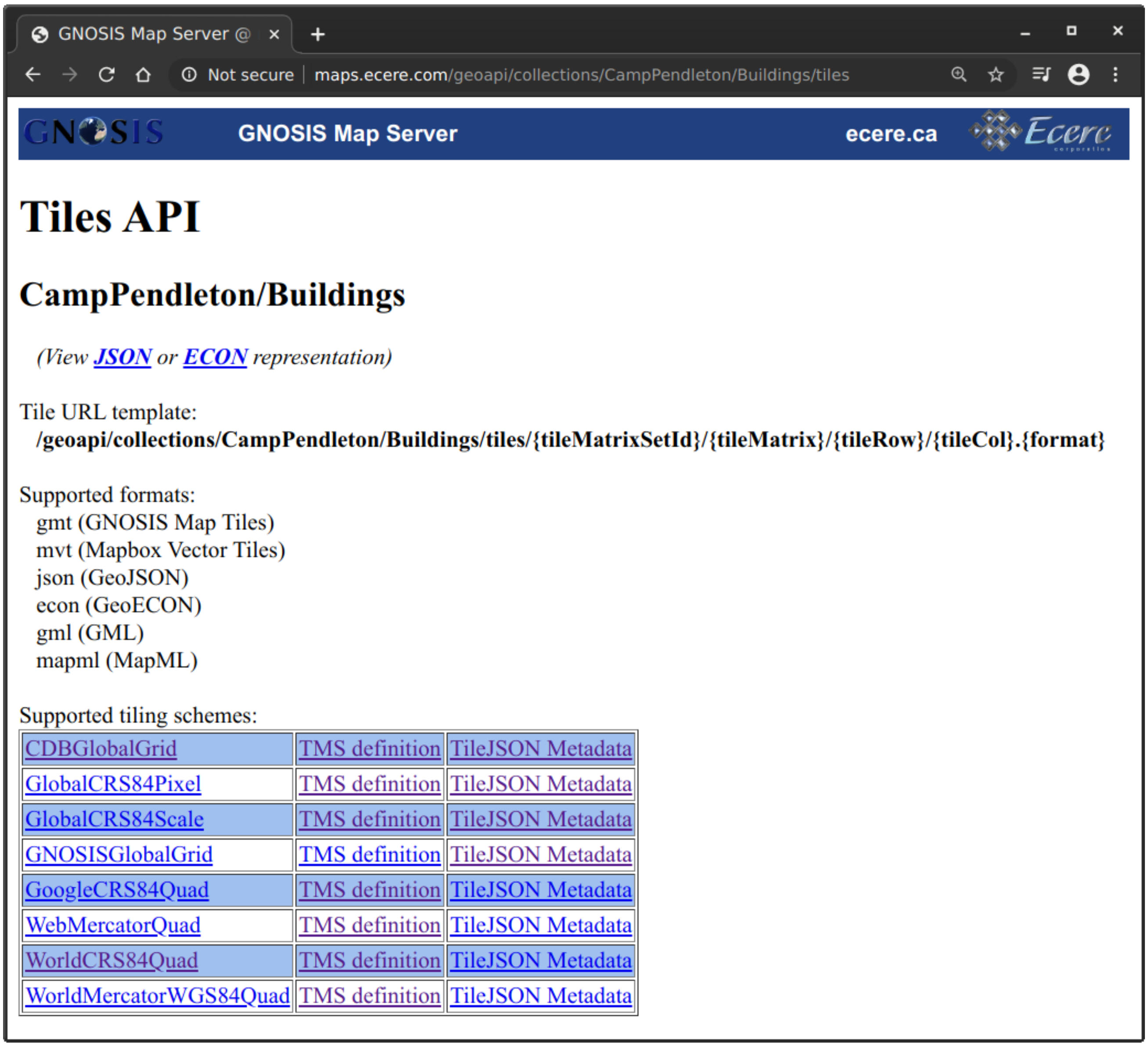
from Tree point features

(Download GeoJSON representation; view on geojson.io)

Geospatial extent: { { lat: 33.2601163, lon: -117.3136121 } - { lat: 33.2601163, lon: -117.3136121 } }

Property	Value
AO1	29.22
CNAM	EC030026-21-8U8R21-0
RTAI	100
SCALx	1.05298
SCALy	1.05298
SCALz	1.05298
BBH	16.612
BBL	14.963
BBW	14.963
BSR	13.451
CMIX	0
FACC	EC030
FSC	26
HGT	16.61
MODL	ASH

Figure 29. Accessing single feature for Camp Pendleton trees from dynamic server



Tiles API

CampPendleton/Buildings

*(View **JSON** or **ECON** representation)*

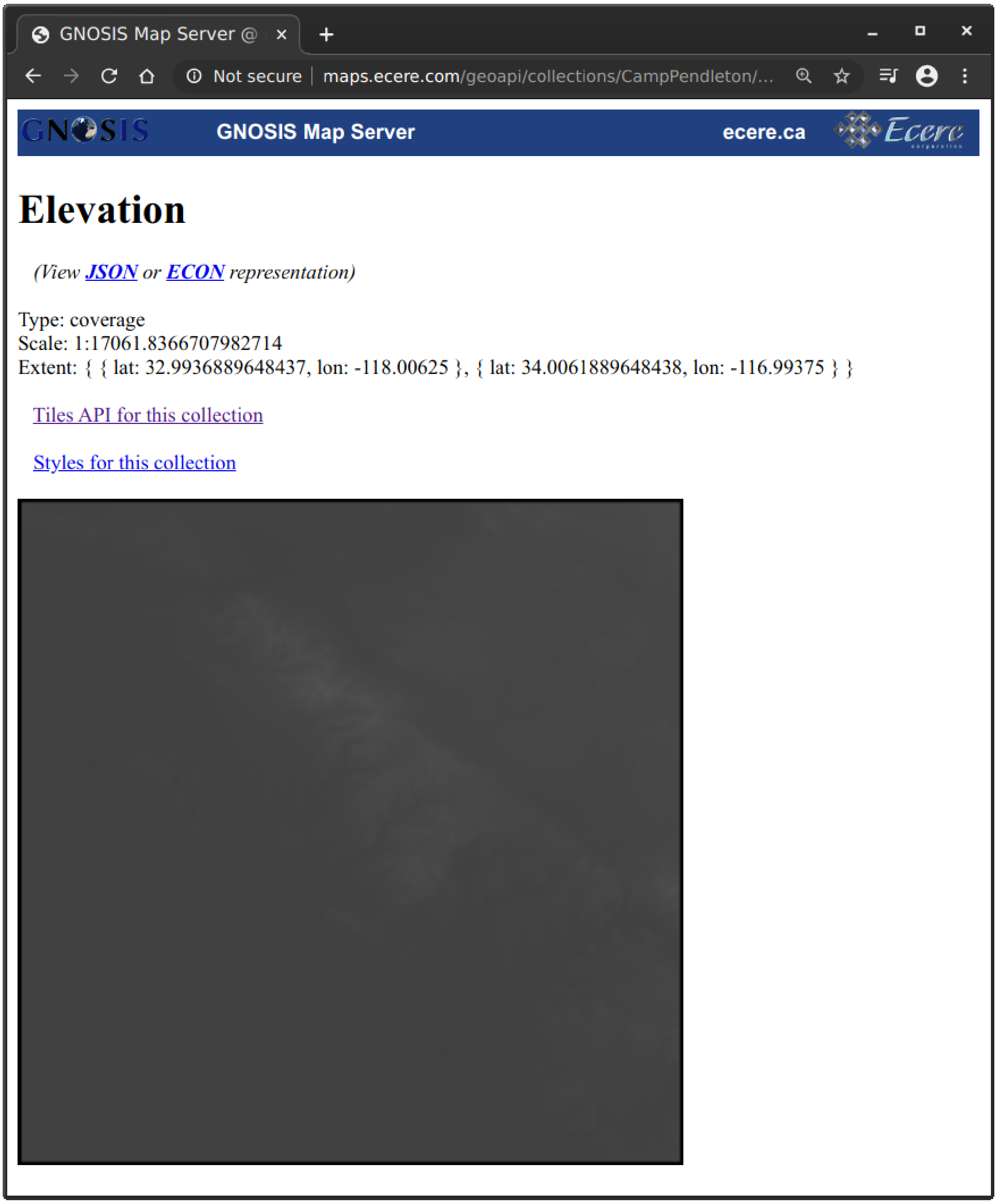
Tile URL template:
/geoapi/collections/CampPendleton/Buildings/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}.{format}

Supported formats:
gmt (GNOSIS Map Tiles)
mvt (Mapbox Vector Tiles)
json (GeoJSON)
econ (GeoECON)
gml (GML)
mapml (MapML)

Supported tiling schemes:

CDBGlobalGrid	TMS definition	TileJSON Metadata
GlobalCRS84Pixel	TMS definition	TileJSON Metadata
GlobalCRS84Scale	TMS definition	TileJSON Metadata
GNOSISGlobalGrid	TMS definition	TileJSON Metadata
GoogleCRS84Quad	TMS definition	TileJSON Metadata
WebMercatorQuad	TMS definition	TileJSON Metadata
WorldCRS84Quad	TMS definition	TileJSON Metadata
WorldMercatorWGS84Quad	TMS definition	TileJSON Metadata

Figure 30. Tiles API for Camp Pendleton buildings from dynamic server



GNOSIS Map Server @
Not secure | maps.ecere.com/geoapi/collections/CampPendleton/...
GNOSIS
GNOSIS Map Server
ecere.ca
Ecere

Elevation

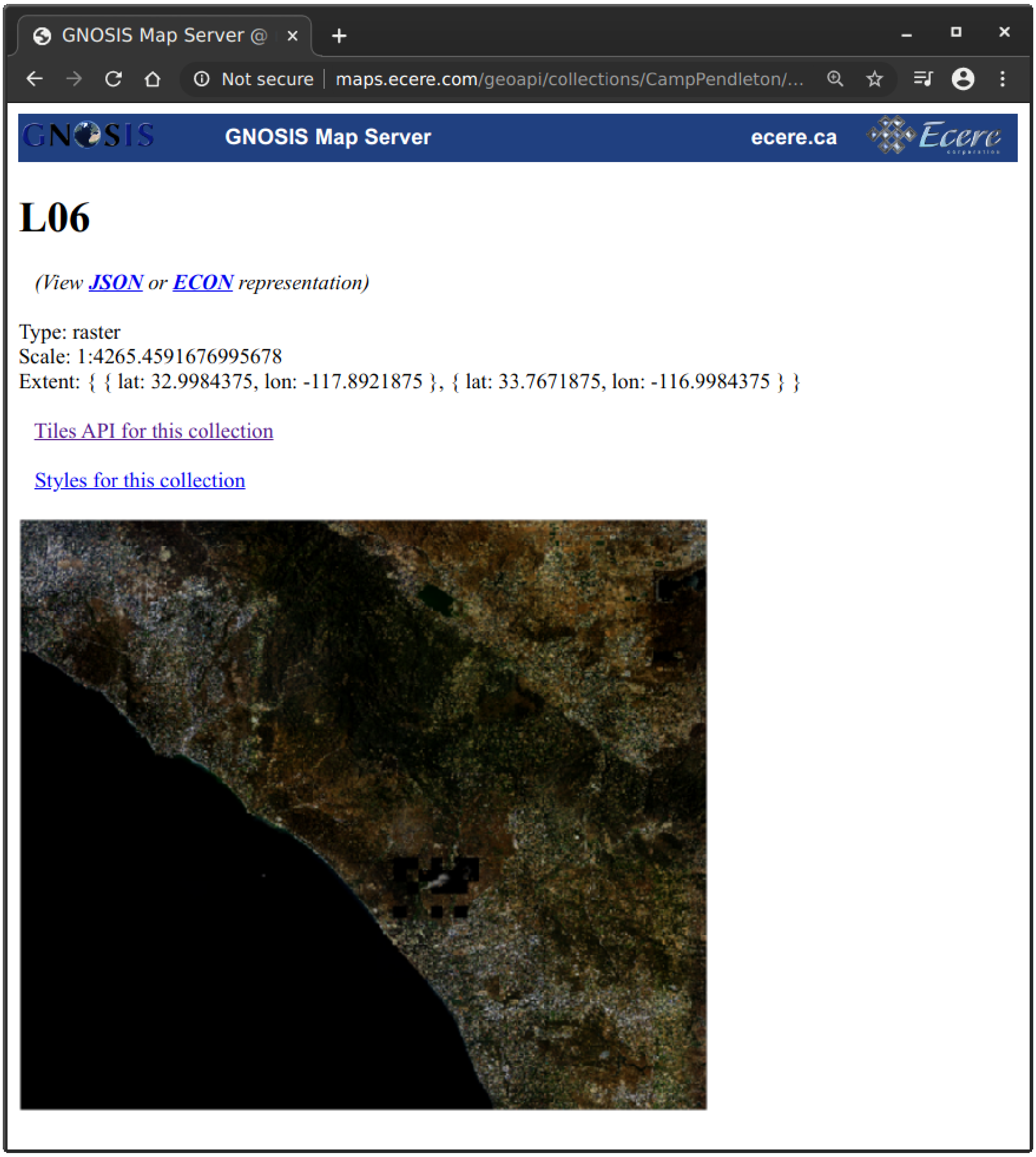
*(View **JSON** or **ECON** representation)*

Type: coverage
Scale: 1:17061.8366707982714
Extent: { { lat: 32.9936889648437, lon: -118.00625 }, { lat: 34.0061889648438, lon: -116.99375 } }

Tiles API for this collection

Styles for this collection

Figure 31. Data layer page for Camp Pendleton elevation data



L06

(View JSON or ECON representation)

Type: raster
Scale: 1:4265.4591676995678
Extent: { { lat: 32.9984375, lon: -117.8921875 }, { lat: 33.7671875, lon: -116.9984375 } }

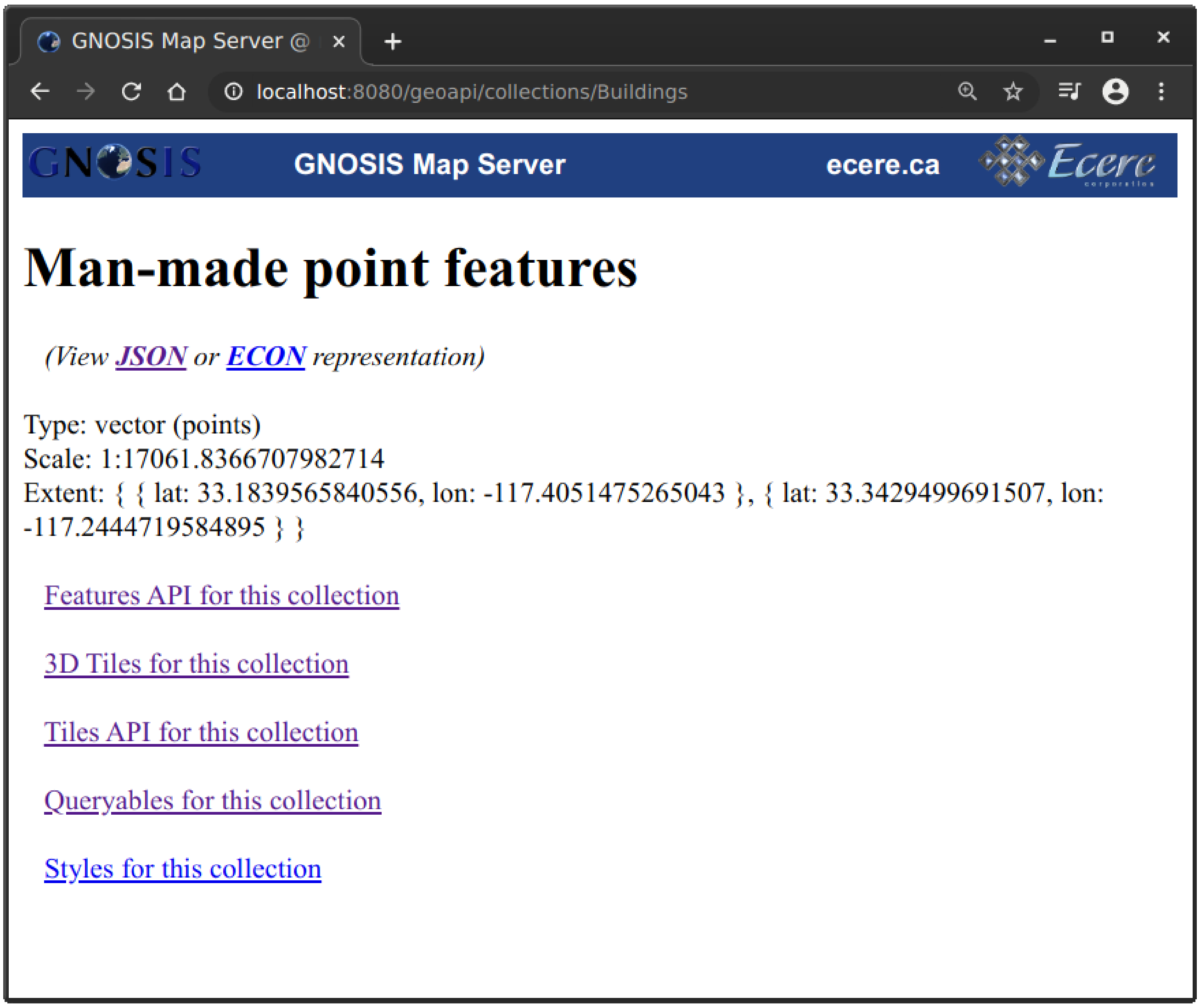
Tiles API for this collection

Styles for this collection

Figure 32. Data layer page for Camp Pendleton imagery data

9.4.1.3. Dynamic server - 3D Tiles

Additionally, work was initiated to provide access to this same data as 3D tiles, by generating batched 3D models and tilesets from the server. In this implementation, a fixed tiling scheme, the GNOSIS Global Grid tile matrix set, was used to partition the data, with the links to individual tiles being compatible with the Tiles API. Initial experiments accessing such tilesets were done with 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 the ability to load 3D datasets from CityGML sources into the GNOSIS Map Server, such as the Montreal dataset used in this initiative.



Man-made point features

(View JSON or ECON representation)

Type: vector (points)
Scale: 1:17061.8366707982714
Extent: { { lat: 33.1839565840556, lon: -117.4051475265043 }, { lat: 33.3429499691507, lon: -117.2444719584895 } }

Features API for this collection

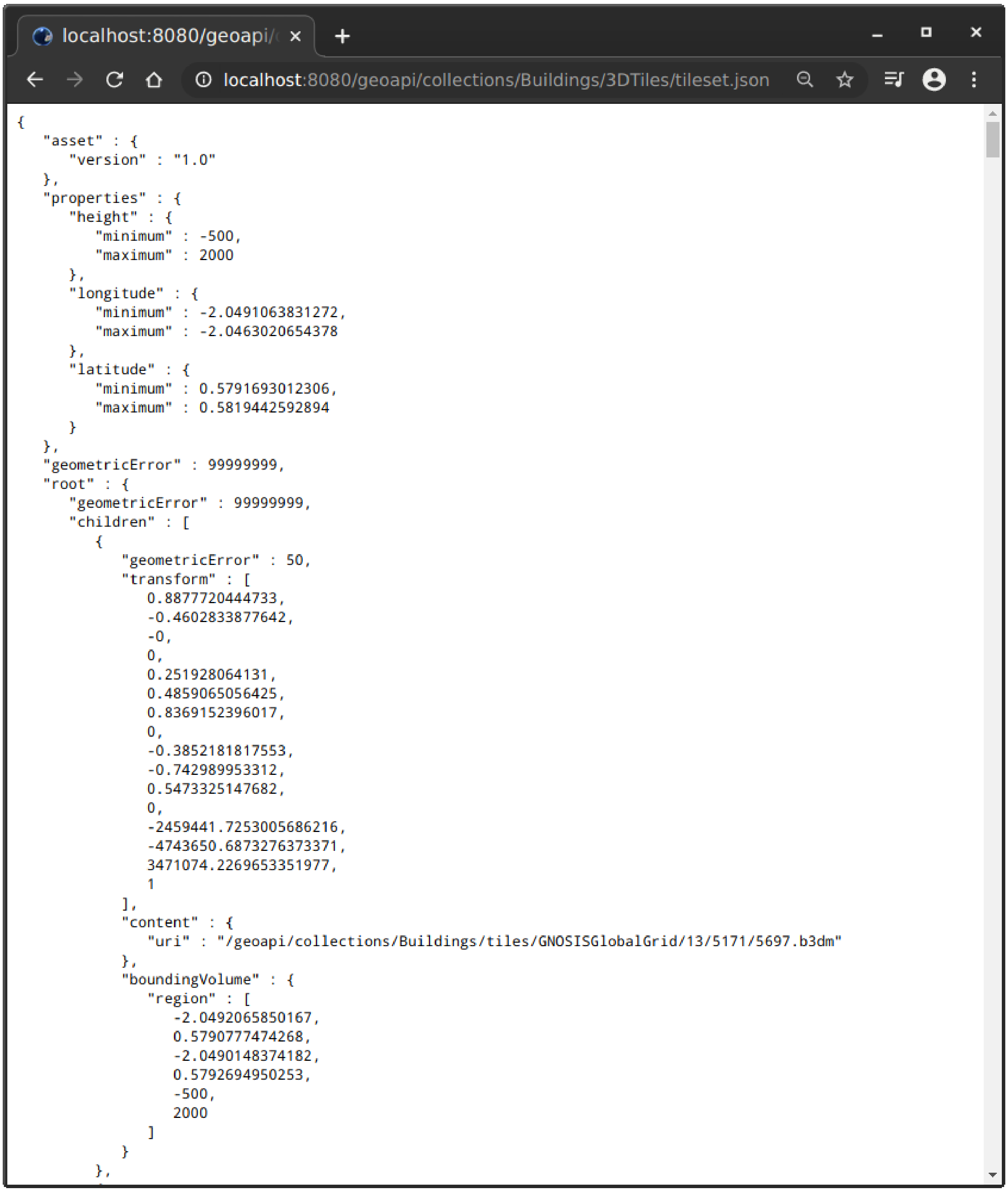
3D Tiles for this collection

Tiles API for this collection

Queryables for this collection

Styles for this collection

Figure 33. Data layer page for Camp Pendleton 3D buildings data



localhost:8080/geoapi/collections/Buildings/3DTiles/tileset.json

```
{
   "asset" : {
      "version" : "1.0"
   },
   "properties" : {
      "height" : {
         "minimum" : -500,
         "maximum" : 2000
      },
      "longitude" : {
         "minimum" : -2.0491063831272,
         "maximum" : -2.0463020654378
      },
      "latitude" : {
         "minimum" : 0.5791693012306,
         "maximum" : 0.5819442592894
      }
   },
   "geometricError" : 99999999,
   "root" : {
      "geometricError" : 99999999,
      "children" : [
         {
            "geometricError" : 50,
            "transform" : [
               0.8877720444733,
               -0.4602833877642,
               -0,
               0,
               0.251928064131,
               0.4859065056425,
               0.8369152396017,
               0,
               -0.3852181817553,
               -0.742989953312,
               0.5473325147682,
               0,
               -2459441.7253005686216,
               -4743650.6873276373371,
               3471074.2269653351977,
               1
            ],
            "content" : {
               "uri" : "/geoapi/collections/Buildings/tiles/GNOSISGlobalGrid/13/5171/5697.b3dm"
            },
            "boundingVolume" : {
               "region" : [
                  -2.0492065850167,
                  0.5790777474268,
                  -2.0490148374182,
                  0.5792694950253,
                  -500,
                  2000
               ]
            }
         },
```

Figure 34. 3D Tiles tileset resource Camp Pendleton 3D buildings data

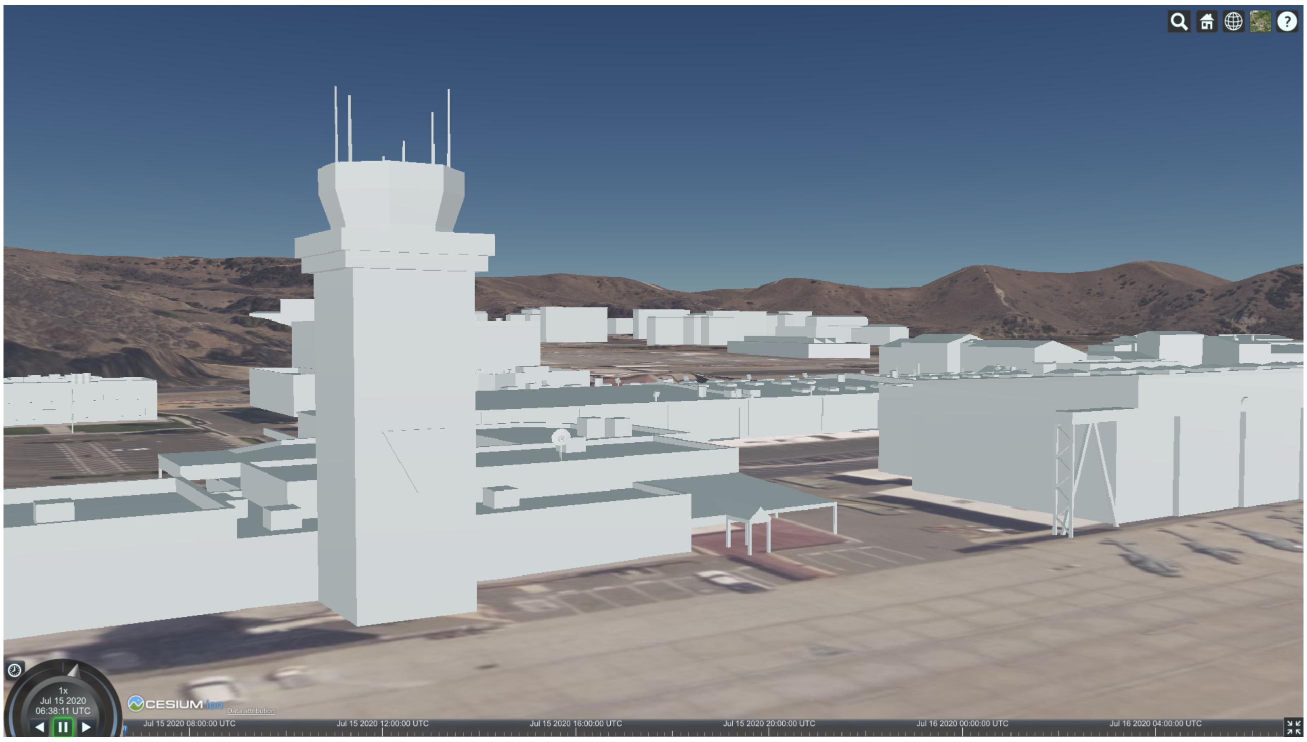


Figure 35. CesiumJS client accessing Camp Pendleton data from GNOSIS Map Server served as 3D Tiles

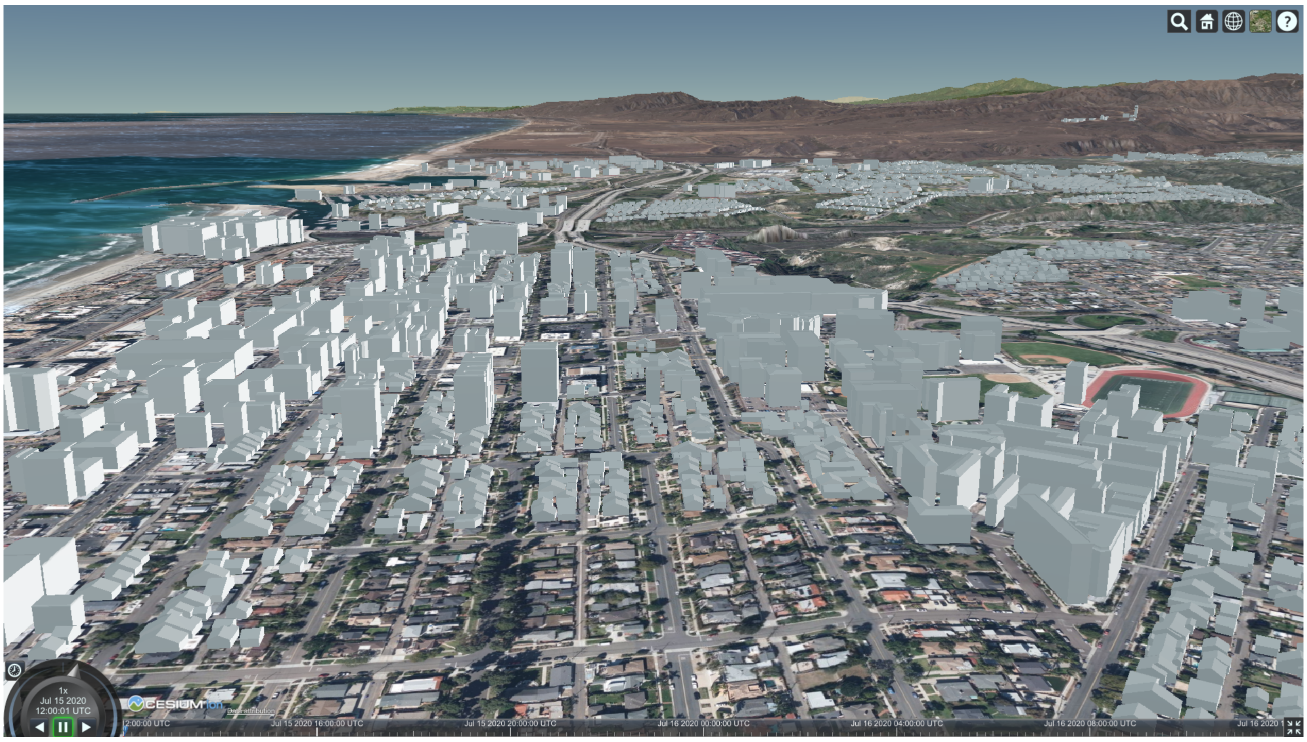


Figure 36. CesiumJS client accessing Camp Pendleton data from GNOSIS Map Server served as 3D Tiles (seaside)

9.4.2. Client

Ecere’s client component was based on its GNOSIS Cartographer cross-platform desktop GIS application.

Because the 3D data API used for the initiative leveraged the foundation of the upcoming OGC API - Common standards, existing support for resources such as the landing page, listing available data layers and directly accessing a specific geospatial data resource could be re-used. This functionality works the same as if accessing other types of data resources, such as vector features or raster coverages.

Those resources allow discovering available data from an OGC API end-point, and provide links to access the data organized. In the case of 3D Tiles and I3S, the 3D data is distributed organized as a Bounding Volume Hierarchy. Experiments were also done using the Tiles API.

9.4.2.1. Tiles API

The client also supports accessing 3D data through the Tiles API, accessing multi-resolution tiles of imagery, terrain elevation, as well as vector data tiles containing points referencing 3D models with position and orientation information. These tiles follow a regular tiling scheme based on the OGC Two Dimensional Tile Matrix Set 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.

The data used in this experiment originated from a sample CDB dataset of Camp Pendleton from the CDB Starter Kit. The GNOSIS engine tessellates a 3D mesh of the terrain on the fly from the elevation gridded coverage tiles.

Ecere’s compact E3D [http://docs.opengeospatial.org/per/18-025.html#E3DSpecs] format was used to transfer the models to the client, but support for serving the models as glTF was also implemented.

As trees for this dataset are defined as "geotypical", with only few models re-used at thousands of location, they are rendered with a single drawing call using model instantiation techniques.

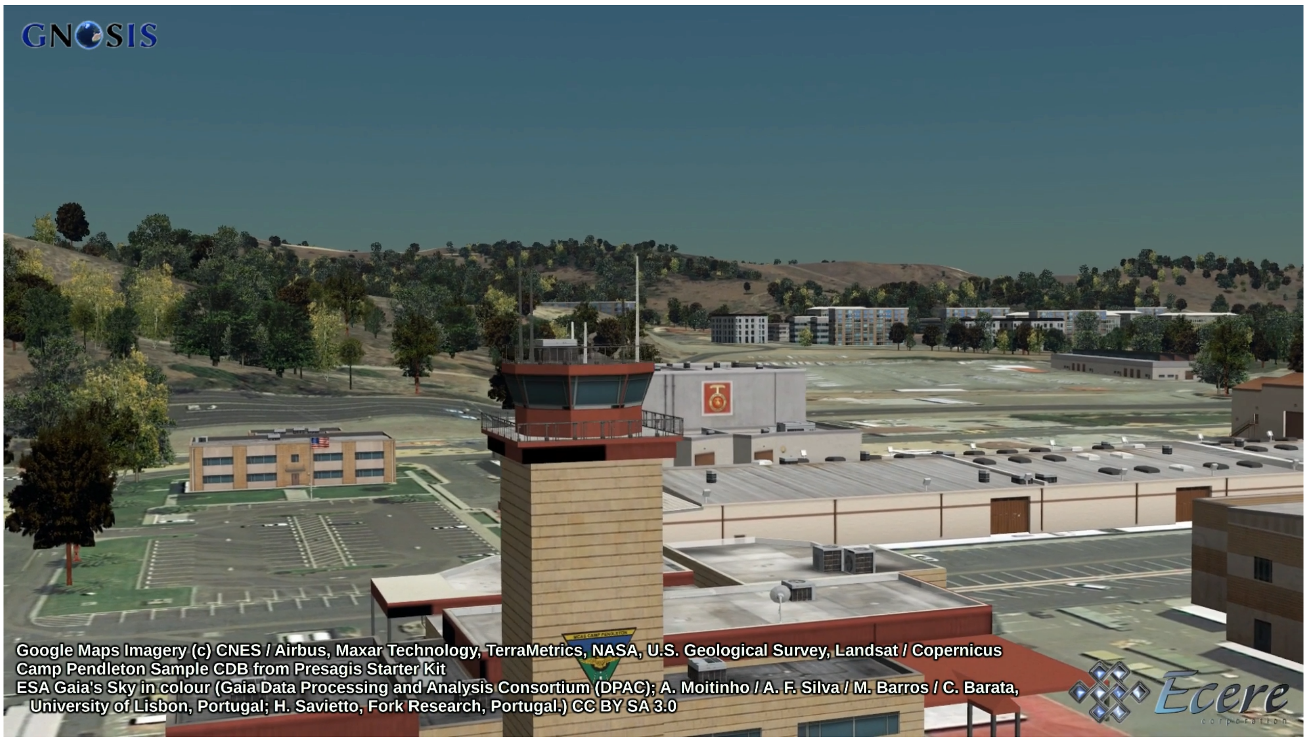

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Camp Pendleton Sample CDB from Presagis Starter Kit
ESA Gaia's Sky in colour (Gaia Data Processing and Analysis Consortium (DPAC); A. Moitinho / A. F. Silva / M. Barros / C. Barata, University of Lisbon, Portugal; H. Savietto, Fork Research, Portugal.) CC BY SA 3.0
Ecere

Figure 37. Camp Pendleton CDB dataset accessed from GNOSIS Map Server using the Tiles API

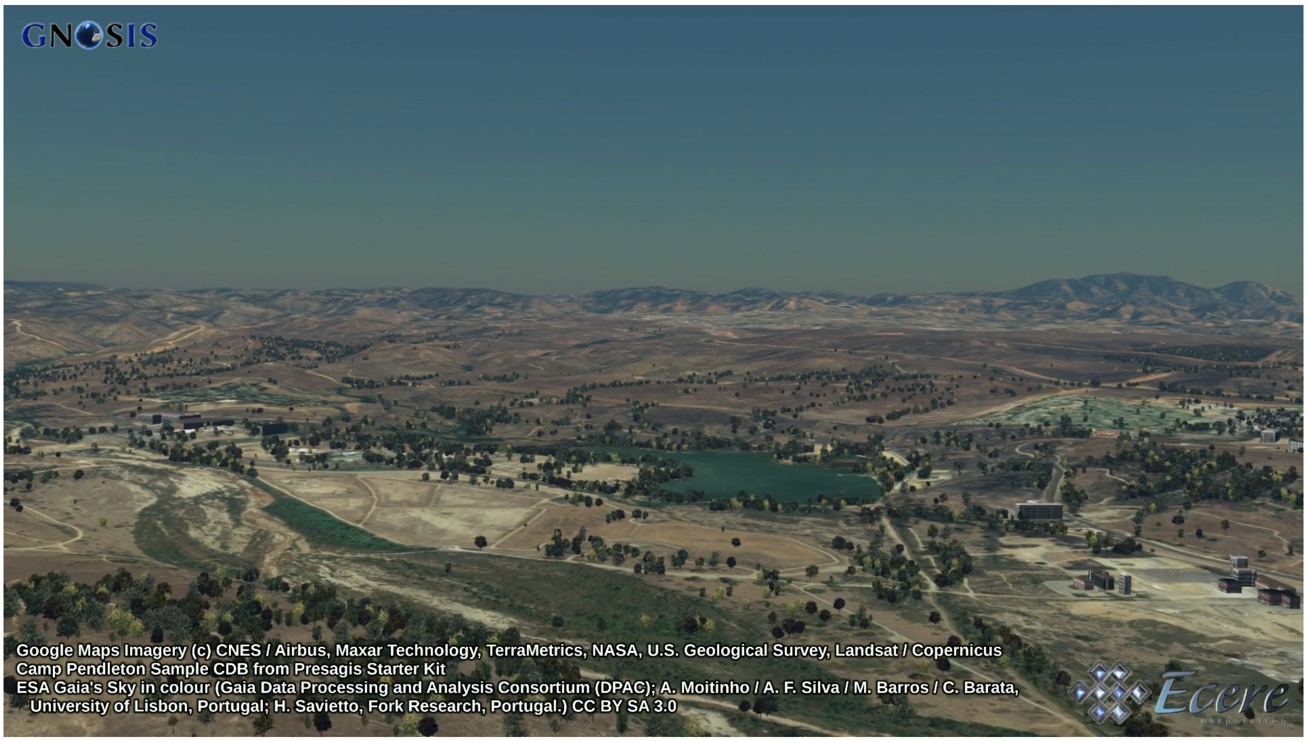

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Camp Pendleton Sample CDB from Presagis Starter Kit
ESA Gaia's Sky in colour (Gaia Data Processing and Analysis Consortium (DPAC); A. Moitinho / A. F. Silva / M. Barros / C. Barata, University of Lisbon, Portugal; H. Savietto, Fork Research, Portugal.) CC BY SA 3.0
Ecere

Figure 38. Camp Pendleton CDB dataset accessed from GNOSIS Map Server using the Tiles API

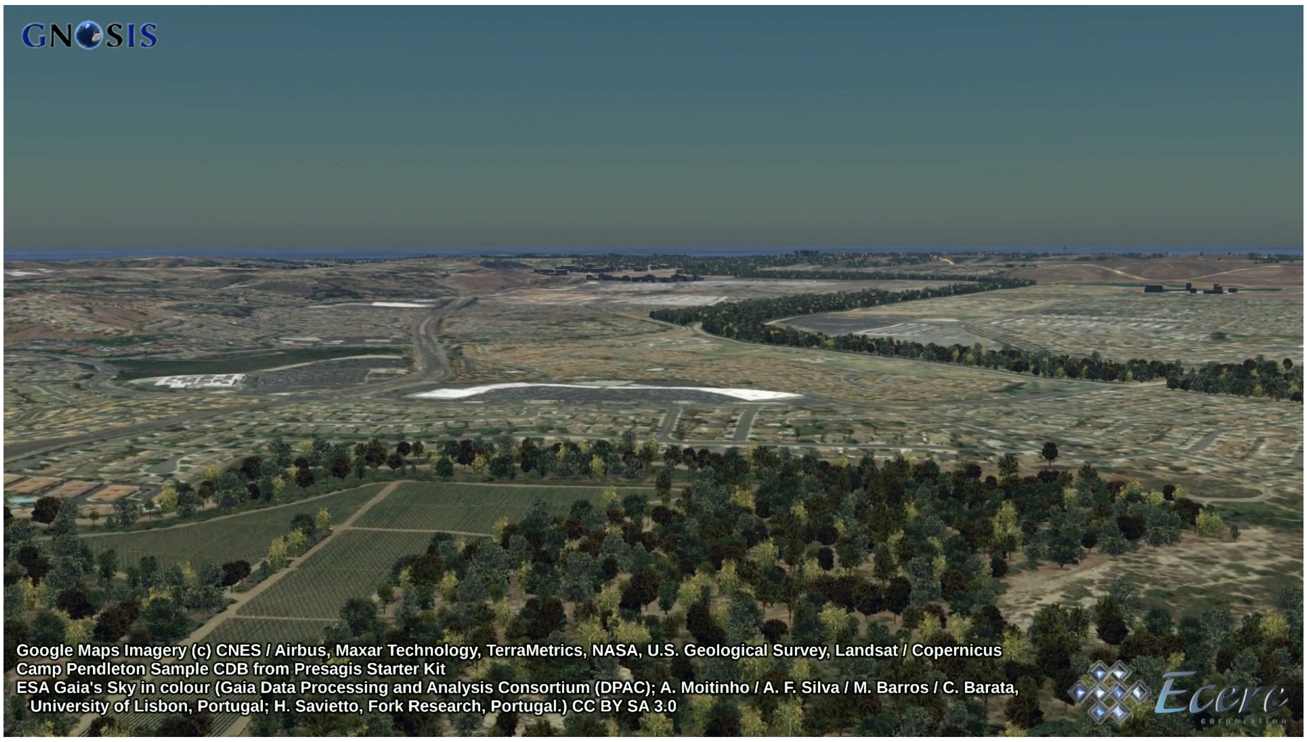

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Camp Pendleton Sample CDB from Presagis Starter Kit
ESA Gaia's Sky in colour (Gaia Data Processing and Analysis Consortium (DPAC); A. Moitinho / A. F. Silva / M. Barros / C. Barata, University of Lisbon, Portugal; H. Savietto, Fork Research, Portugal.) CC BY SA 3.0
Ecere

Figure 39. Camp Pendleton CDB dataset accessed from GNOSIS Map Server using the Tiles API

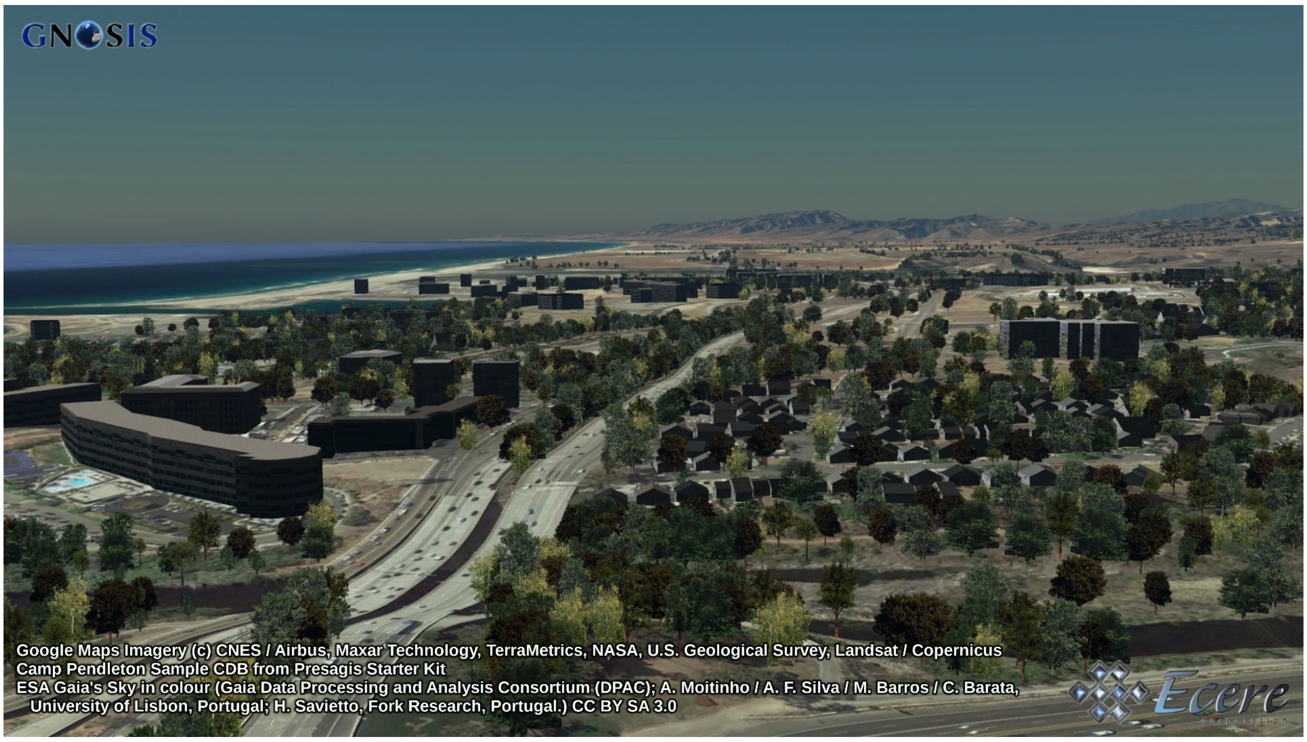

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Camp Pendleton Sample CDB from Presagis Starter Kit
ESA Gaia's Sky in colour (Gaia Data Processing and Analysis Consortium (DPAC); A. Moitinho / A. F. Silva / M. Barros / C. Barata, University of Lisbon, Portugal; H. Savietto, Fork Research, Portugal.) CC BY SA 3.0
Ecere

Figure 40. Camp Pendleton CDB dataset accessed from GNOSIS Map Server using the Tiles API

9.4.2.2. 3D Tiles

Support for the OGC 3D Tiles Community Standard was implemented in the latest version of Ecere’s GNOSIS cross-platform Software Development Kit during the pilot. This capability to visualize 3D Tiles was tested along with the other OGC API resources by performing Technology Integration Experiments with services from other participants of the pilot. The Ecere client could access all data from other services being provided as batched 3D models (.b3dm) 3D Tiles.

A number of datasets was offered by different participants. One dataset of New York City buildings was being served by all. Other content such as imagery and terrain elevation was also integrated within the view using a 2D Tiles API.

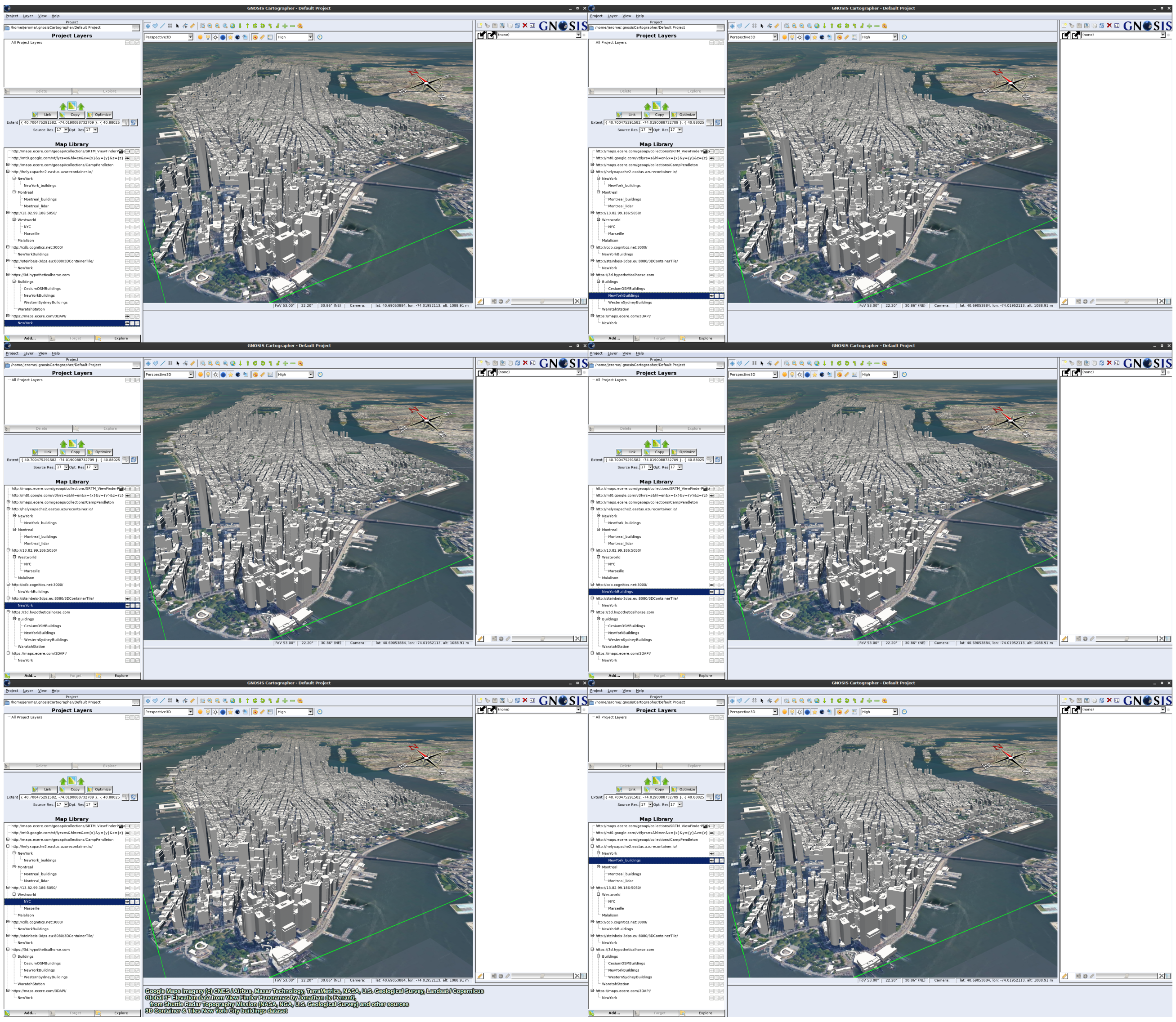

GNOSIS Cartographer - Default Project
Project Layers
Map Library
FoV 53.00° | 22.20° | 30.86° (NE) | Camera: | lat: 40.69053884, lon: -74.01952113, alt: 1088.91 m
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Global 3" Elevation data from View Finder Panoramas by Jonathan de Ferranti,
from Shuttle Radar Topography Mission (NASA, NGA, U.S. Geological Survey) and other sources
3D Container & Tiles New York City buildings dataset

Figure 41. New York City dataset being accessed as 3D Tiles from all participants services

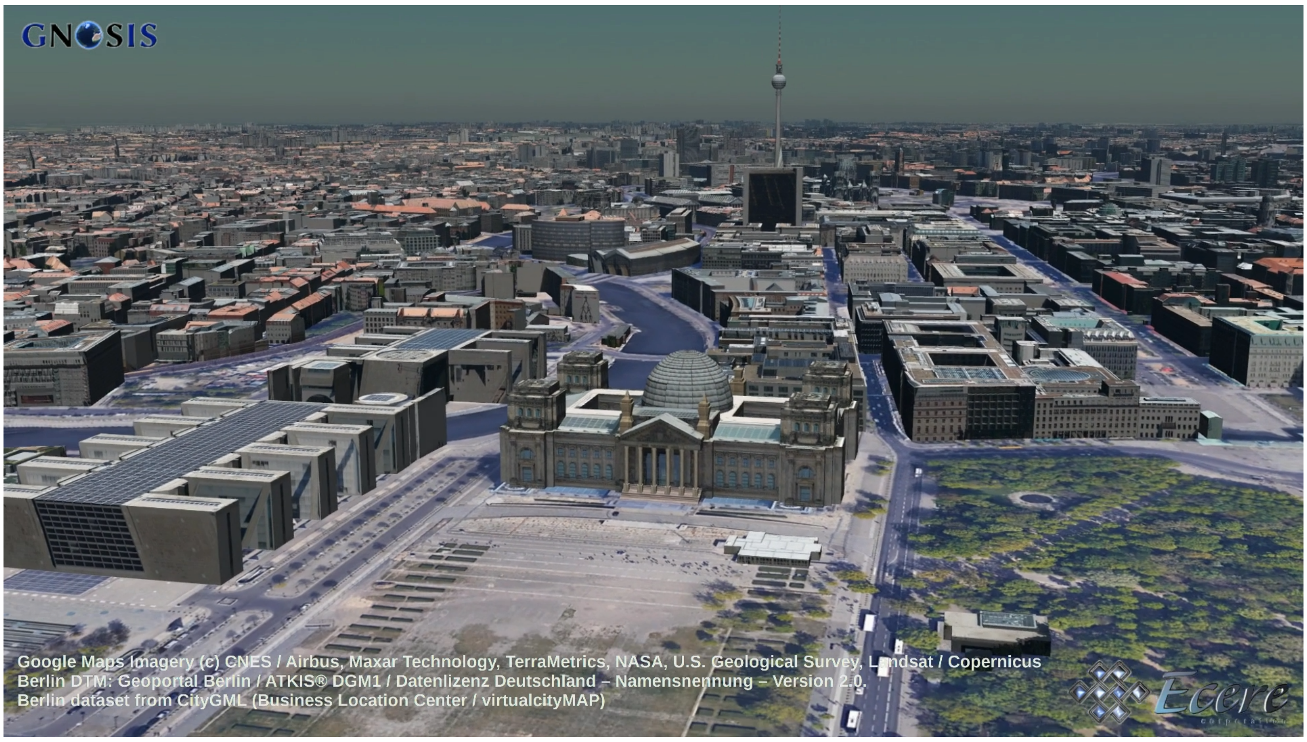

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Berlin DTM: Geoportal Berlin / ATKIS® DGM1 / Datenlizenz Deutschland – Namensnennung – Version 2.0.
Berlin dataset from CityGML (Business Location Center / virtualcityMAP)
Ecere

Figure 42. Textured dataset of Berlin originating from CityGML being served as 3D Tiles by virtualcityMAP from virtualcitySYSTEMS (additional non-participating service tests)

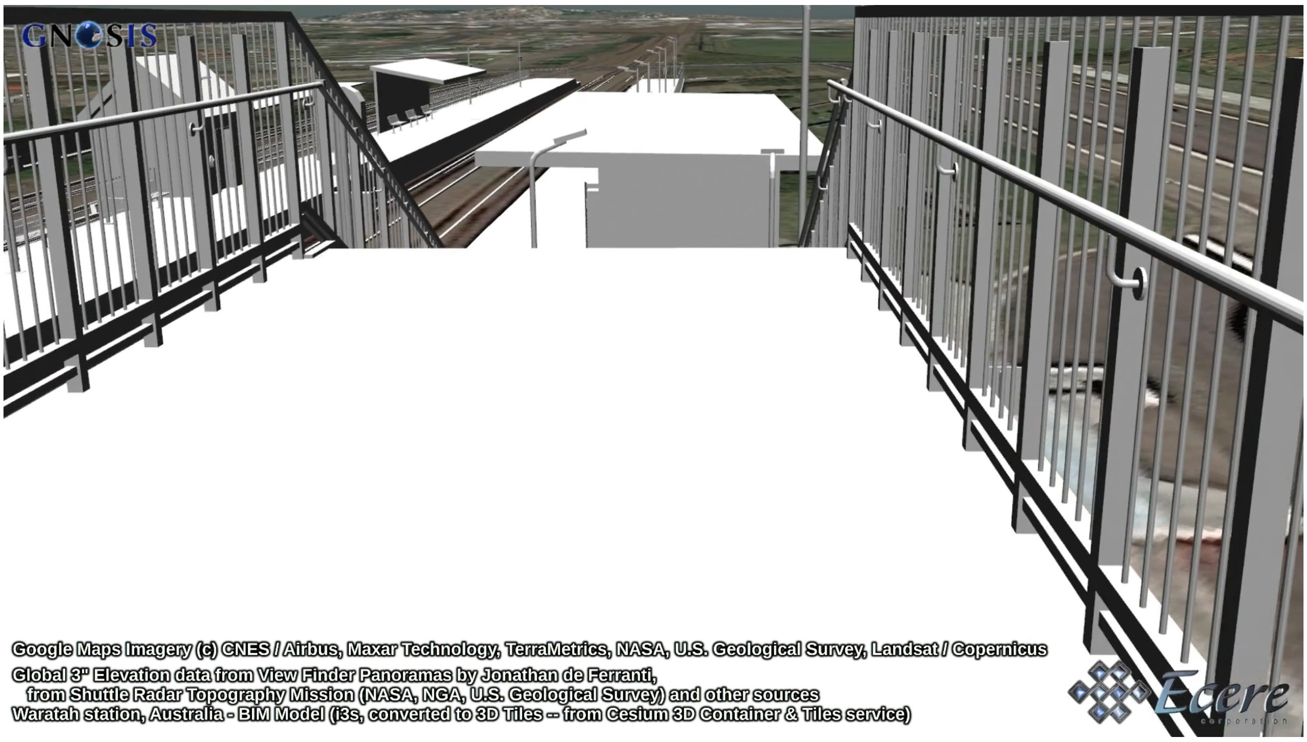

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Global 3" Elevation data from View Finder Panoramas by Jonathan de Ferranti,
from Shuttle Radar Topography Mission (NASA, NGA, U.S. Geological Survey) and other sources
Waratah station, Australia - BIM Model (i3s, converted to 3D Tiles -- from Cesium 3D Container & Tiles service)
Ecere

Figure 43. BIM dataset of the Waratah railway station in New South Wales, Australia accessed from Cesium service, streamed as i3s and converted to 3D Tiles

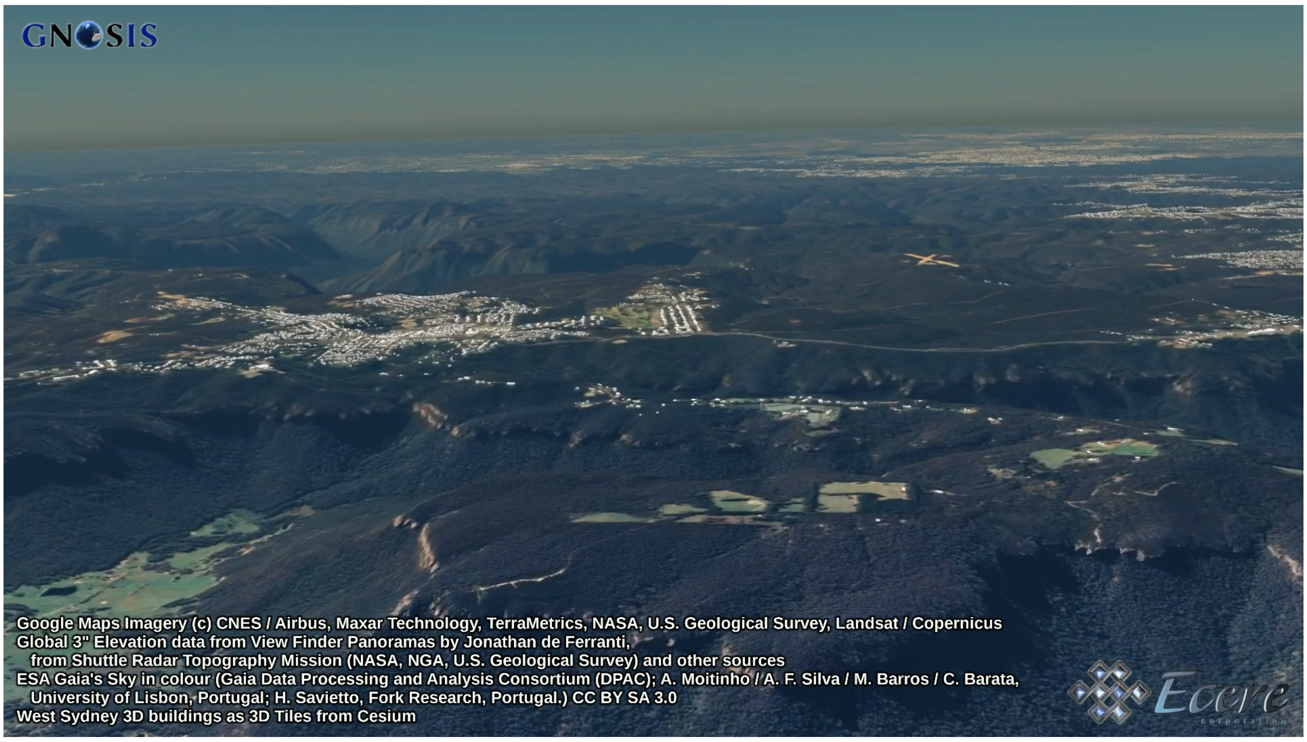

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Global 3" Elevation data from View Finder Panoramas by Jonathan de Ferranti,
from Shuttle Radar Topography Mission (NASA, NGA, U.S. Geological Survey) and other sources
ESA Gaia's Sky in colour (Gaia Data Processing and Analysis Consortium (DPAC); A. Moitinho / A. F. Silva / M. Barros / C. Barata,
University of Lisbon, Portugal; H. Savietto, Fork Research, Portugal.) CC BY SA 3.0
West Sydney 3D buildings as 3D Tiles from Cesium
Ecere

Figure 44. 3D buildings of West Sydney accessed as 3D Tiles from Cesium service

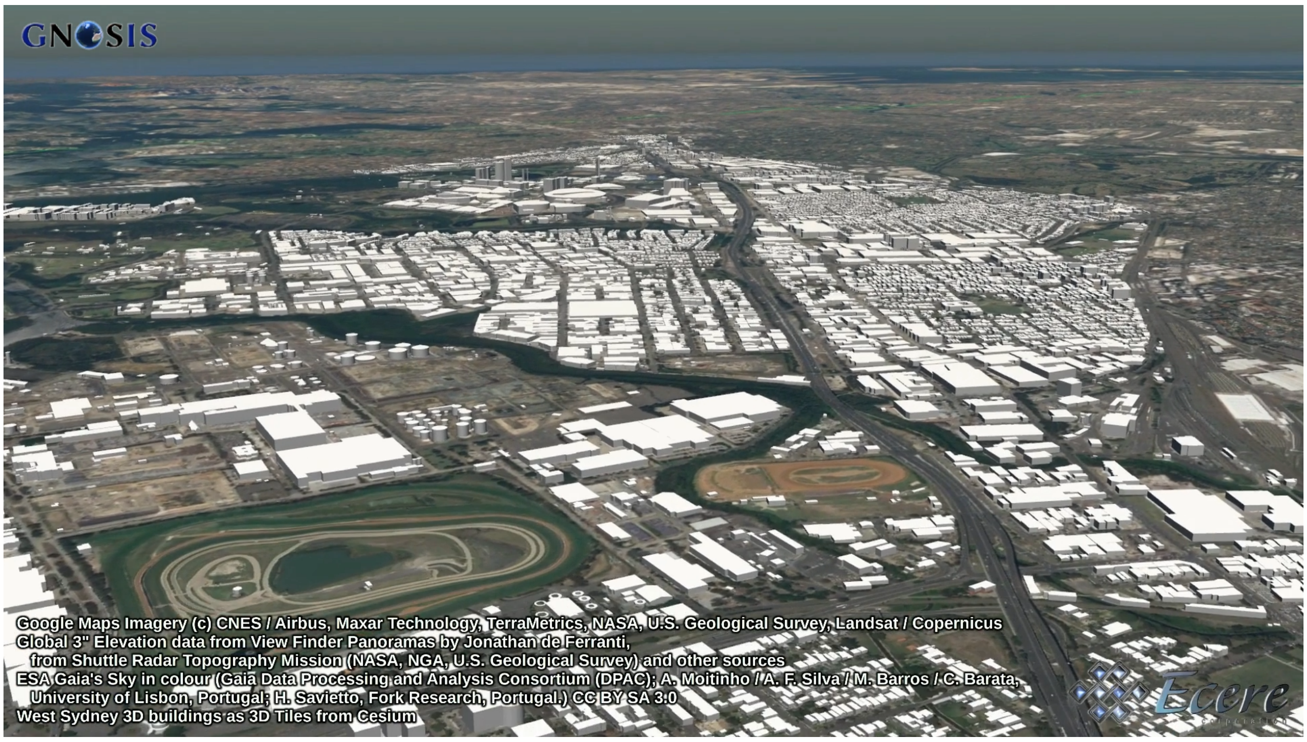

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Global 3" Elevation data from View Finder Panoramas by Jonathan de Ferranti,
from Shuttle Radar Topography Mission (NASA, NGA, U.S. Geological Survey) and other sources
ESA Gaia's Sky in colour (Gaia Data Processing and Analysis Consortium (DPAC); A. Moitinho / A. F. Silva / M. Barros / C. Barata,
University of Lisbon, Portugal; H. Savietto, Fork Research, Portugal.) CC BY SA 3.0
West Sydney 3D buildings as 3D Tiles from Cesium
Ecere

Figure 45. 3D buildings of West Sydney accessed as 3D Tiles from Cesium service (larger buildings)

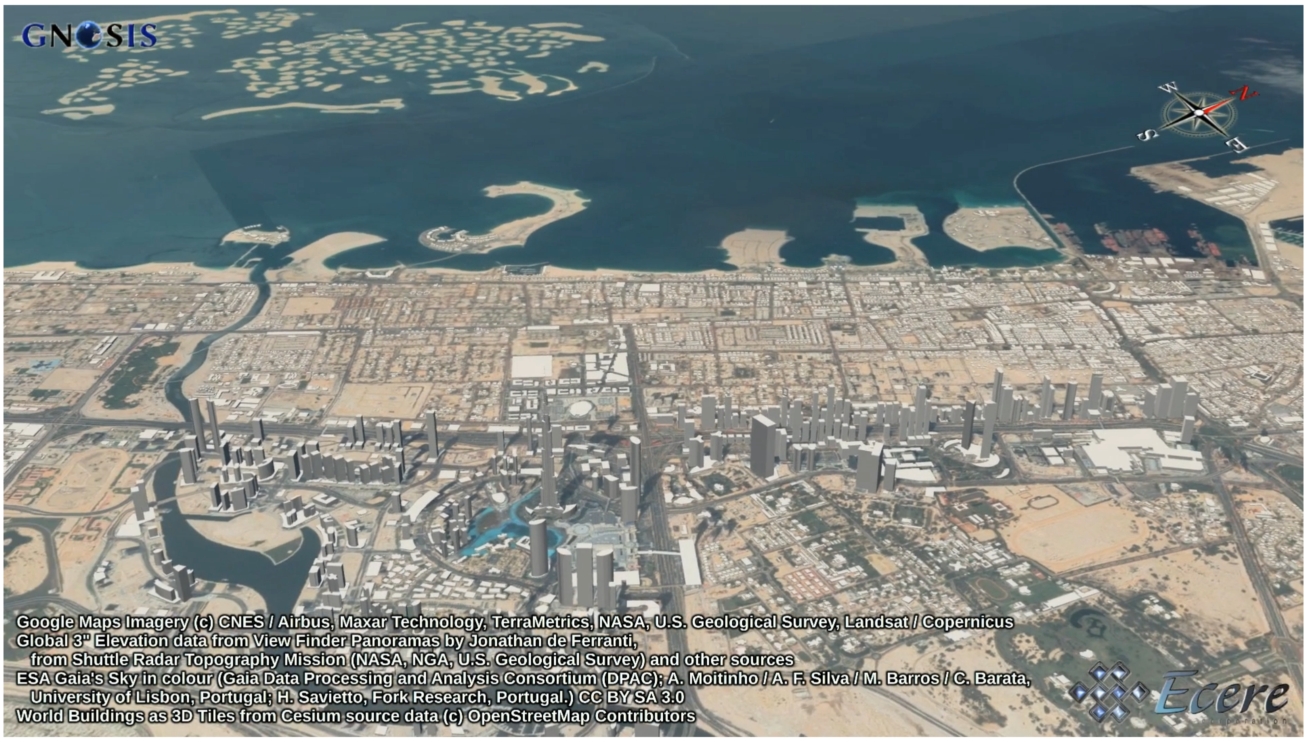

GNOSIS
W
N
S
E
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Global 3" Elevation data from View Finder Panoramas by Jonathan de Ferranti,
from Shuttle Radar Topography Mission (NASA, NGA, U.S. Geological Survey) and other sources
ESA Gaia's Sky in colour (Gaia Data Processing and Analysis Consortium (DPAC); A. Moitinho / A. F. Silva / M. Barros / C. Barata,
University of Lisbon, Portugal; H. Savietto, Fork Research, Portugal.) CC BY SA 3.0
World Buildings as 3D Tiles from Cesium source data (c) OpenStreetMap Contributors
Ecere

Figure 46. Worldwide buildings from OpenStreetMap accessed as 3D Tiles from Cesium service (Dubai)

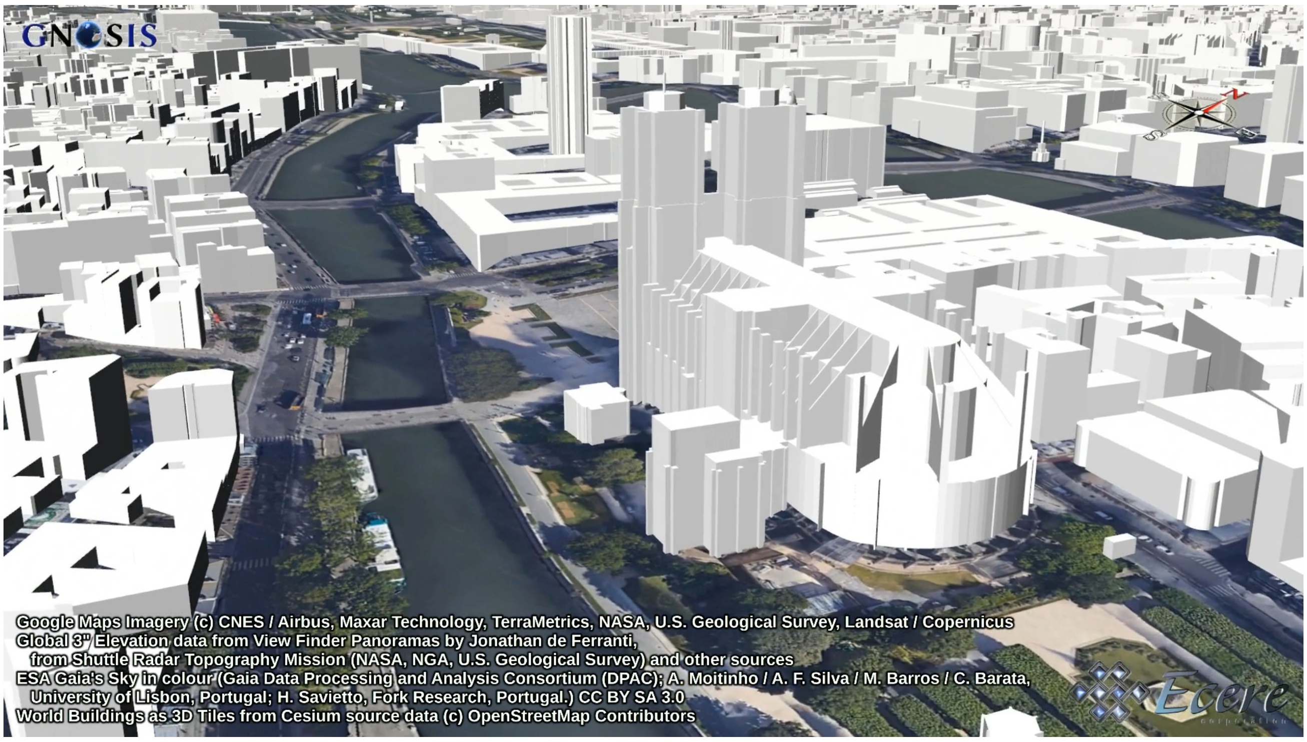

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Global 3" Elevation data from View Finder Panoramas by Jonathan de Ferranti,
from Shuttle Radar Topography Mission (NASA, NGA, U.S. Geological Survey) and other sources
ESA Gaia's Sky in colour (Gaia Data Processing and Analysis Consortium (DPAC); A. Moitinho / A. F. Silva / M. Barros / C. Barata,
University of Lisbon, Portugal; H. Savietto, Fork Research, Portugal.) CC BY SA 3.0
World Buildings as 3D Tiles from Cesium source data (c) OpenStreetMap Contributors
Ecere

Figure 47. Worldwide buildings from OpenStreetMap accessed as 3D Tiles from Cesium service (Paris)

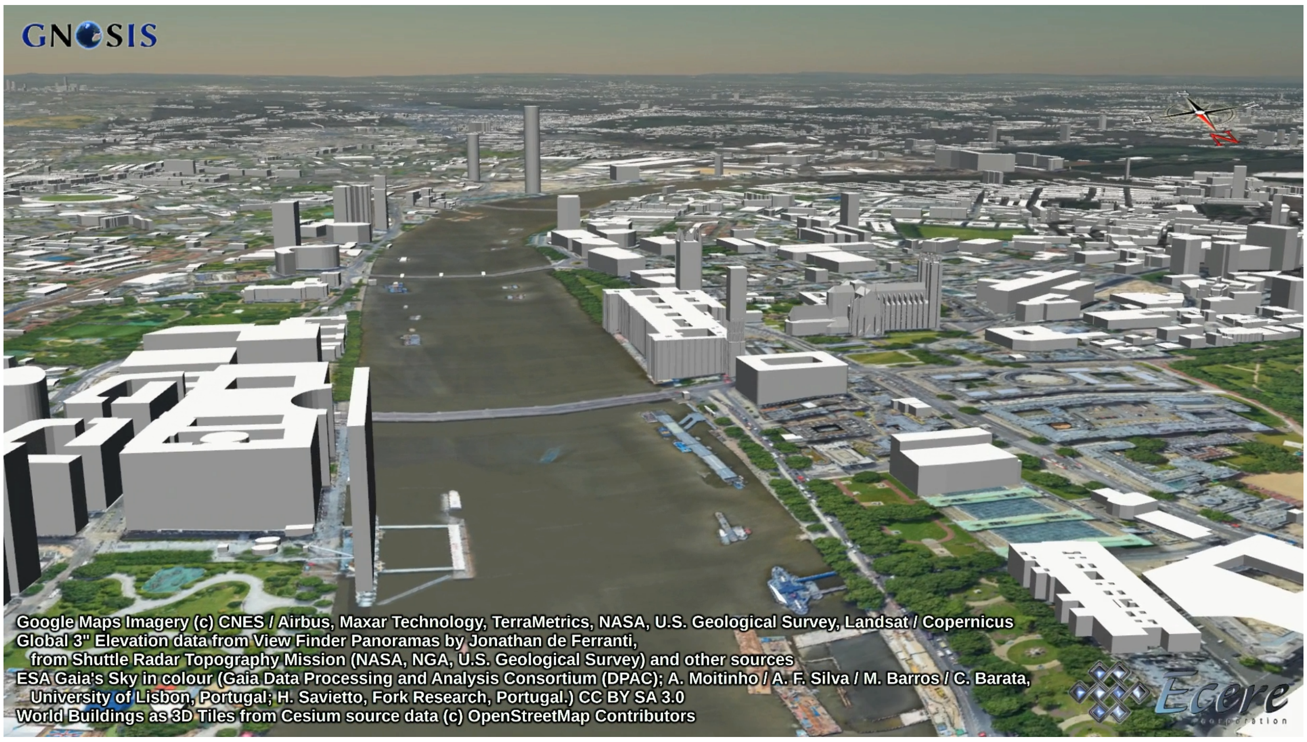

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Global 3" Elevation data from View Finder Panoramas by Jonathan de Ferranti,
from Shuttle Radar Topography Mission (NASA, NGA, U.S. Geological Survey) and other sources
ESA Gaia's Sky in colour (Gaia Data Processing and Analysis Consortium (DPAC); A. Moitinho / A. F. Silva / M. Barros / C. Barata,
University of Lisbon, Portugal; H. Savietto, Fork Research, Portugal.) CC BY SA 3.0
World Buildings as 3D Tiles from Cesium source data (c) OpenStreetMap Contributors
Ecere corporation

Figure 48. Worldwide buildings from OpenStreetMap accessed as 3D Tiles from Cesium service (London)

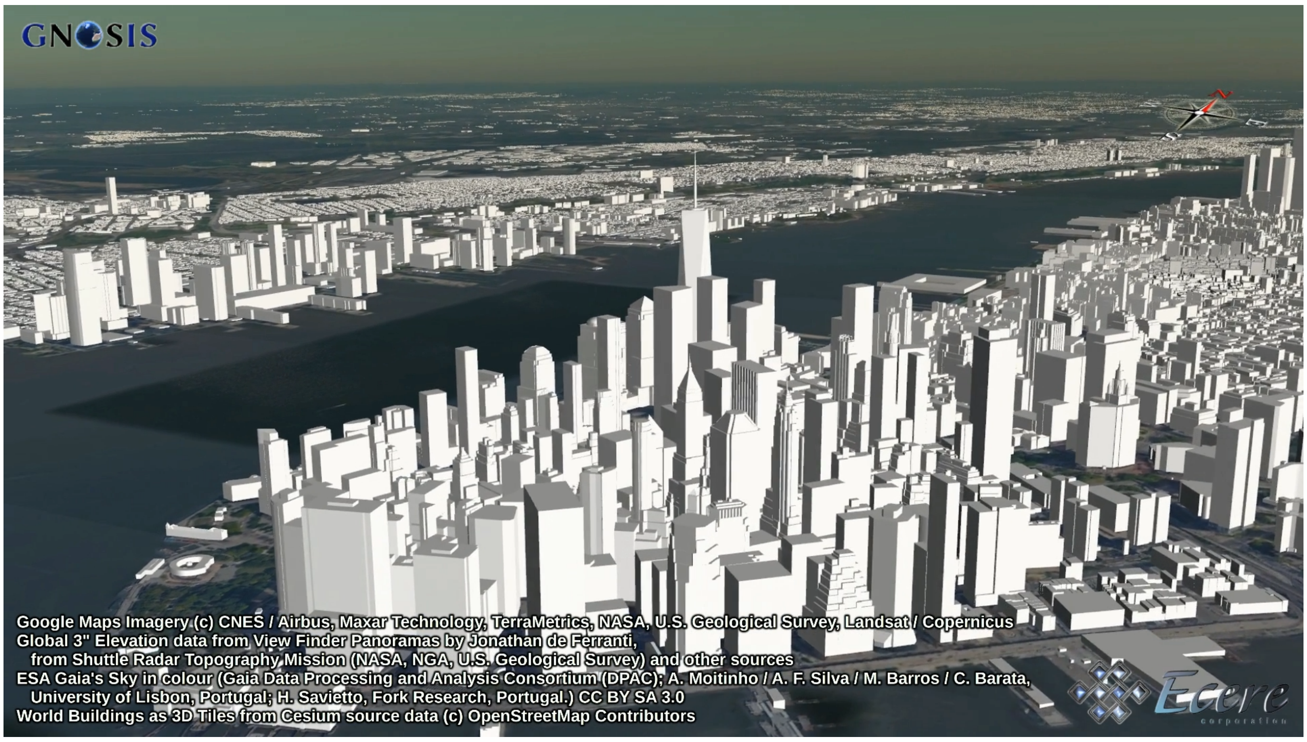

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Global 3" Elevation data from View Finder Panoramas by Jonathan de Ferranti,
from Shuttle Radar Topography Mission (NASA, NGA, U.S. Geological Survey) and other sources
ESA Gaia's Sky in colour (Gaia Data Processing and Analysis Consortium (DPAC); A. Moitinho / A. F. Silva / M. Barros / C. Barata,
University of Lisbon, Portugal; H. Savietto, Fork Research, Portugal.) CC BY SA 3.0
World Buildings as 3D Tiles from Cesium source data (c) OpenStreetMap Contributors
Ecere corporation

Figure 49. Worldwide buildings from OpenStreetMap accessed as 3D Tiles from Cesium service (New York)

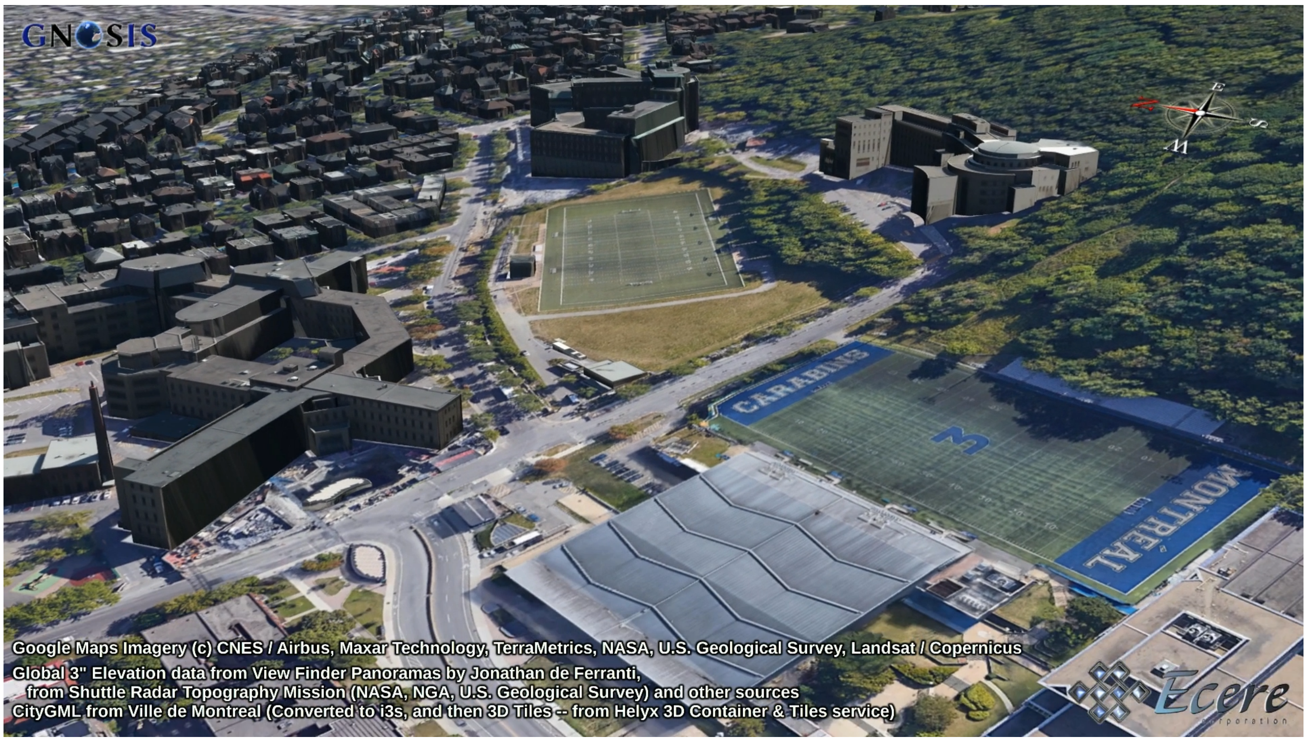

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Global 3" Elevation data from View Finder Panoramas by Jonathan de Ferranti,
from Shuttle Radar Topography Mission (NASA, NGA, U.S. Geological Survey) and other sources
CityGML from Ville de Montreal (Converted to i3s, and then 3D Tiles -- from Helyx 3D Container & Tiles service)
CARABINS
MONTRÉAL
Ecere corporation

Figure 50. Montreal buildings originating from CityGML, served as I3S from Helyx Service, and converted as 3D Tiles

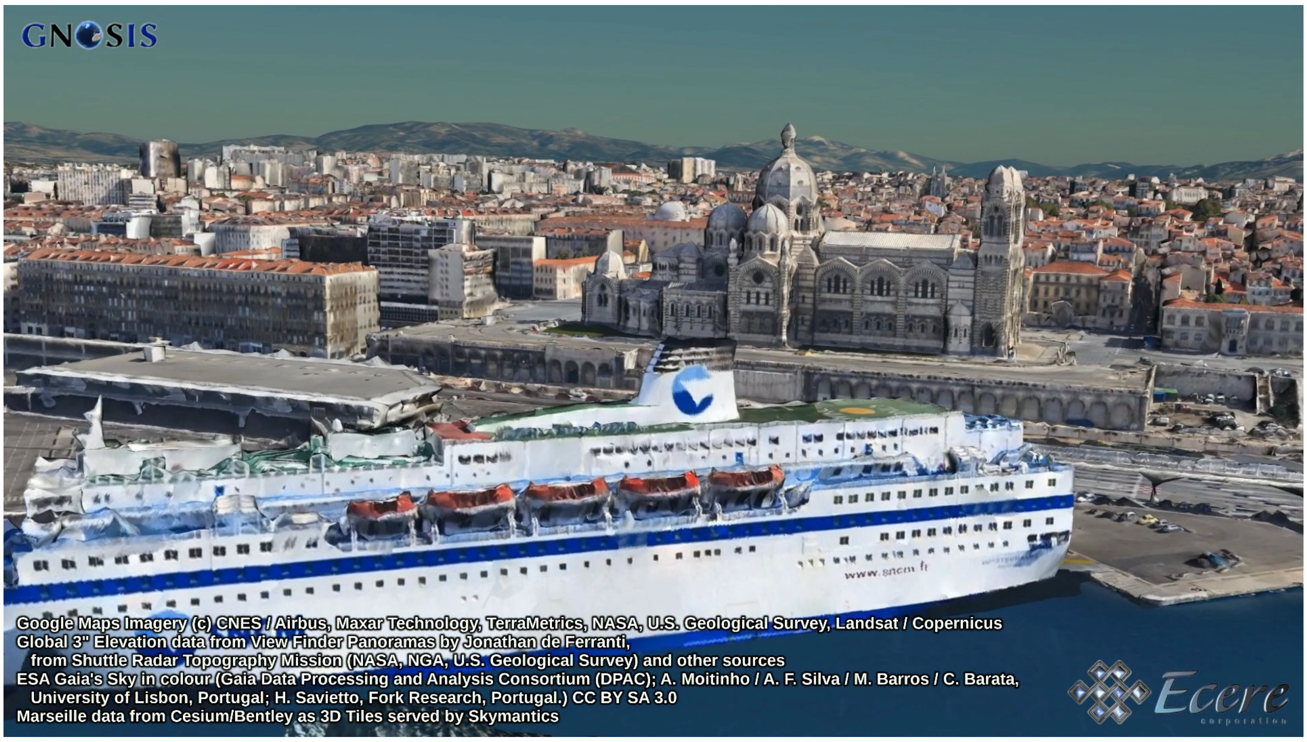

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Global 3" Elevation data from View Finder Panoramas by Jonathan de Ferranti,
from Shuttle Radar Topography Mission (NASA, NGA, U.S. Geological Survey) and other sources
ESA Gaia's Sky in colour (Gaia Data Processing and Analysis Consortium (DPAC); A. Moitinho / A. F. Silva / M. Barros / C. Barata,
University of Lisbon, Portugal; H. Savietto, Fork Research, Portugal.) CC BY SA 3.0
Marseille data from Cesium/Bentley as 3D Tiles served by Skymantics
www.sncm.fr
Ecere corporation

Figure 51. Dataset of Marseille provided by Bentley & Cesium being served as 3D Tiles from the Skymantics service (La Major)

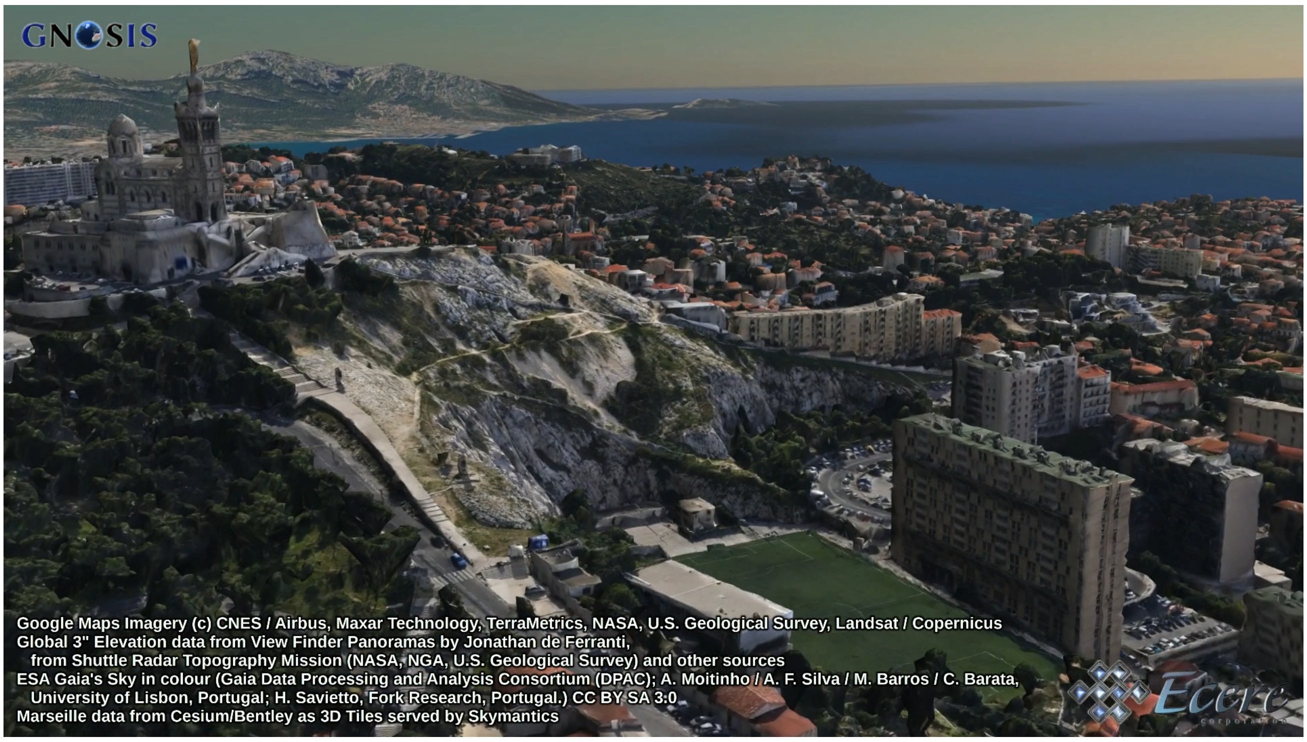
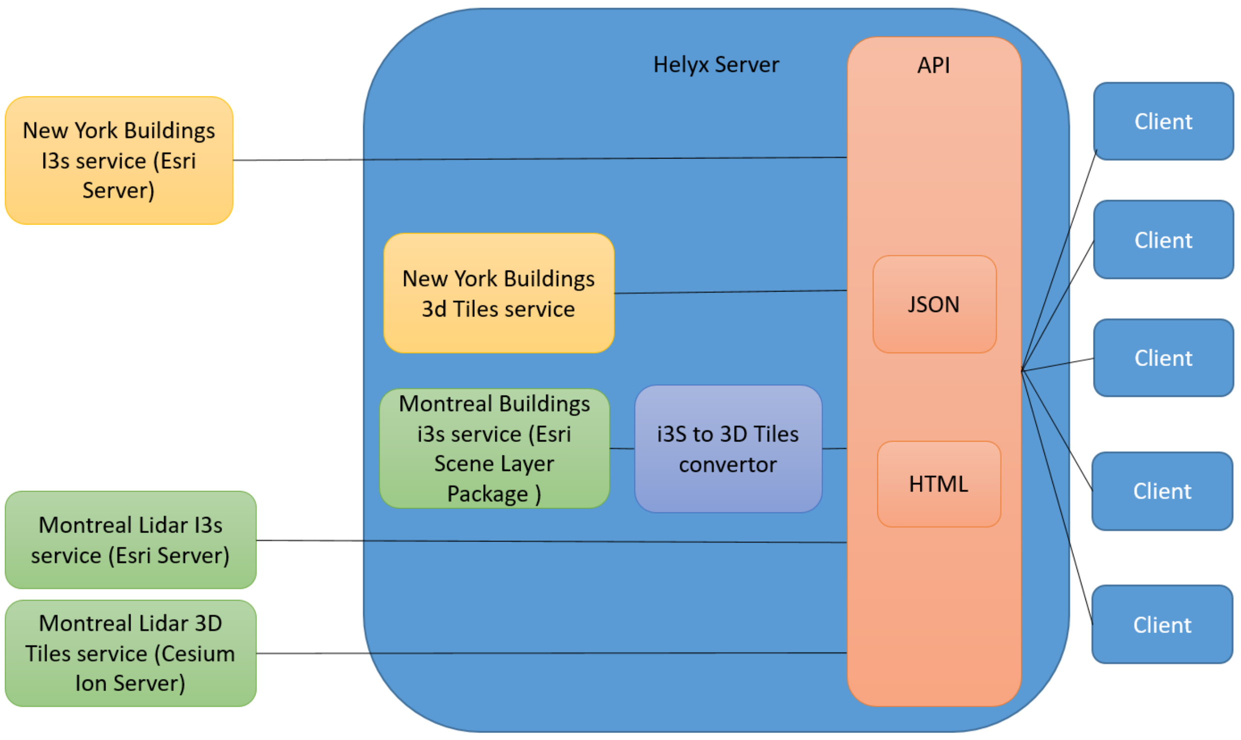

GNOSIS
Google Maps Imagery (c) CNES / Airbus, Maxar Technology, TerraMetrics, NASA, U.S. Geological Survey, Landsat / Copernicus
Global 3" Elevation data from View Finder Panoramas by Jonathan de Ferranti,
from Shuttle Radar Topography Mission (NASA, NGA, U.S. Geological Survey) and other sources
ESA Gaia's Sky in colour (Gaia Data Processing and Analysis Consortium (DPAC); A. Moitinho / A. F. Silva / M. Barros / C. Barata,
University of Lisbon, Portugal; H. Savietto, Fork Research, Portugal.) CC BY SA 3.0
Marseille data from Cesium/Bentley as 3D Tiles served by Skymantics
Ecere

Figure 52. Dataset of Marseille provided by Bentley & Cesium being served as 3D Tiles from the Skymantics service (Notre-Dame de la Garde)

9.5. Helyx

9.5.1. Introduction

For this Pilot Helyx provided a static 3D server 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. An overview of the architecture is shown below:



New York Buildings I3s service (Esri Server)
Helyx Server
API
Client
Client
Client
Client
Client
New York Buildings 3d Tiles service
JSON
Montreal Buildings i3s service (Esri Scene Layer Package)
i3S to 3D Tiles convertor
HTML
Montreal Lidar I3s service (Esri Server)
Montreal Lidar 3D Tiles service (Cesium Ion Server)

Figure 53. Helyx Server Architecture

9.5.2. Server Functionality

9.5.2.1. Data

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 are 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, described below.

9.5.2.2. API Landing Page

The API Landing Page is shown below. The landing page provides details of the API service description, a link to any conformance classes, and a link to the Collections page.

http://helyxapache2.eastus.azurecontainer.io/

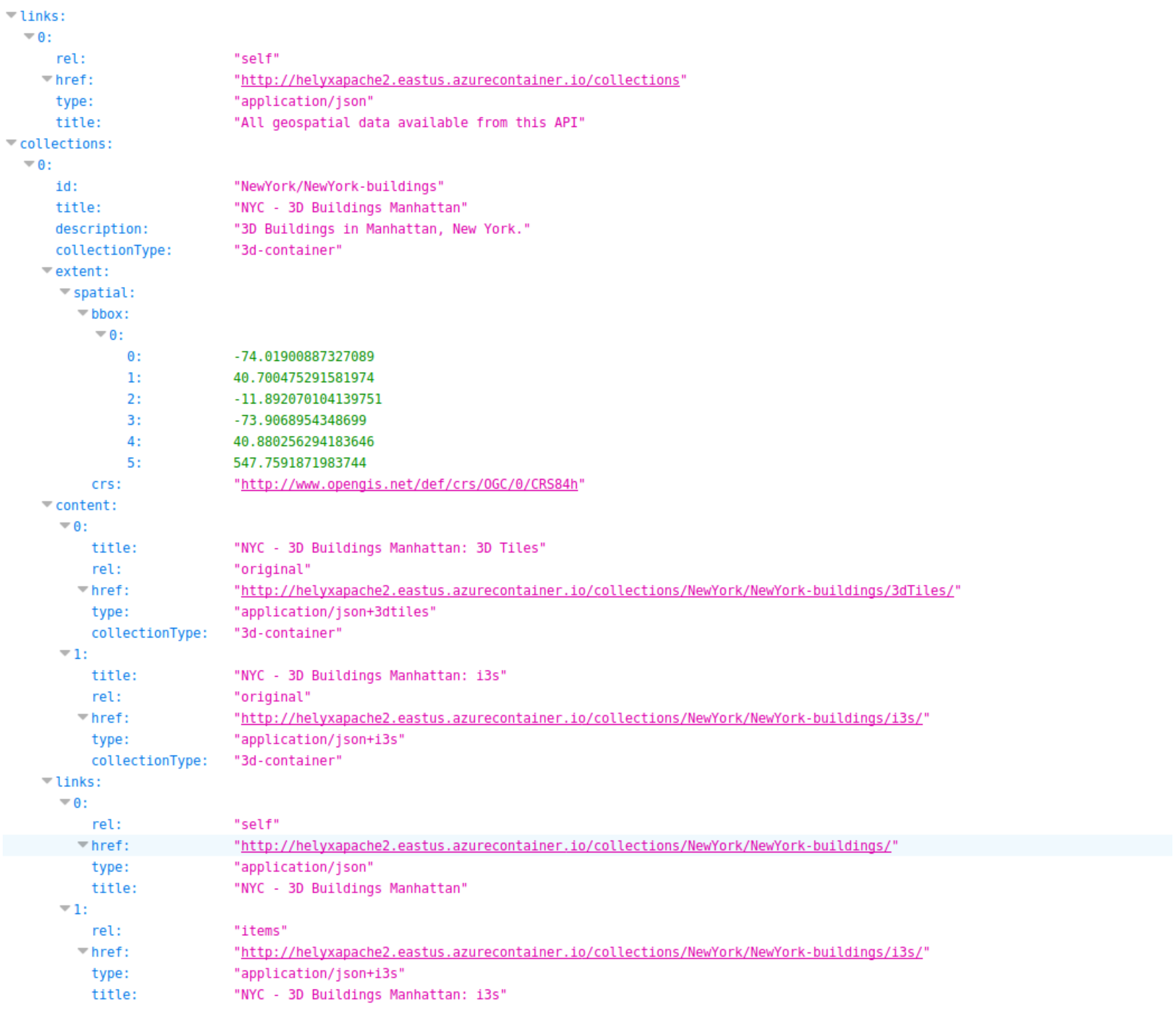
```
{
    "title": "Pilot 3D Container API",
    "description": "A pilot of an API for 3D containers and tiles.",
    "links": [
        {
            "type": "application/openapi+json;version=3.0",
            "title": "Service Description",
            "href": "https://app.swaggerhub.com/apis/timothy-miller/3d-
container/1.0.0",
            "rel": "service-desc"
        },
        {
            "type": "application/json",
            "title": "Conformance",
            "href":
"http://helyxapache2.eastus.azurecontainer.io/conformance/",
            "rel": "conformance"
        },
        {
            "type": "application/json",
            "title": "Collections",
            "href":
"http://helyxapache2.eastus.azurecontainer.io/collections/",
            "rel": "data"
        }
    ]
}
```

9.5.2.3. Collections

The overarching structure of the Collections landing page is shown below:

```
collections/
 ├──── NewYork/
 │     ├──── NewYork-buildings
|     |     ├──── 3DTiles
|     |     ├──── I3S
 ├──── Montreal/
 │     ├──── Montreal-buildings
|     |     ├──── 3DTiles
|     |     ├──── I3S
 │     ├──── Montreal-lidar
|     |     ├──── 3DTiles
|     |     ├──── I3S
```

A subset of the JSON landing page is shown below:



```
▼links:
  ▼0:
      rel:                "self"
    ▼href:                "http://helyxapache2.eastus.azurecontainer.io/collections"
      type:               "application/json"
      title:              "All geospatial data available from this API"
▼collections:
  ▼0:
      id:                 "NewYork/NewYork-buildings"
      title:              "NYC - 3D Buildings Manhattan"
      description:        "3D Buildings in Manhattan, New York."
      collectionType:     "3d-container"
    ▼extent:
      ▼spatial:
        ▼bbox:
          ▼0:
              0:          -74.01900887327089
              1:          40.700475291581974
              2:          -11.892070104139751
              3:          -73.9068954348699
              4:          40.880256294183646
              5:          547.7591871983744
          crs:            "http://www.opengis.net/def/crs/OGC/0/CRS84h"
    ▼content:
      ▼0:
          title:          "NYC - 3D Buildings Manhattan: 3D Tiles"
          rel:            "original"
        ▼href:            "http://helyxapache2.eastus.azurecontainer.io/collections/NewYork/NewYork-buildings/3dTiles/"
          type:           "application/json+3dtiles"
          collectionType: "3d-container"
      ▼1:
          title:          "NYC - 3D Buildings Manhattan: i3s"
          rel:            "original"
        ▼href:            "http://helyxapache2.eastus.azurecontainer.io/collections/NewYork/NewYork-buildings/i3s/"
          type:           "application/json+i3s"
          collectionType: "3d-container"
    ▼links:
      ▼0:
          rel:            "self"
        ▼href:            "http://helyxapache2.eastus.azurecontainer.io/collections/NewYork/NewYork-buildings/"
          type:           "application/json"
          title:          "NYC - 3D Buildings Manhattan"
      ▼1:
          rel:            "items"
        ▼href:            "http://helyxapache2.eastus.azurecontainer.io/collections/NewYork/NewYork-buildings/i3s/"
          type:           "application/json+i3s"
          title:          "NYC - 3D Buildings Manhattan: i3s"
```

9.5.2.4. Bounding Box query

A simple Bounding Box query was implemented on the server - as it was not a feature of the server before, only a simple 'Completely Contains' bounding box query was applied in the TIE time. However, this issue is not a reflection on the difficulty to implement the API, but is only a server configuration issue.

9.5.2.5. Features and Benefits

HTML and JSON representation: For ease of navigation, the implementation created an HTML version of the API as well as a JSON version. Which was returned as a function of a query parameter, actioned by f=html, or f=json.

3D Container Conversion service: The server utilized the I3S to 3D Tiles converter service to provide a further option for clients using the API. This allows chaining and conversion of services which is a powerful feature.

9.5.2.6. Challenges

A specific challenge was encountered between server and client when a CORS redirect occurred. The 301 redirect HTTP response code meant that CORS headers were not passed on to the client. An example was the case of trailing slashes. If a client visited a page without a trailing slash, typically this would redirect to the page on the server with the trailing slash. This was by default achieved using a redirect, but then the 301 CORS issue occurred. This was remedied by using a rewrite instead of redirect.

9.6. Skymantics

9.6.1. General architecture:

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

9.6.1.1. 3D data

The 3D data offered by Skymantics 3D Tiles API is structured as hierarchical 3D Containers. A 3D container can contain either the 3D data (either 3DTiles or I3S representations) or other 3D containers. As depicted below, for the Pilot there are two 3D containers at the top level: Westworld and Malalison. Westworld contains two 3D containers: NYC (3D buildings in New York, USA, both in 3D Tiles and I3S representations) and Marseille (the city of Marseille, France, in 3D Tiles representation). Malalison contains a point cloud of the Malalison Island in the Philippines (in 3D Tiles representation).

```
collections/
 ├──── Westworld/
 │     ├──── NYC
|     |     ├──── 3DTiles
|     |     ├──── I3S
 │     ├──── Marseille
|     |     ├──── 3DTiles
 ├──── Malalison/
 │     ├──── 3DTiles
```

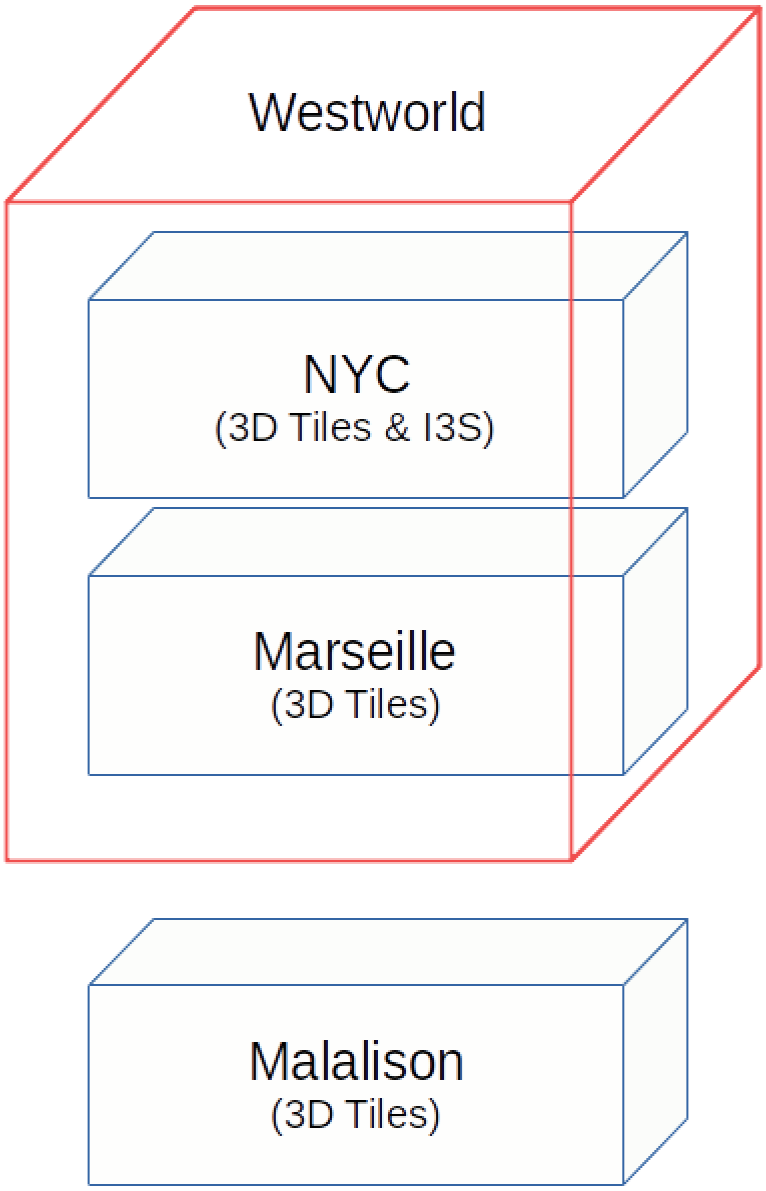


Figure 54. 3D Container Structure

9.6.1.2. Service architecture

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.

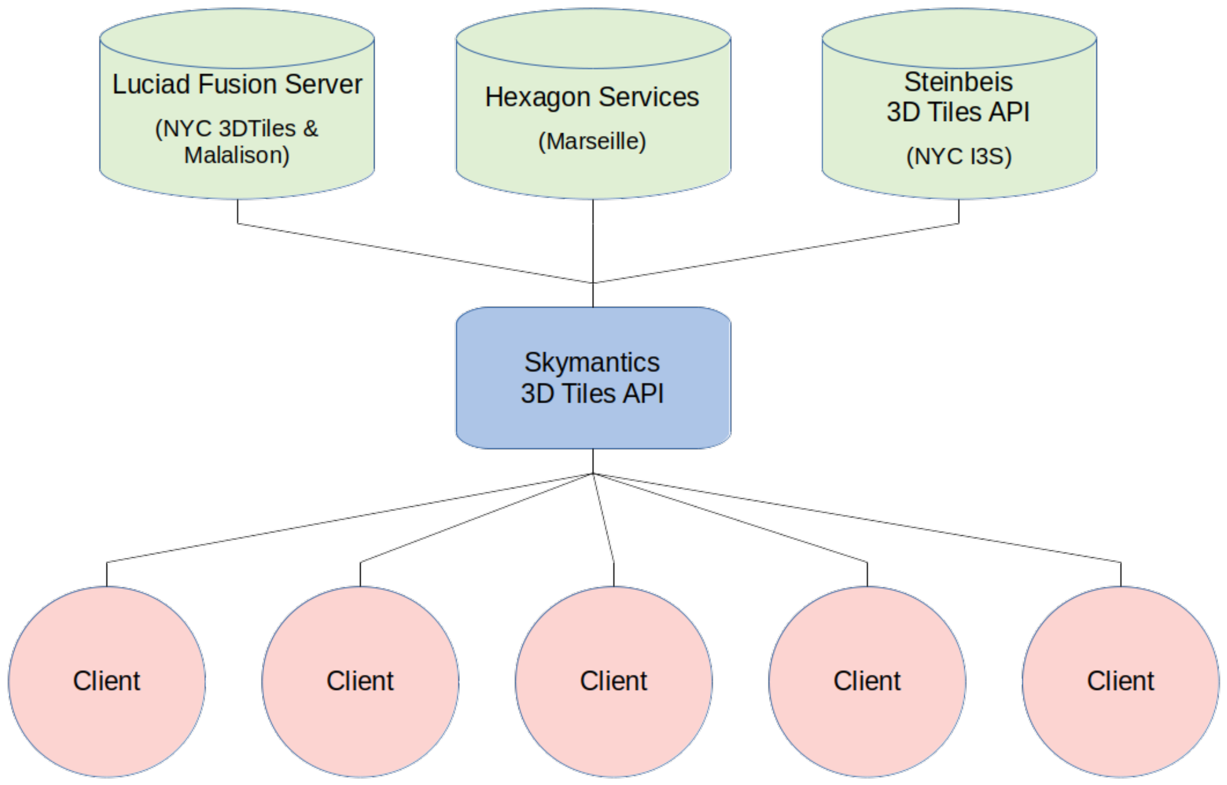


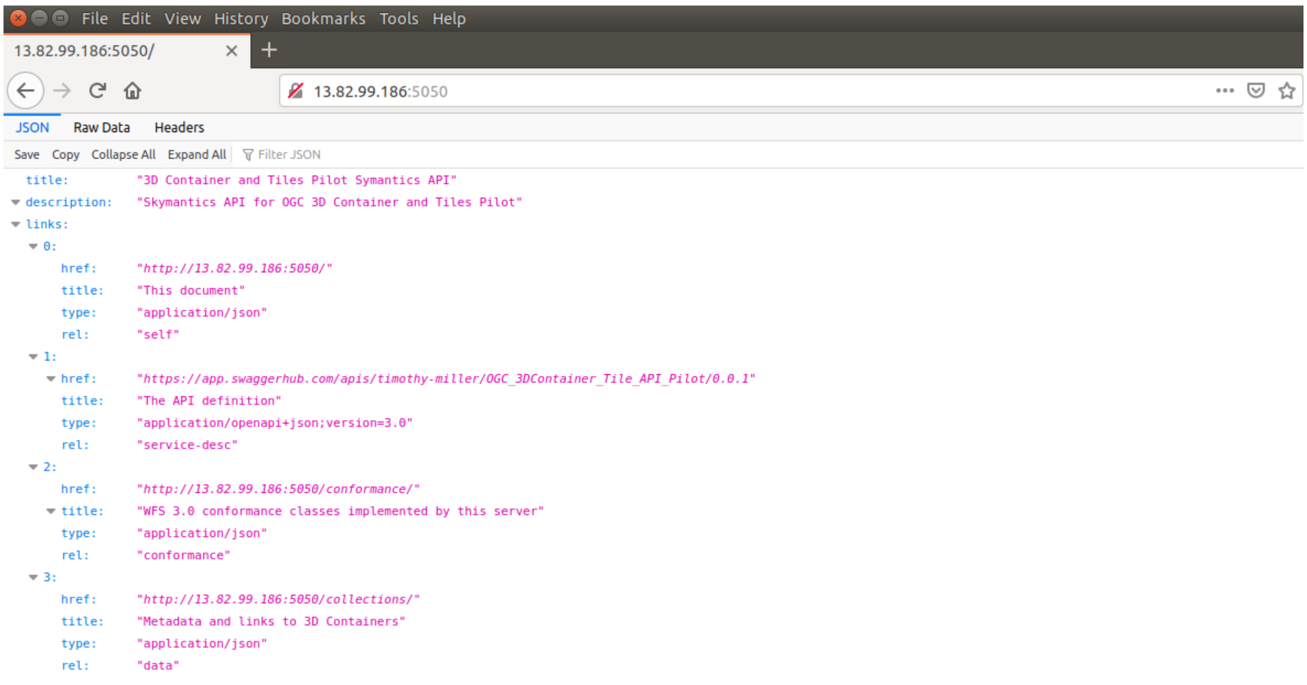
Figure 55. Service Architecture

For the client implementation, Skymantics chose Hexagon's Luciad RIA. At the start of the Pilot, the available version of RIA was 2019.1.03, which was deployed on Azure and accessed through a web client. An additional RIA instance was deployed locally to test new versions and extra features. At the end of the pilot, the RIA version on the test environment was 2020.0.09, which included several fixes and the access to Hexagon's satellite imagery.

9.6.2. Server Functionality:

9.6.2.1. Landing Page

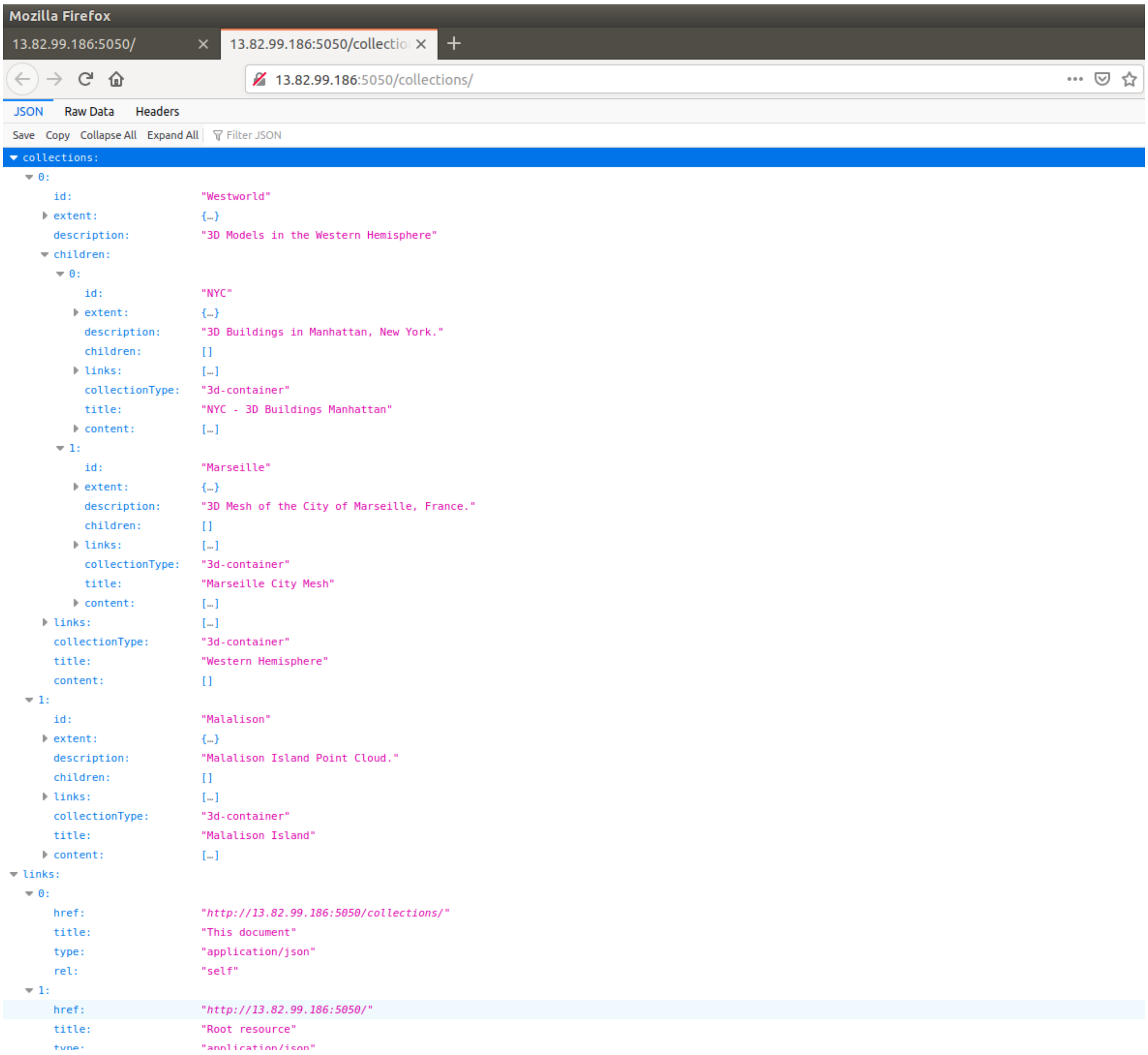
The landing page of the Skymantics API is available at http://13.82.99.186:5050. The landing page follows the structure specified in OGC API - Common, with general information on the API as well as links to the API definition, conformance classes and the 3D containers.


File Edit View History Bookmarks Tools Help
13.82.99.186:5050/
13.82.99.186:5050
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON

```
title:          "3D Container and Tiles Pilot Symantics API"
description:    "Skymantics API for OGC 3D Container and Tiles Pilot"
links:
  0:
    href:       "http://13.82.99.186:5050/"
    title:      "This document"
    type:       "application/json"
    rel:        "self"
  1:
    href:       "https://app.swaggerhub.com/apis/timothy-miller/OGC_3DContainer_Tile_API_Pilot/0.0.1"
    title:      "The API definition"
    type:       "application/openapi+json;version=3.0"
    rel:        "service-desc"
  2:
    href:       "http://13.82.99.186:5050/conformance/"
    title:      "WFS 3.0 conformance classes implemented by this server"
    type:       "application/json"
    rel:        "conformance"
  3:
    href:       "http://13.82.99.186:5050/collections/"
    title:      "Metadata and links to 3D Containers"
    type:       "application/json"
    rel:        "data"
```

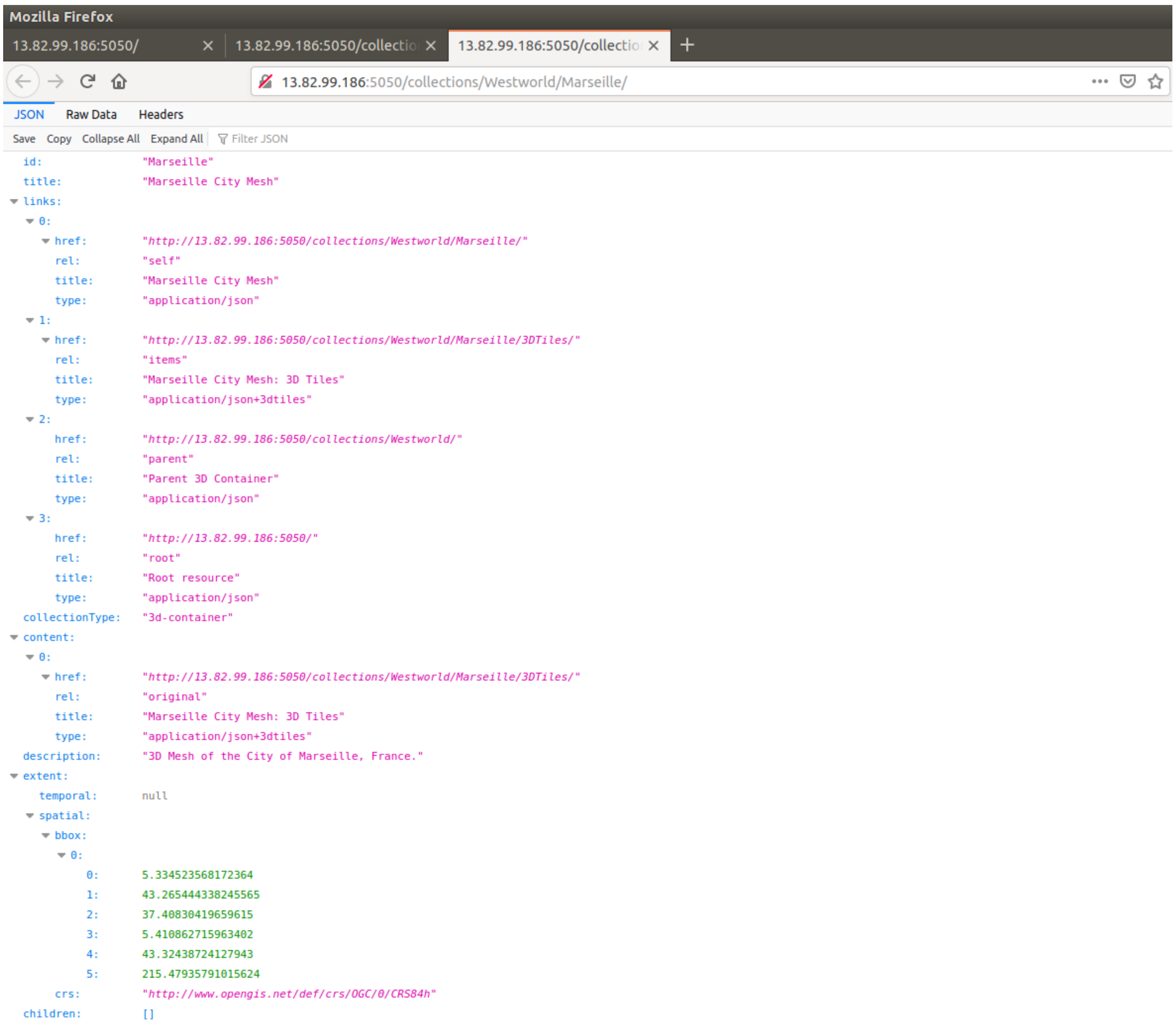
9.6.2.2. 3D Container

Clicking on the data link from the API landing page (http://13.82.99.186:5050/collections/), the API returns detailed information on all the 3D containers offered. This information is sufficient to understand the 3D data offered by the API and to select the appropriate collections. The screenshot below depicts the hierarchy structure of the Skymantics implementation, with Westworld containing both NYC and Marseille.



```
▼ collections:
  ▼ 0:
      id:                 "Westworld"
    ▶ extent:             {…}
      description:        "3D Models in the Western Hemisphere"
    ▼ children:
      ▼ 0:
          id:             "NYC"
        ▶ extent:         {…}
          description:    "3D Buildings in Manhattan, New York."
          children:       []
        ▶ links:          […]
          collectionType: "3d-container"
          title:          "NYC - 3D Buildings Manhattan"
        ▶ content:        […]
      ▼ 1:
          id:             "Marseille"
        ▶ extent:         {…}
          description:    "3D Mesh of the City of Marseille, France."
          children:       []
        ▶ links:          […]
          collectionType: "3d-container"
          title:          "Marseille City Mesh"
        ▶ content:        […]
    ▶ links:              […]
      collectionType:     "3d-container"
      title:              "Western Hemisphere"
      content:            []
  ▼ 1:
      id:                 "Malalison"
    ▶ extent:             {…}
      description:        "Malalison Island Point Cloud."
      children:           []
    ▶ links:              […]
      collectionType:     "3d-container"
      title:              "Malalison Island"
    ▶ content:            […]
▼ links:
  ▼ 0:
      href:               "http://13.82.99.186:5050/collections/"
      title:              "This document"
      type:               "application/json"
      rel:                "self"
  ▼ 1:
      href:               "http://13.82.99.186:5050/"
      title:              "Root resource"
```

It is possible to follow each 3D Container link to get access just to its data. The screenshot below depicts the detailed information of the Marseille 3D container (http://13.82.99.186:5050/collections/Westworld/Marseille/).



Mozilla Firefox

13.82.99.186:5050/ | 13.82.99.186:5050/collectio | 13.82.99.186:5050/collectio

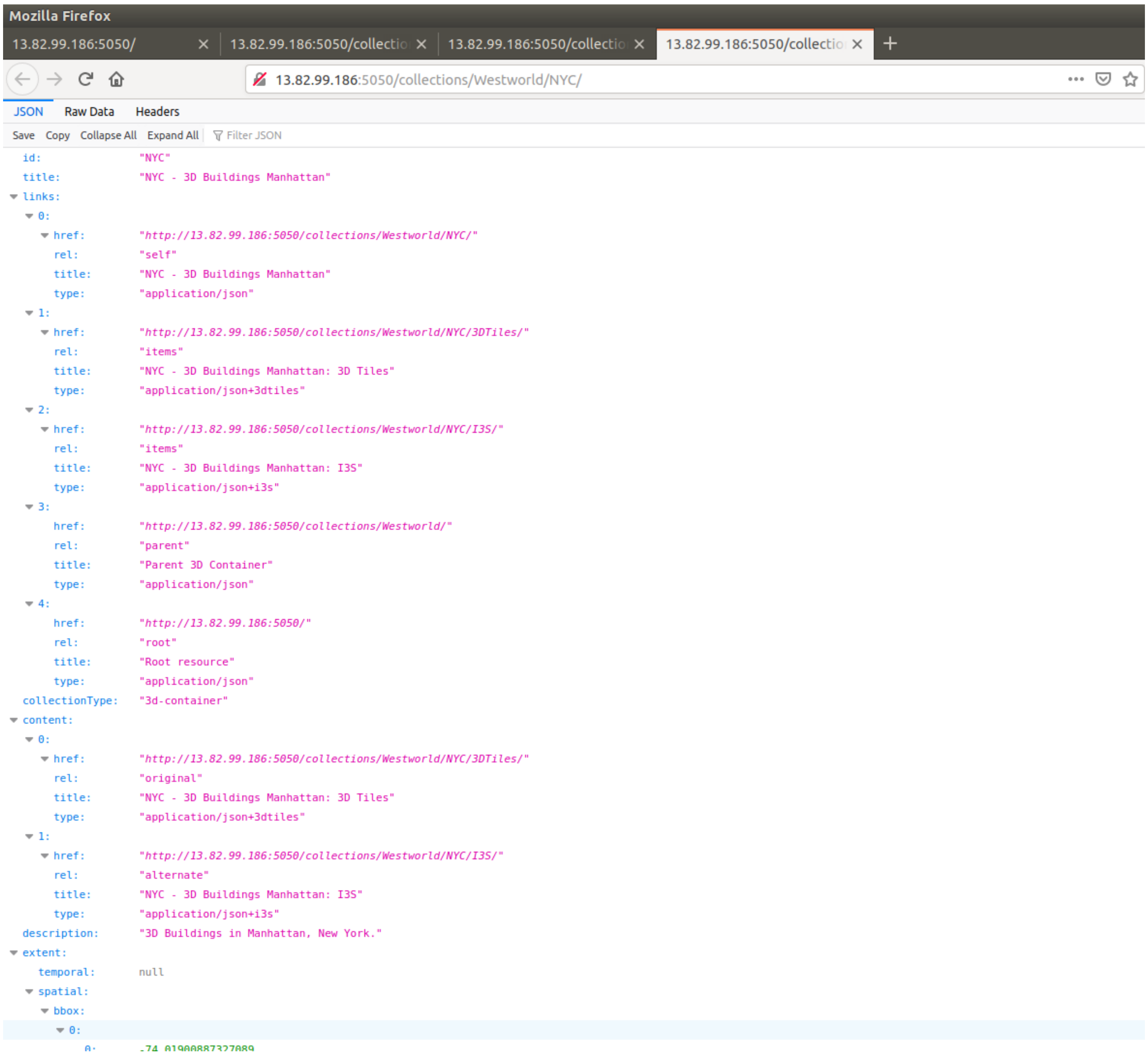
13.82.99.186:5050/collections/Westworld/Marseille/

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```
id:                 "Marseille"
title:              "Marseille City Mesh"
▼ links:
  ▼ 0:
    ▼ href:         "http://13.82.99.186:5050/collections/Westworld/Marseille/"
      rel:          "self"
      title:        "Marseille City Mesh"
      type:         "application/json"
  ▼ 1:
    ▼ href:         "http://13.82.99.186:5050/collections/Westworld/Marseille/3DTiles/"
      rel:          "items"
      title:        "Marseille City Mesh: 3D Tiles"
      type:         "application/json+3dtiles"
  ▼ 2:
      href:         "http://13.82.99.186:5050/collections/Westworld/"
      rel:          "parent"
      title:        "Parent 3D Container"
      type:         "application/json"
  ▼ 3:
      href:         "http://13.82.99.186:5050/"
      rel:          "root"
      title:        "Root resource"
      type:         "application/json"
collectionType:     "3d-container"
▼ content:
  ▼ 0:
    ▼ href:         "http://13.82.99.186:5050/collections/Westworld/Marseille/3DTiles/"
      rel:          "original"
      title:        "Marseille City Mesh: 3D Tiles"
      type:         "application/json+3dtiles"
description:        "3D Mesh of the City of Marseille, France."
▼ extent:
    temporal:       null
  ▼ spatial:
    ▼ bbox:
      ▼ 0:
          0:        5.334523568172364
          1:        43.265444338245565
          2:        37.40830419659615
          3:        5.410862715963402
          4:        43.32438724127943
          5:        215.47935791015624
      crs:          "http://www.opengis.net/def/crs/OGC/0/CRS84h"
children:           []
```

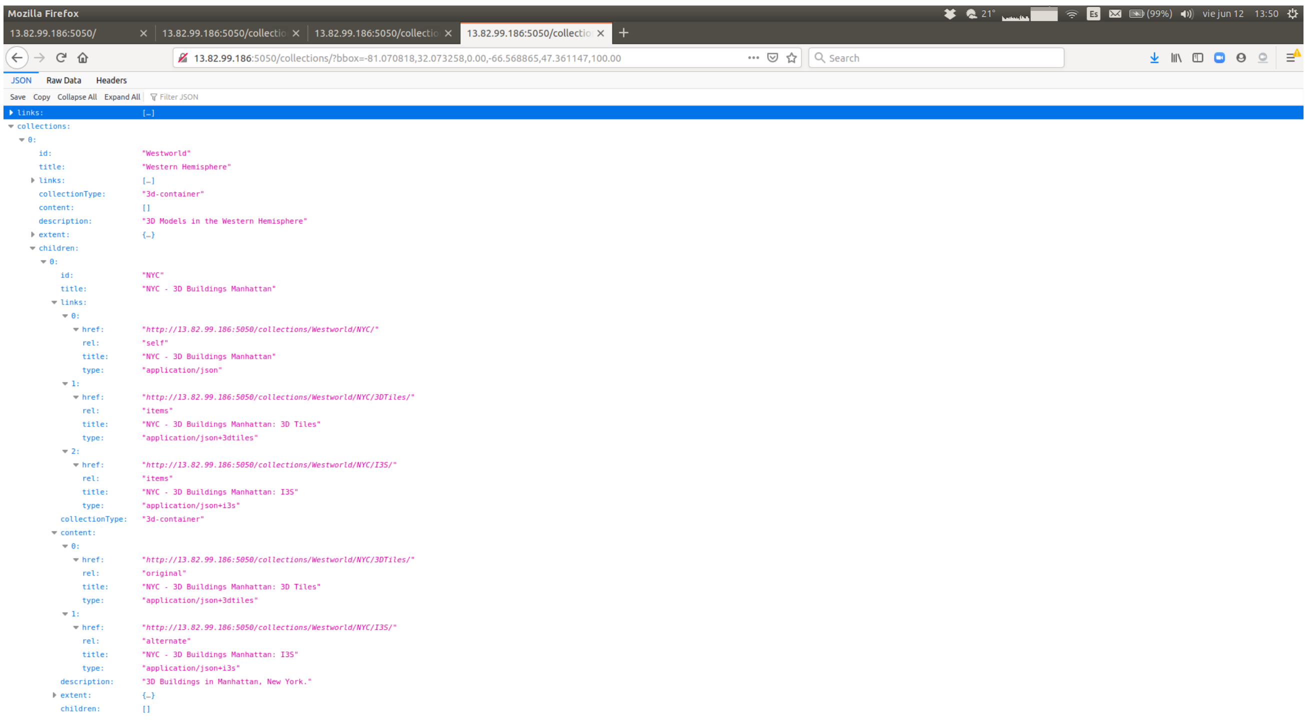
The values of parameter 'rel' in content links indicate which of the representations is recommended as the primary source of data (where 'rel' is 'original'). The screenshot below shows an example in the NYC 3D container (http://13.82.99.186:5050/collections/Westworld/NYC/), that offers 3D Tiles and I3S representations but tags the first one as the original and the second as the alternate.


Mozilla Firefox
13.82.99.186:5050/
13.82.99.186:5050/collectio
13.82.99.186:5050/collectio
13.82.99.186:5050/collectio
13.82.99.186:5050/collections/Westworld/NYC/
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON

```
  id:              "NYC"
  title:           "NYC - 3D Buildings Manhattan"
▼ links:
  ▼ 0:
    ▼ href:        "http://13.82.99.186:5050/collections/Westworld/NYC/"
      rel:         "self"
      title:       "NYC - 3D Buildings Manhattan"
      type:        "application/json"
  ▼ 1:
    ▼ href:        "http://13.82.99.186:5050/collections/Westworld/NYC/3DTiles/"
      rel:         "items"
      title:       "NYC - 3D Buildings Manhattan: 3D Tiles"
      type:        "application/json+3dtiles"
  ▼ 2:
    ▼ href:        "http://13.82.99.186:5050/collections/Westworld/NYC/I3S/"
      rel:         "items"
      title:       "NYC - 3D Buildings Manhattan: I3S"
      type:        "application/json+i3s"
  ▼ 3:
      href:        "http://13.82.99.186:5050/collections/Westworld/"
      rel:         "parent"
      title:       "Parent 3D Container"
      type:        "application/json"
  ▼ 4:
      href:        "http://13.82.99.186:5050/"
      rel:         "root"
      title:       "Root resource"
      type:        "application/json"
  collectionType:  "3d-container"
▼ content:
  ▼ 0:
    ▼ href:        "http://13.82.99.186:5050/collections/Westworld/NYC/3DTiles/"
      rel:         "original"
      title:       "NYC - 3D Buildings Manhattan: 3D Tiles"
      type:        "application/json+3dtiles"
  ▼ 1:
    ▼ href:        "http://13.82.99.186:5050/collections/Westworld/NYC/I3S/"
      rel:         "alternate"
      title:       "NYC - 3D Buildings Manhattan: I3S"
      type:        "application/json+i3s"
  description:     "3D Buildings in Manhattan, New York."
▼ extent:
    temporal:      null
  ▼ spatial:
    ▼ bbox:
      ▼ 0:
          0:       -74.01900887327089
```

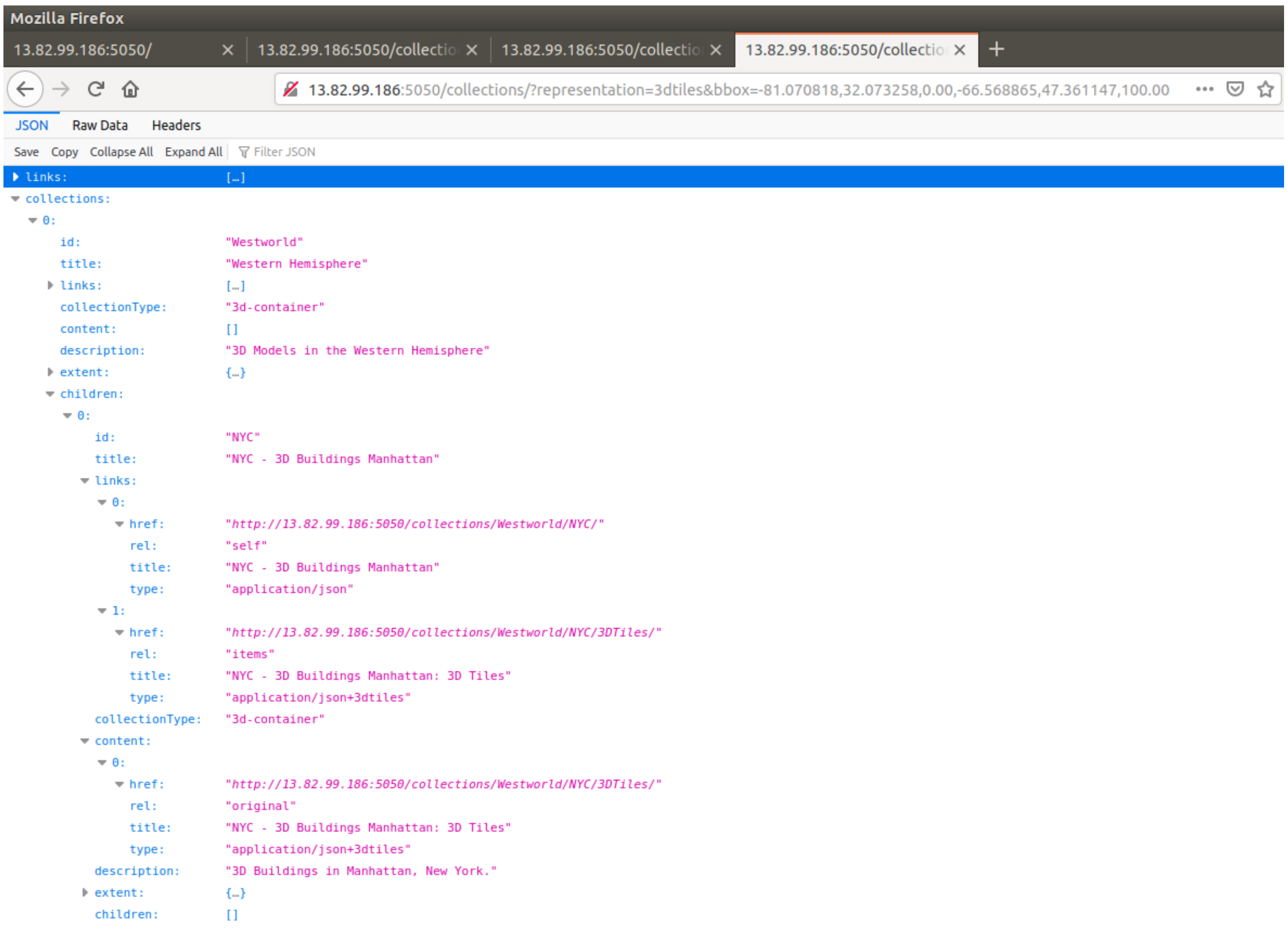
9.6.2.3. Bounding Box query

The API offers the possibility to query 3D Containers within the limits of a bounding box. The bounding box can be either 2D or 3D. The screenshot below shows the result of a query with a bounding box for the East coast in the US (http://13.82.99.186:5050/collections/?bbox=-81.070818,32.073258,0.00,-66.568865,47.361147,100.00), that results only in the NYC 3D container returned, together with its parent Westworld 3D container. Marseille and Malalison are filtered out.


```
13.82.99.186:5050/collections/?bbox=-81.070818,32.073258,0.00,-66.568865,47.361147,100.00

▶ links:                  […]
▼ collections:
  ▼ 0:
      id:                 "Westworld"
      title:              "Western Hemisphere"
    ▶ links:              […]
      collectionType:     "3d-container"
      content:            []
      description:        "3D Models in the Western Hemisphere"
    ▶ extent:             {…}
    ▼ children:
      ▼ 0:
          id:             "NYC"
          title:          "NYC - 3D Buildings Manhattan"
        ▼ links:
          ▼ 0:
            ▼ href:       "http://13.82.99.186:5050/collections/Westworld/NYC/"
              rel:        "self"
              title:      "NYC - 3D Buildings Manhattan"
              type:       "application/json"
          ▼ 1:
            ▼ href:       "http://13.82.99.186:5050/collections/Westworld/NYC/3DTiles/"
              rel:        "items"
              title:      "NYC - 3D Buildings Manhattan: 3D Tiles"
              type:       "application/json+3dtiles"
          ▼ 2:
            ▼ href:       "http://13.82.99.186:5050/collections/Westworld/NYC/I3S/"
              rel:        "items"
              title:      "NYC - 3D Buildings Manhattan: I3S"
              type:       "application/json+i3s"
          collectionType: "3d-container"
        ▼ content:
          ▼ 0:
            ▼ href:       "http://13.82.99.186:5050/collections/Westworld/NYC/3DTiles/"
              rel:        "original"
              title:      "NYC - 3D Buildings Manhattan: 3D Tiles"
              type:       "application/json+3dtiles"
          ▼ 1:
            ▼ href:       "http://13.82.99.186:5050/collections/Westworld/NYC/I3S/"
              rel:        "alternate"
              title:      "NYC - 3D Buildings Manhattan: I3S"
              type:       "application/json+i3s"
          description:    "3D Buildings in Manhattan, New York."
        ▶ extent:         {…}
          children:       []
```

Additionally, collections can be filtered by representation, which is useful when clients do not support all representations. The query parameter "representation" accepts values "3dtiles" or "i3s". The screenshot below shows the result of the previous query but specifying the 3DTiles representation (http://13.82.99.186:5050/collections/?representation=3dtiles&bbox=-81.070818,32.073258,0.00,-66.568865,47.361147,100.00). This request results in the NYC 3D container with just the link to 3D Tiles returned.



```
13.82.99.186:5050/collections/?representation=3dtiles&bbox=-81.070818,32.073258,0.00,-66.568865,47.361147,100.00
```

```
▸ links:                  […]
▾ collections:
  ▾ 0:
      id:                 "Westworld"
      title:              "Western Hemisphere"
    ▸ links:              […]
      collectionType:     "3d-container"
      content:            []
      description:        "3D Models in the Western Hemisphere"
    ▸ extent:             {…}
    ▾ children:
      ▾ 0:
          id:             "NYC"
          title:          "NYC - 3D Buildings Manhattan"
        ▾ links:
          ▾ 0:
            ▾ href:       "http://13.82.99.186:5050/collections/Westworld/NYC/"
              rel:        "self"
              title:      "NYC - 3D Buildings Manhattan"
              type:       "application/json"
          ▾ 1:
            ▾ href:       "http://13.82.99.186:5050/collections/Westworld/NYC/3DTiles/"
              rel:        "items"
              title:      "NYC - 3D Buildings Manhattan: 3D Tiles"
              type:       "application/json+3dtiles"
          collectionType: "3d-container"
        ▾ content:
          ▾ 0:
            ▾ href:       "http://13.82.99.186:5050/collections/Westworld/NYC/3DTiles/"
              rel:        "original"
              title:      "NYC - 3D Buildings Manhattan: 3D Tiles"
              type:       "application/json+3dtiles"
          description:    "3D Buildings in Manhattan, New York."
        ▸ extent:         {…}
          children:       []
```

9.6.3. Client Functionality:

9.6.3.1. Request and parse Landing Page

Skymantics developed a visual, mouse-governed client to test the pilot’s 3D Tiles API implementations. Skymantics client is available at http://13.82.99.186:8072/web/3dcat/index.html

The screenshot below shows the client screen right after loading the client with Firefox browser. The bottom-left corner provides direct access to all the pilot participants' APIs. Clicking each button triggers the process to query the API landing page and explore through the 3D containers finding the available data. The top-right corner shows the map layers of the app, which apart from the World map, consists of all the visible 3D containers found in the API. The bottom-right controller allows to filter by bounding box.

The default API that the client queries is Skymantics 3D Tiles API. As such, the top-right corner lists all the 3D containers offered by the API, as described before: Malalison, Westworld/Marseille and Westworld/NYC.

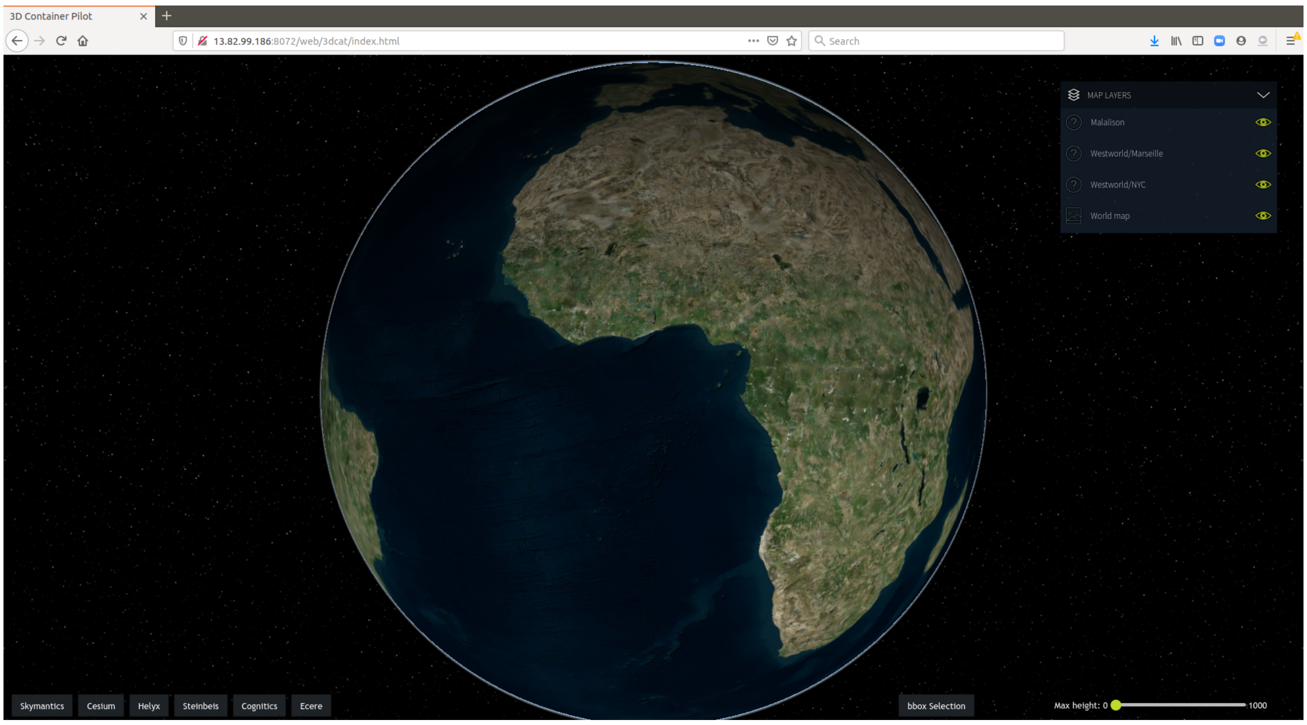
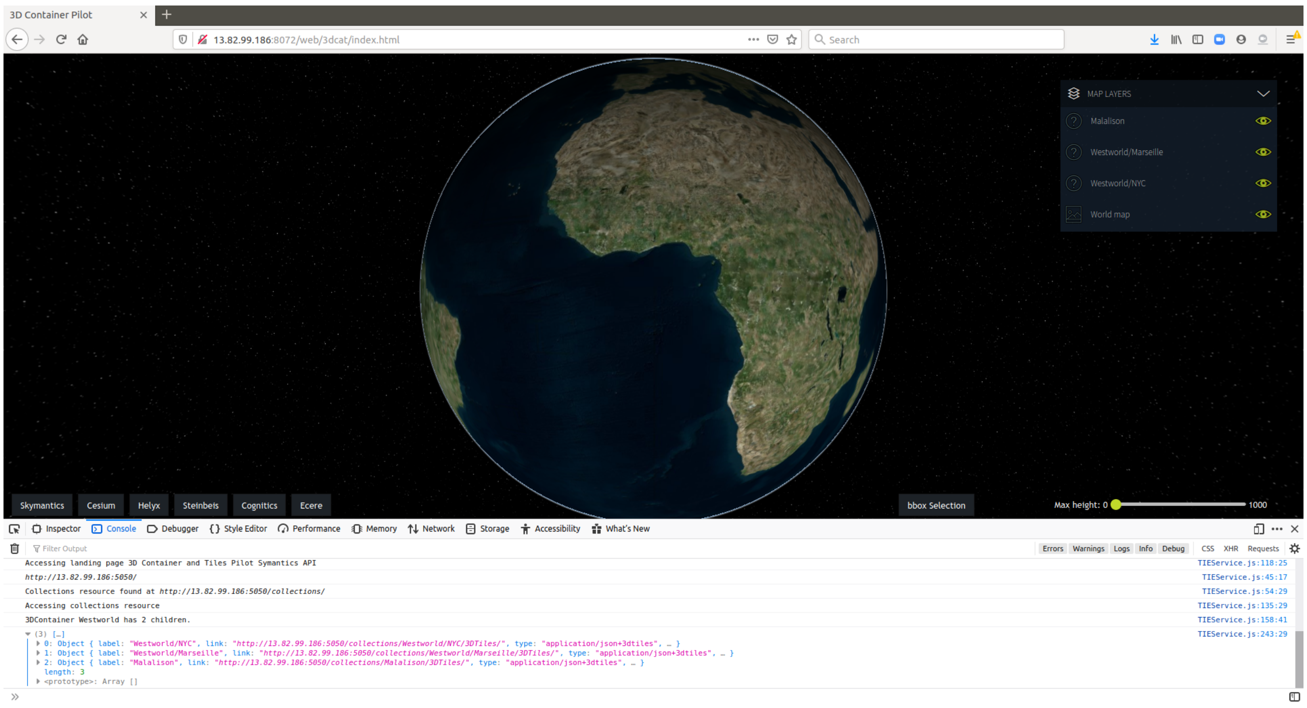


Figure 56. Request and Parse Landing Page

9.6.3.2. Request and parse 3D container

By opening the web console of the Firefox browser (other browsers offer similar tools) it is possible to review the details of how the call is processed. The screenshot below shows the logs in the console after clicking on the "Skymantics" button. These are the steps described:

1. The client accesses the Skymantics 3DTiles API landing page at http://13.82.99.186:5050/.
2. The client finds the collections resource at http://13.82.99.186:5050/collections/.
3. The client accesses the collections resource.
4. The client finds a 3D container (Westworld) with two children and browses the content.
5. The client logs the information of all the three visible 3D containers found, including label, link to the data, type of representation and bounding box.
6. The client shows at the top-right the list of the three visible 3D containers.



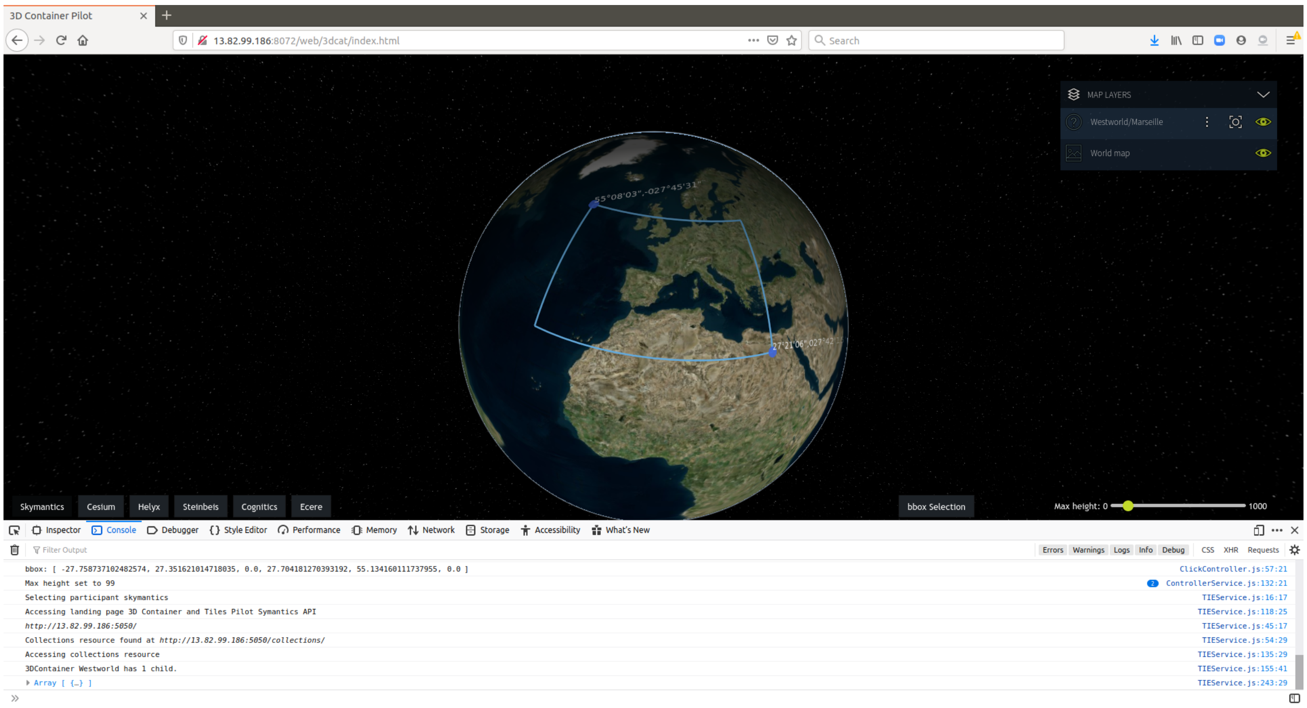
```
Accessing landing page 3D Container and Tiles Pilot Symantics API          TIEService.js:118:25
http://13.82.99.186:5050/                                                  TIEService.js:45:17
Collections resource found at http://13.82.99.186:5050/collections/        TIEService.js:54:29
Accessing collections resource                                             TIEService.js:135:29
3DContainer Westworld has 2 children.                                      TIEService.js:158:41
▼ (3) […]                                                                  TIEService.js:243:29
  ▶ 0: Object { label: "Westworld/NYC", link: "http://13.82.99.186:5050/collections/Westworld/NYC/3DTiles/", type: "application/json+3dtiles", … }
  ▶ 1: Object { label: "Westworld/Marseille", link: "http://13.82.99.186:5050/collections/Westworld/Marseille/3DTiles/", type: "application/json+3dtiles", … }
  ▶ 2: Object { label: "Malalison", link: "http://13.82.99.186:5050/collections/Malalison/3DTiles/", type: "application/json+3dtiles", … }
    length: 3
  ▶ <prototype>: Array []
```

Figure 57. Request and Parse 3D Container

9.6.3.3. Filter by bounding box

3D containers offered by the 3D Tiles API can be filtered by bounding box. The screenshot below shows the results after drawing a bounding box in Europe and clicking the "Skymantics" button. These are the steps taken:

1. The user clicks on "bbox Selection" button. Now a bounding box can be drawn on the globe.
2. The user clicks on two points in the globe. The client calculates the bounding box and draws its limits. At this moment the bbox selected is 2D.
3. The user moves the slider to select a maximum height (minimum height will always be 0). At this moment the bbox selected is 3D.
4. The user clicks on "Skymantics" button. The client accesses Skymantics 3DTiles API landing page.
5. The client finds the collections resource and accesses it adding the bbox filter.
6. The API responds with only one 3D container (Marseille) inside a parent container (Westworld). All other 3D containers have been filtered out.
7. The client shows at the top-right the only visible 3D container.



```
bbox: [ -27.758737102482574, 27.351621014718035, 0.0, 27.704181270393192, 55.134160111737955, 0.0 ]   ClickController.js:57:21
Max height set to 99                                                                          2 ControllerService.js:132:21
Selecting participant skymantics                                                                    TIEService.js:16:17
Accessing landing page 3D Container and Tiles Pilot Symantics API                                   TIEService.js:118:25
http://13.82.99.186:5050/                                                                           TIEService.js:45:17
Collections resource found at http://13.82.99.186:5050/collections/                                 TIEService.js:54:29
Accessing collections resource                                                                      TIEService.js:135:29
3DContainer Westworld has 1 child.                                                                  TIEService.js:155:41
▸ Array [ {…} ]                                                                                     TIEService.js:243:29
```

Figure 58. Filter by Bounding Box

9.6.3.4. Request and parse 3D tiles

Once the 3D Tiles API has been properly queried and parsed, the client knows all the visible 3D containers available for visualization, as well as the link to fetch their data. In order to visualize the returned content, click on the "Fit to data" button of the 3D container at the top-right list and the client will zoom into the bounding box of the 3D data and start rendering it. The screenshot below shows the visualization of the Marseille 3D container.

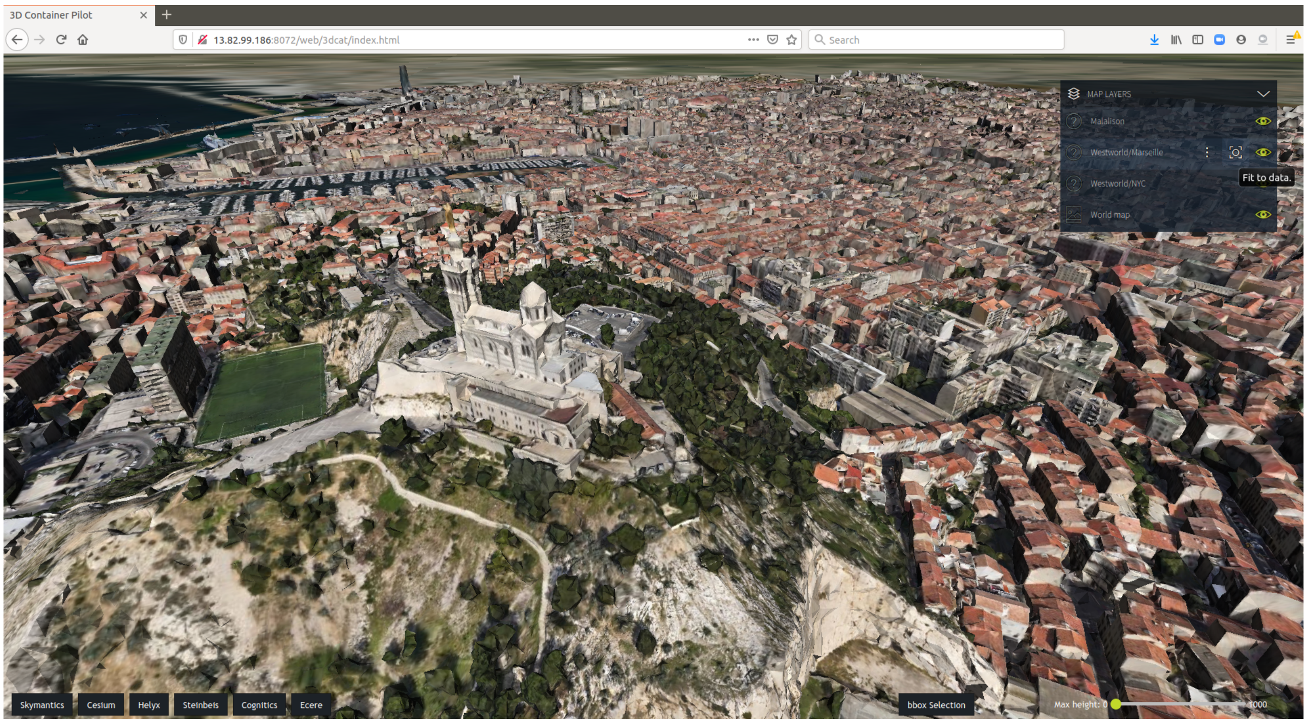


Figure 59. Marseille 3D Container

The screenshot below shows the visualization of the NYC 3D container. Note that this client version is using the latest RIA updates and Hexagon's maps.

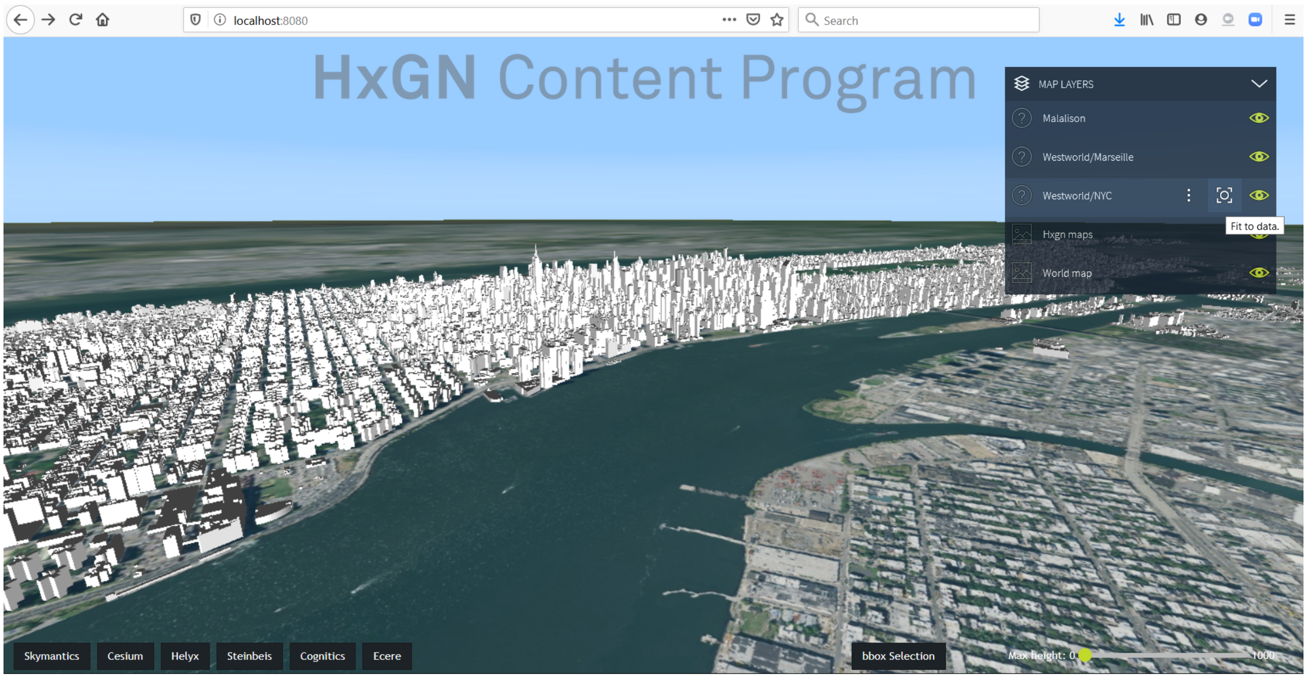


Figure 60. NYC 3D Container

The process to test other participants' APIs is just the same: The user clicks on the participant's button, waits until the client has browsed via the API and populated the list of visible 3D containers, and then selects the one to visualize. The screenshot below shows the visualization of Cesium's NewYorkBuildings 3D containers. Note that Cesium's data structure also follows a hierarchy, with WaratahStation and Buildings 3D containers at the top, and WesternSydneyBuildings, NewYorkBuildings and CesiumOSMBuildings inside Buildings.

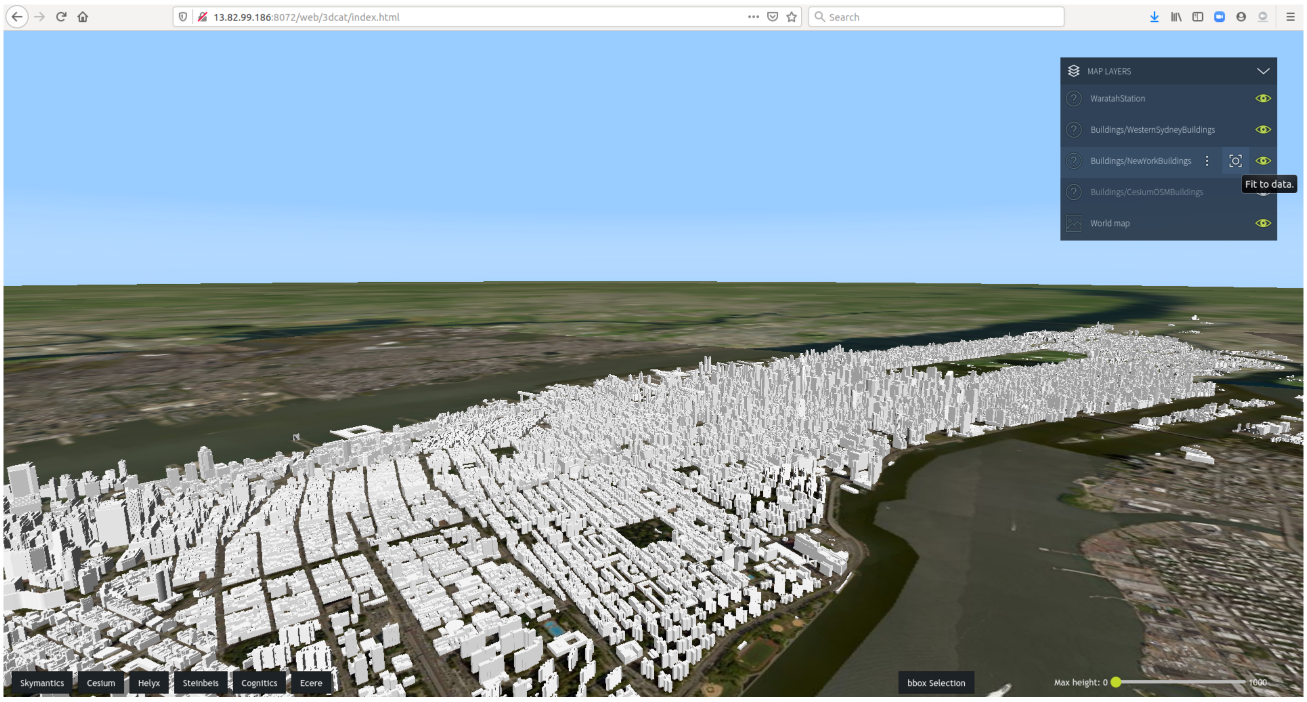

13.82.99.186:8072/web/3dcat/index.html
MAP LAYERS
WaratahStation
Buildings/WesternSydneyBuildings
Buildings/NewYorkBuildings
Fit to data.
Buildings/CesiumOSMBuildings
World map
Skymantics
Cesium
Helyx
Steinbeis
Cognitics
Ecere
bbox Selection
Max height: 0
1000

Figure 61. Cesium's NewYorkBuildings 3D Containers

9.6.4. Lessons learned:

9.6.4.1. 3D rendering issues

The rendering of 3D content was not always smooth across different clients. During this Pilot there were numerous examples of 3D models suffering rendering issues, sometimes showing severe problems. For example, the dataset for Malalison Island showed issues in Luciad RIA 2019, where the point cloud was displayed with mixed heights, as documented in the following image.

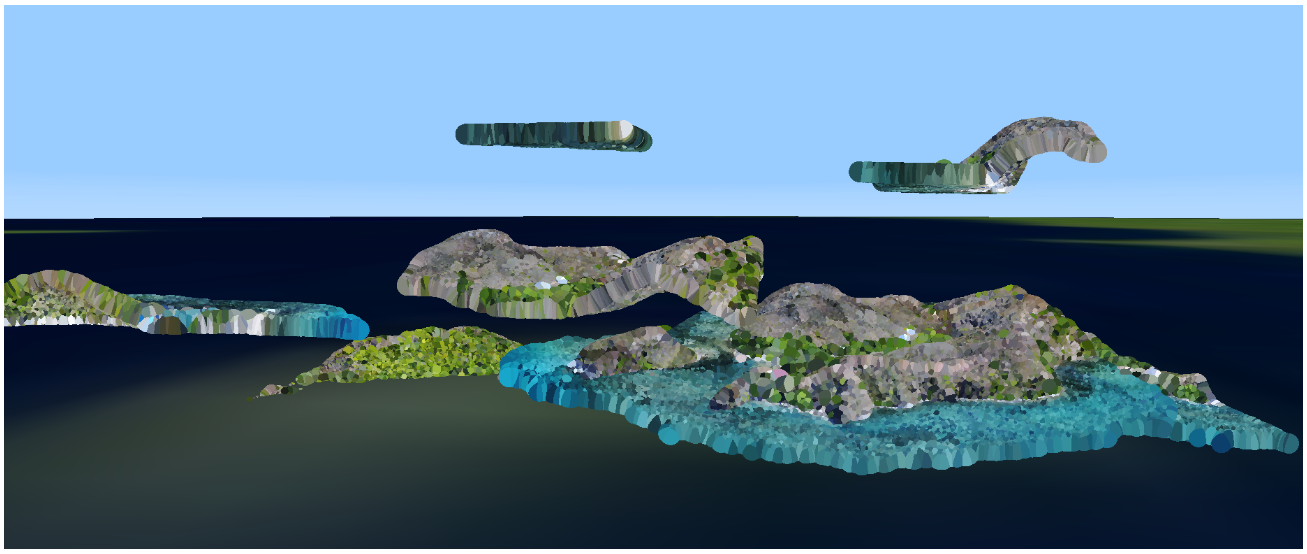


Figure 62. 3D Rendering Issues

The issue was fixed thanks to the feedback provided during the Pilot and Luciad RIA 2020 now renders a "neat" point cloud, as documented in the following image.

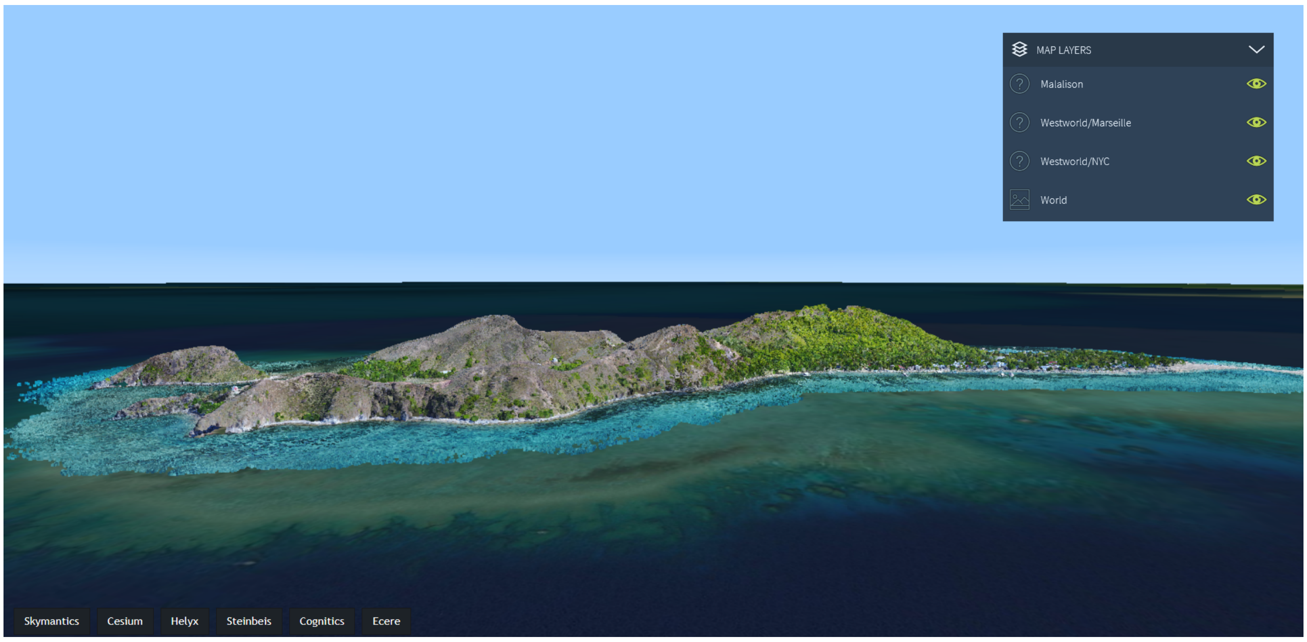


Figure 63. Rendering Issues Resolved

There are still some pending issues, with several of Cesium's datasets displaying triangles instead of straight lines. This issue is currently under investigation by Luciad's development team. The following image shows an example of this issue, with Cesium's Waratah station dataset rendered by Luciad RIA.

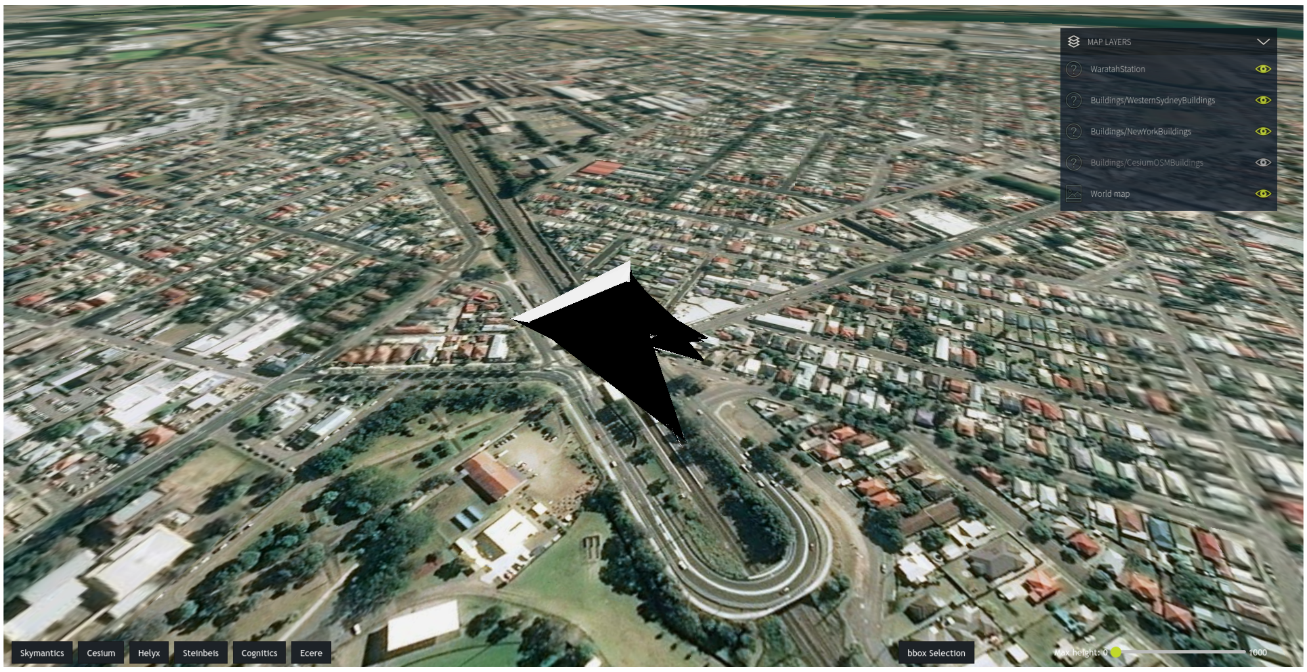


Figure 64. Cesium’s Waratah Station Dataset Rendered by Luciad RIA

The Cesium client also had issues rendering Luciad’s datasets. The images below show the church of Nôtre Dame in Marseille rendered by Luciad RIA and Steinbeis client (based on Cesium software). The Cesium client had similar issues.

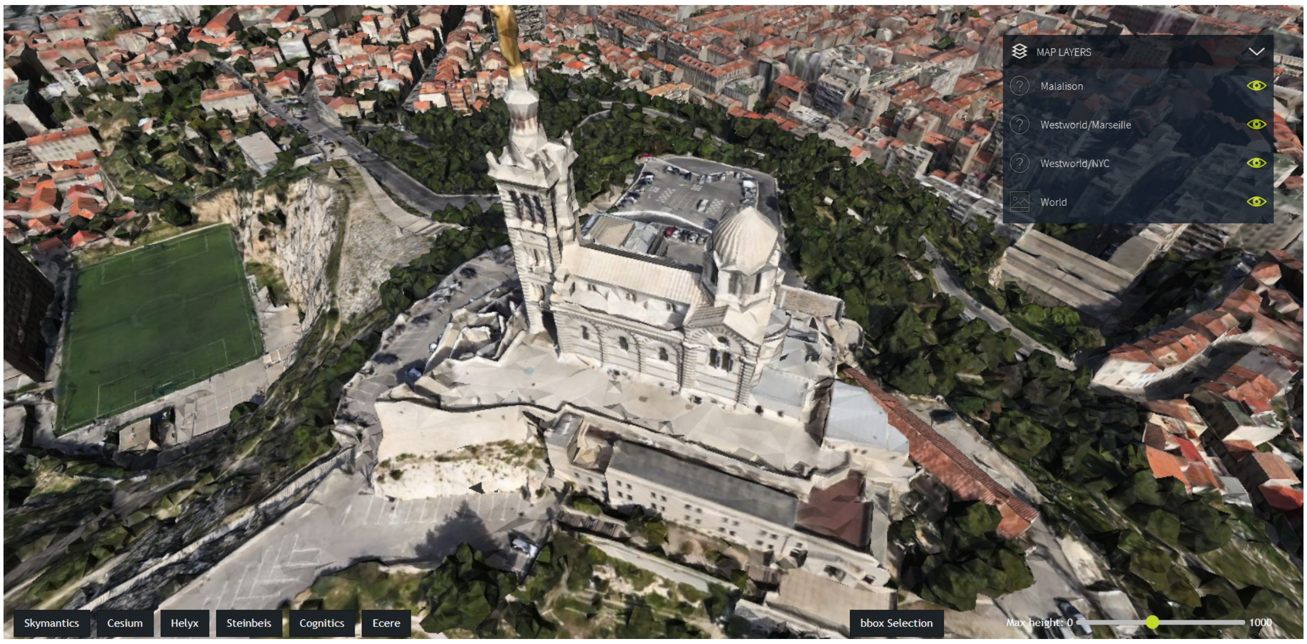
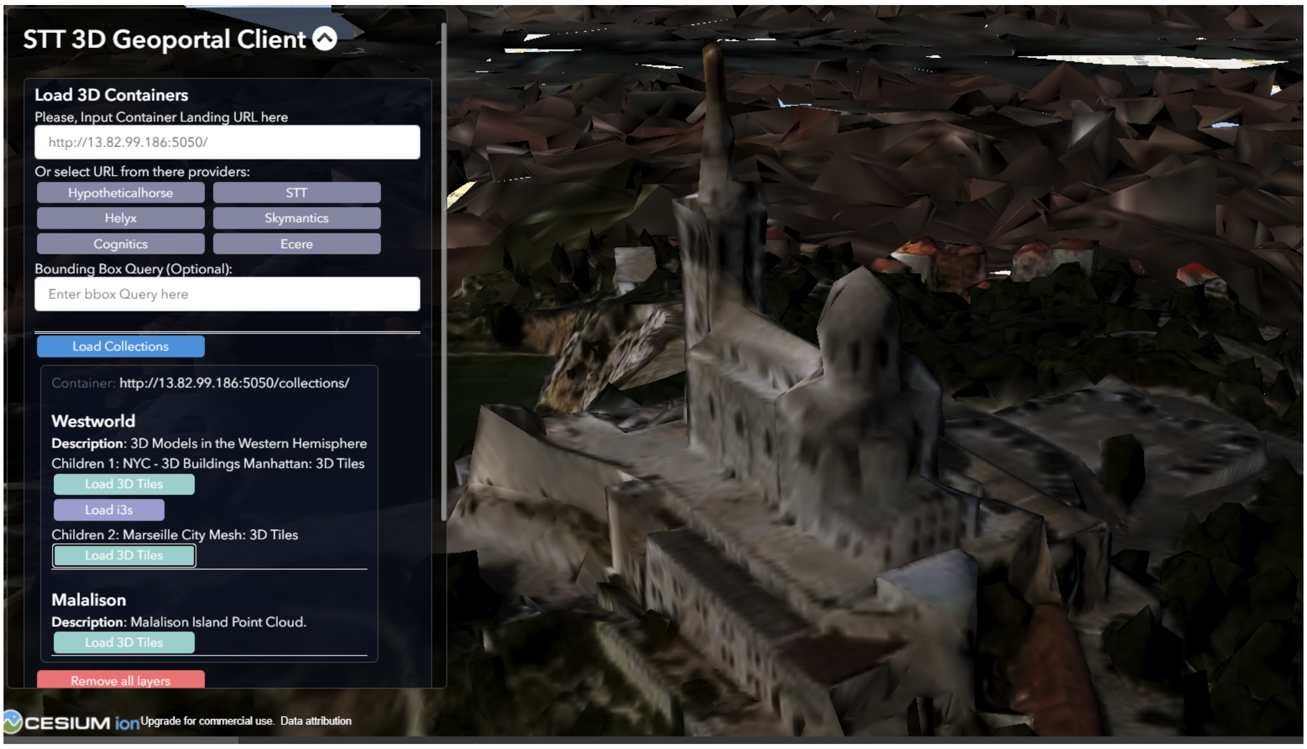


Figure 65. Church of Nôtre Dame (Marseille) Rendered by Luciad RIA


STT 3D Geoportal Client

Load 3D Containers

Please, Input Container Landing URL here

http://13.82.99.186:5050/

Or select URL from there providers:

Bounding Box Query (Optional):

Enter bbox Query here

Container: http://13.82.99.186:5050/collections/

Westworld

Description: 3D Models in the Western Hemisphere

Children 1: NYC - 3D Buildings Manhattan: 3D Tiles

Children 2: Marseille City Mesh: 3D Tiles

Malalison

Description: Malalison Island Point Cloud.

Figure 66. Church of Nôtre Dame (Marseille) Rendered by Steinbeis Client

Although the goal of this pilot was not to evaluate rendering operability among different clients, taking into account that 3D rendering is a core part of the 3D Information Management domain and that there were quite a few issues, this should be a high priority topic to address.

9.6.4.2. APIs can now combine 3D Tiles and I3S

This Pilot has drafted a new API specification 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 been also tested in this pilot by some participants. This is an important step forward, as it provides two distinct advantages:

1. APIs can be more interoperable, as they can offer two different representations for the same data. If a client only supports one representation, a client can still render the model.
2. For ecosystems where clients support both representations, by combining 3D Tiles and I3S representations to deliver a model, 3D Models can be richer and more performant. This allows using each standard for the use cases for which it is 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.

9.6.4.3. I3S-to-3DTiles tool can bridge the gap

Taking into account that most clients support 3D Tiles but few support I3S, the I3S-to-3DTiles tool tested during this pilot (or a similar tool) can help to quickly extend the support for I3S in clients. This can potentially accelerate the scenario where APIs offer both representations for the same data and where clients support both representations.

Unfortunately, the datasets generated through the I3S-to-3DTiles tool suffered rendering issues, as documented before, which is an issue hindering the adoption of this tool.

9.6.4.4. 3D Container federated catalogs

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 deploying 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).

The organization of Skymantic's datasets in this pilot is a good example of the potential of 3D Container federated catalogs:

1. The world is divided in several 3D Containers that can contain other 3D Containers and so on.
2. Datasets can be hosted in a myriad of servers. However, the API serves them as if they were hosted in a single server.
3. Some of these servers in the backend can be 3D Tiles APIs, potentially aggregating datasets from other servers and extending the network ad-infinitum.
4. The 3D Tiles and I3S representations of the same data do not need to reside in the same server, but can be fetched from different sources.

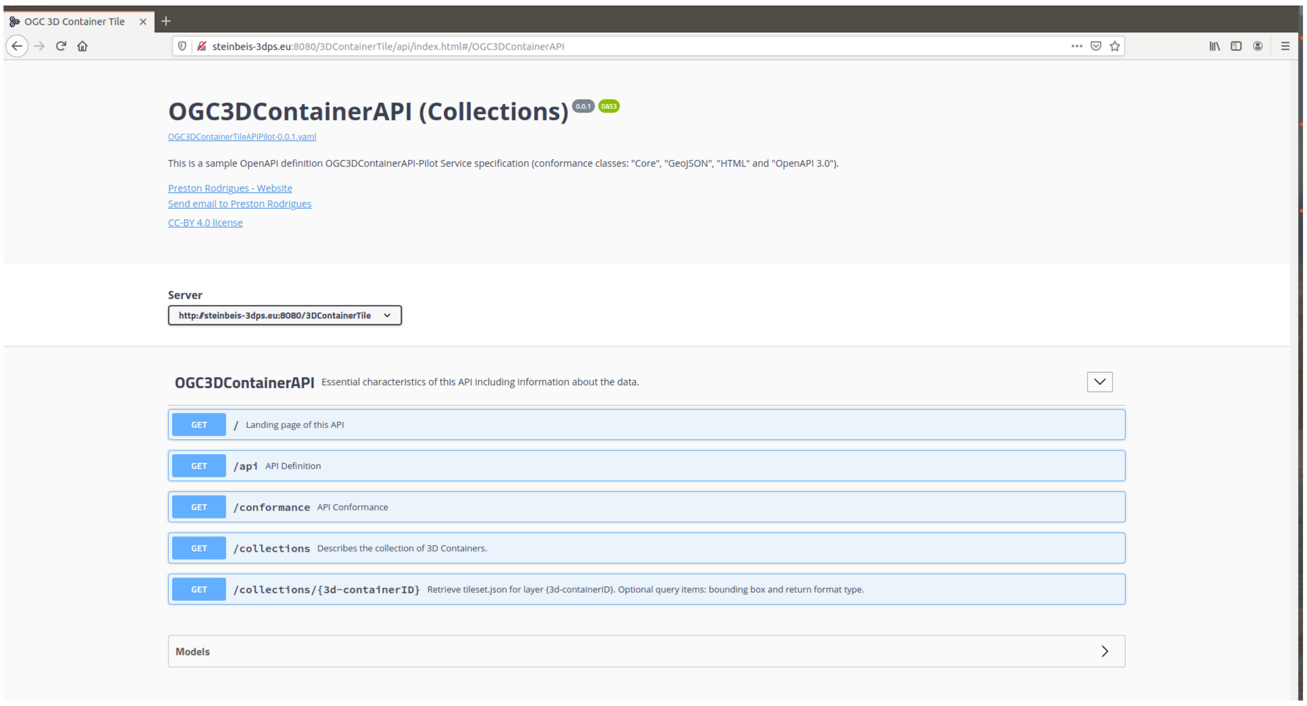
9.7. Steinbeis

Introduction

For this Pilot Steinbeis provided an API server to deliver 3D content supporting New York buildings, in both the 3D Tiles and I3S data delivery format. The API was implemented as specified in OGC API - Common, with general information on the API as well as links to the API definition, conformance classes and the 3D containers.

Server Functionality

All the functionalities supported by the server can be tested by interacting with the API available at http://steinbeis-3dps.eu:8080/3DContainerTile/api/index.html. To test a specific API call, first click on the *Try it out* button and then the *Execute* button.



OGC 3D Container Tile

steinbeis-3dps.eu:8080/3DContainerTile/api/index.html#/OGC3DContainerAPI

OGC3DContainerAPI (Collections) 0.0.1 OAS3

OGC3DContainerTileAPIPilot-0.0.1.yaml

This is a sample OpenAPI definition OGC3DContainerAPI-Pilot Service specification (conformance classes: "Core", "GeoJSON", "HTML" and "OpenAPI 3.0").

Preston Rodrigues - Website
Send email to Preston Rodrigues
CC-BY 4.0 license

Server
http://steinbeis-3dps.eu:8080/3DContainerTile

OGC3DContainerAPI Essential characteristics of this API including information about the data.

GET `/` Landing page of this API
GET `/api` API Definition
GET `/conformance` API Conformance
GET `/collections` Describes the collection of 3D Containers.
GET `/collections/{3d-containerID}` Retrieve tileset.json for layer (3d-containerID). Optional query items: bounding box and return format type.

Models

Figure 67. Steinbeis API

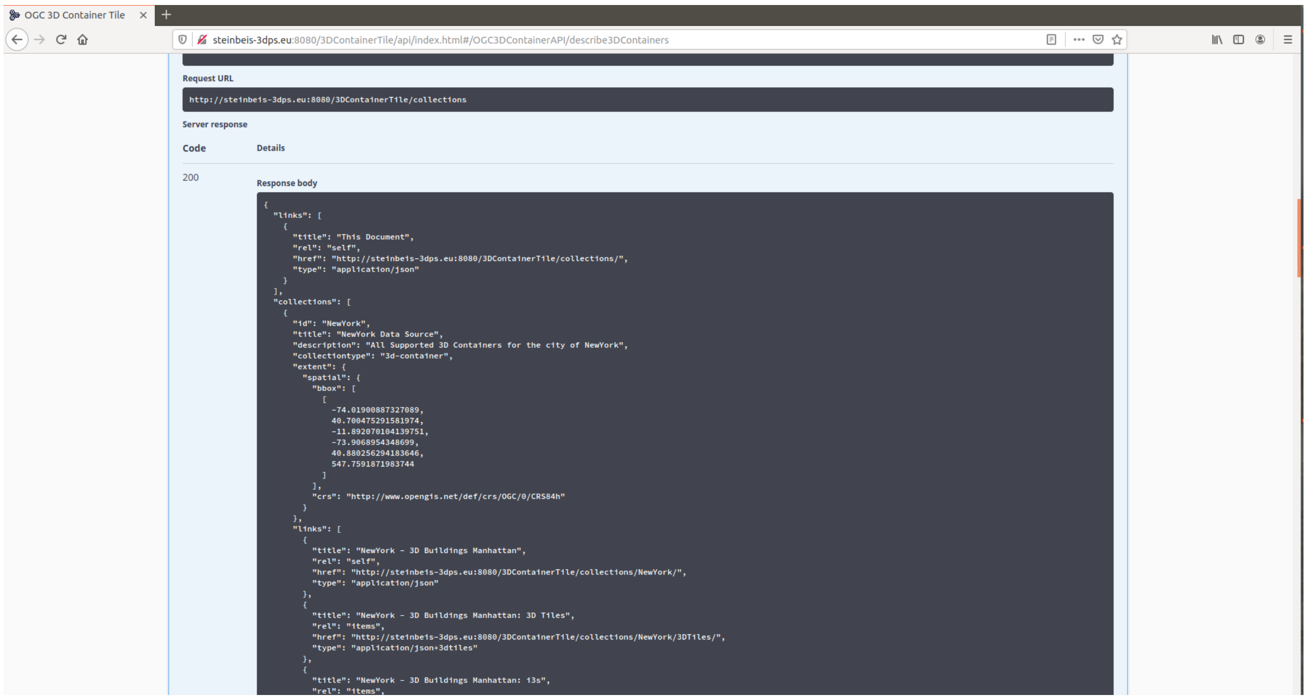
- Landing Page

The Landing page is available at http://steinbeis-3dps.eu:8080/3DContainerTile. The result of a successfully request is shown below.

```
{
  "title":"3D Container and Tiles Pilot Steinbeis API",
  "description":"Steinbeis API for OGC 3D Container and Tiles Pilot",
  "links":[
    {
      "title":"Collections",
      "rel":"data",
      "href":"http://steinbeis-3dps.eu:8080/3DContainerTile/collections/",
      "type":"application/json"
    },
    {
      "title":"Conformance",
      "rel":"conformance",
      "href":"http://steinbeis-3dps.eu:8080/3DContainerTile/conformance/",
      "type":"application/json"
    },
    {
      "title":"Service Description",
      "rel":"service-desc",
      "href":"https://app.swaggerhub.com/apis/timothy-
miller/OGC_3DContainer_Tile_API_Pilot/0.0.1/",
      "type":"application/openapi+json;version=3.0"
    },
    {
      "title":"Service API",
      "rel":"service",
      "href":"http://steinbeis-3dps.eu:8080/3DContainerTile/api/index.html",
      "type":"text/html"
    }
  ]
}
```

- 3D Container

The supported 3D collections is available at http://steinbeis-3dps.eu:8080/3DContainerTile/collections/ The result of a successfully request is shown below.


OGC 3D Container Tile
steinbeis-3dps.eu:8080/3DContainerTile/api/index.html#/OGC3DContainerAPI/describe3DContainers

Request URL

```
http://steinbeis-3dps.eu:8080/3DContainerTile/collections
```

Server response

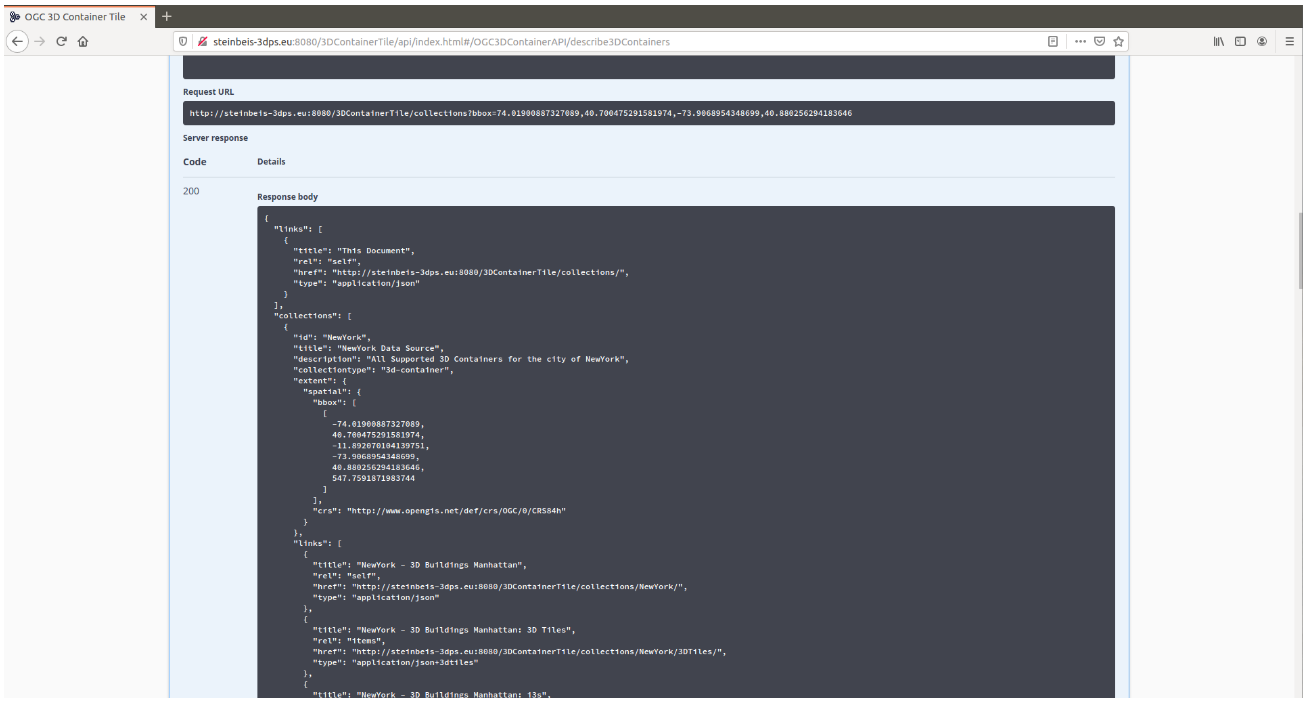
Code Details

200 Response body

```
{
  "links": [
    {
      "title": "This Document",
      "rel": "self",
      "href": "http://steinbeis-3dps.eu:8080/3DContainerTile/collections/",
      "type": "application/json"
    }
  ],
  "collections": [
    {
      "id": "NewYork",
      "title": "NewYork Data Source",
      "description": "All Supported 3D Containers for the city of NewYork",
      "collectiontype": "3d-container",
      "extent": {
        "spatial": {
          "bbox": [
            [
              -74.01900887327089,
              40.700475291581974,
              -11.892070104139751,
              -73.9068954348699,
              40.880256294183646,
              547.7591871983744
            ]
          ],
          "crs": "http://www.opengis.net/def/crs/OGC/0/CRS84h"
        }
      },
      "links": [
        {
          "title": "NewYork - 3D Buildings Manhattan",
          "rel": "self",
          "href": "http://steinbeis-3dps.eu:8080/3DContainerTile/collections/NewYork/",
          "type": "application/json"
        },
        {
          "title": "NewYork - 3D Buildings Manhattan: 3D Tiles",
          "rel": "items",
          "href": "http://steinbeis-3dps.eu:8080/3DContainerTile/collections/NewYork/3DTiles/",
          "type": "application/json+3dtiles"
        },
        {
          "title": "NewYork - 3D Buildings Manhattan: i3s",
          "rel": "items",
```

- Bounding Box query

The server also supports a filtering functionality in the form of a bounding box. The bounding box support is at both collection level and 3D container level. The result of a successfully request at collection level is shown below.


OGC 3D Container Tile
steinbeis-3dps.eu:8080/3DContainerTile/api/index.html#/OGC3DContainerAPI/describe3DContainers

Request URL

```
http://steinbeis-3dps.eu:8080/3DContainerTile/collections?bbox=74.01900887327089,40.700475291581974,-73.9068954348699,40.880256294183646
```

Server response

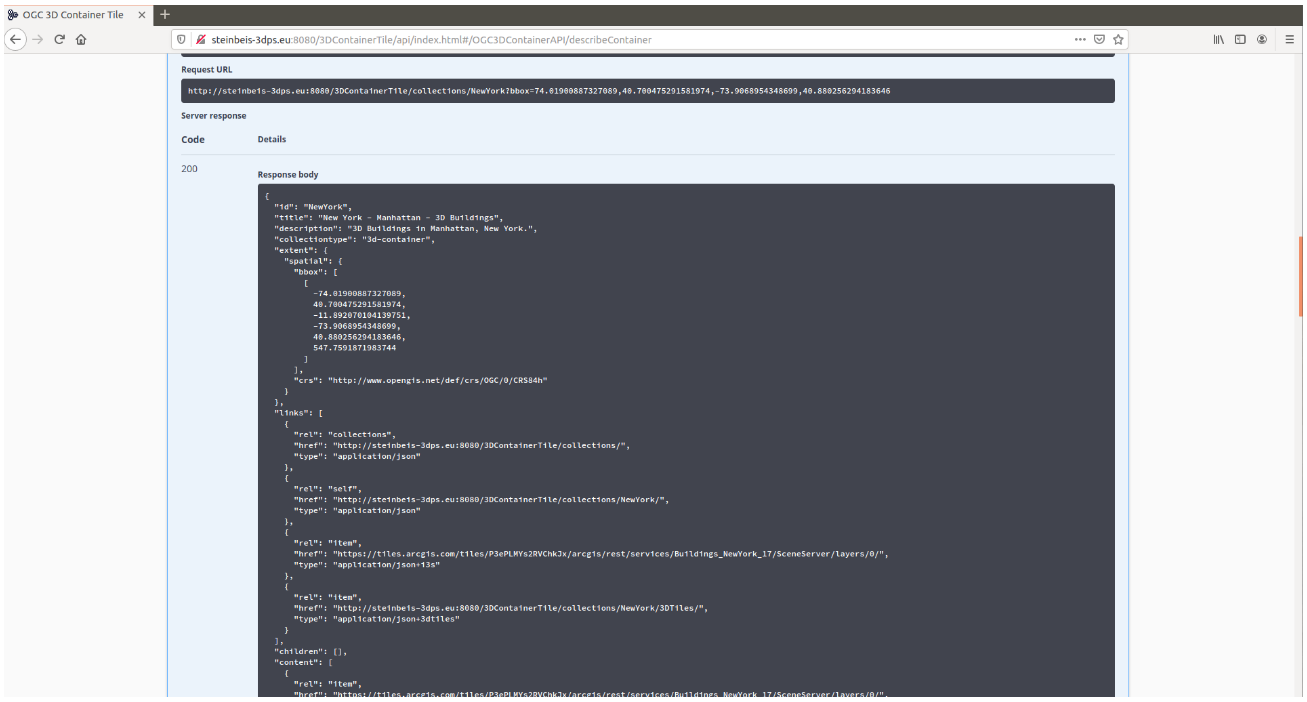
Code Details

200 Response body

```
{
  "links": [
    {
      "title": "This Document",
      "rel": "self",
      "href": "http://steinbeis-3dps.eu:8080/3DContainerTile/collections/",
      "type": "application/json"
    }
  ],
  "collections": [
    {
      "id": "NewYork",
      "title": "NewYork Data Source",
      "description": "All Supported 3D Containers for the city of NewYork",
      "collectiontype": "3d-container",
      "extent": {
        "spatial": {
          "bbox": [
            [
              -74.01900887327089,
              40.700475291581974,
              -11.892070104139751,
              -73.9068954348699,
              40.880256294183646,
              547.7591871983744
            ]
          ],
          "crs": "http://www.opengis.net/def/crs/OGC/0/CRS84h"
        }
      },
      "links": [
        {
          "title": "NewYork - 3D Buildings Manhattan",
          "rel": "self",
          "href": "http://steinbeis-3dps.eu:8080/3DContainerTile/collections/NewYork/",
          "type": "application/json"
        },
        {
          "title": "NewYork - 3D Buildings Manhattan: 3D Tiles",
          "rel": "items",
          "href": "http://steinbeis-3dps.eu:8080/3DContainerTile/collections/NewYork/3DTiles/",
          "type": "application/json+3dtiles"
        },
        {
          "title": "NewYork - 3D Buildings Manhattan: i3s",
```

http://steinbeis-3dps.eu:8080/3DContainerTile/collections?bbox=-74.01900887327089,40.700475291581974,-73.9068954348699,40.880256294183646

The result of a successful request at 3D container level is shown below.



OGC 3D Container Tile
steinbeis-3dps.eu:8080/3DContainerTile/api/index.html#/OGC3DContainerAPI/describeContainer

Request URL

```
http://steinbeis-3dps.eu:8080/3DContainerTile/collections/NewYork?bbox=74.01900887327089,40.700475291581974,-73.9068954348699,40.880256294183646
```

Server response

Code Details

200

Response body

```
{
  "id": "NewYork",
  "title": "New York - Manhattan - 3D Buildings",
  "description": "3D Buildings in Manhattan, New York.",
  "collectiontype": "3d-container",
  "extent": {
    "spatial": {
      "bbox": [
        [
          -74.01900887327089,
          40.700475291581974,
          -11.892070104139751,
          -73.9068954348699,
          40.880256294183646,
          547.7591871983744
        ]
      ],
      "crs": "http://www.opengis.net/def/crs/OGC/0/CRS84h"
    }
  },
  "links": [
    {
      "rel": "collections",
      "href": "http://steinbeis-3dps.eu:8080/3DContainerTile/collections/",
      "type": "application/json"
    },
    {
      "rel": "self",
      "href": "http://steinbeis-3dps.eu:8080/3DContainerTile/collections/NewYork/",
      "type": "application/json"
    },
    {
      "rel": "item",
      "href": "https://tiles.arcgis.com/tiles/P3ePLMYs2RVChkJx/arcgis/rest/services/Buildings_NewYork_17/SceneServer/layers/0/",
      "type": "application/json+i3s"
    },
    {
      "rel": "item",
      "href": "http://steinbeis-3dps.eu:8080/3DContainerTile/collections/NewYork/3DTiles/",
      "type": "application/json+3dtiles"
    }
  ],
  "children": [],
  "content": [
    {
      "rel": "item",
      "href": "https://tiles.arcgis.com/tiles/P3ePLMYs2RVChkJx/arcgis/rest/services/Buildings_NewYork_17/SceneServer/layers/0/",
```

http://steinbeis-3dps.eu:8080/3DContainerTile/collections?bbox=-74.01900887327089,40.700475291581974,-73.9068954348699,40.880256294183646

Steinbeis Geoportal Client Functionality

- Request and parse Landing Page
- Request and parse 3D container

The STT 3D web client (http://steinbeis-3dps.eu/STT3DClient/) 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. The CesiumJS library is used to access and visualize the 3D Tiles contents and the ArcGIS for JavaScript library is used to interact with the I3S contents. The main user interface of the client is shown in the figure below and consisted of two main sections: the menu section and the map section.

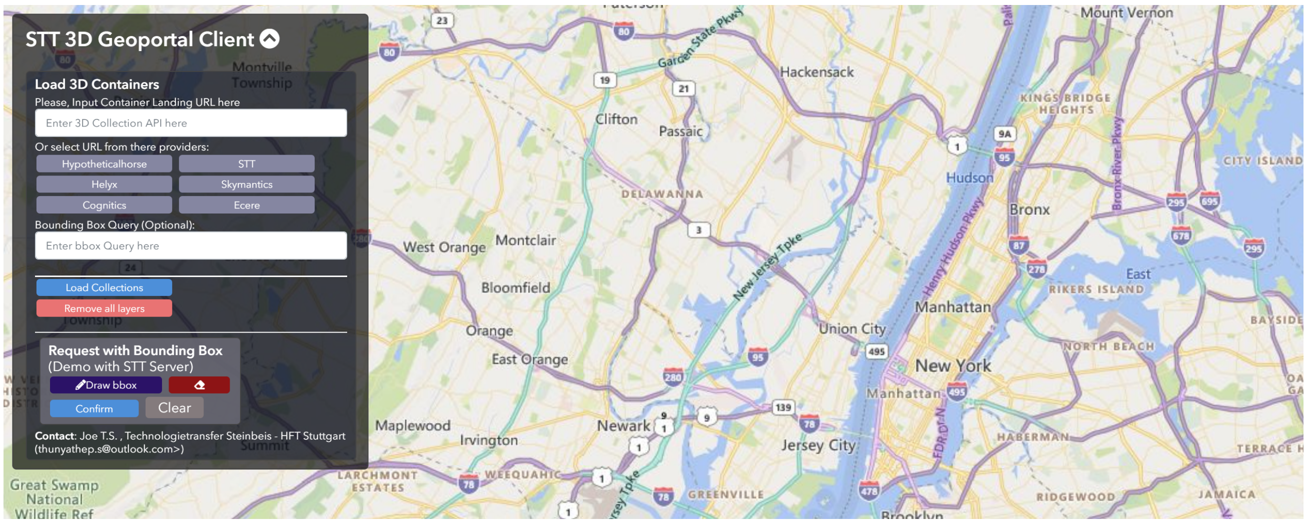
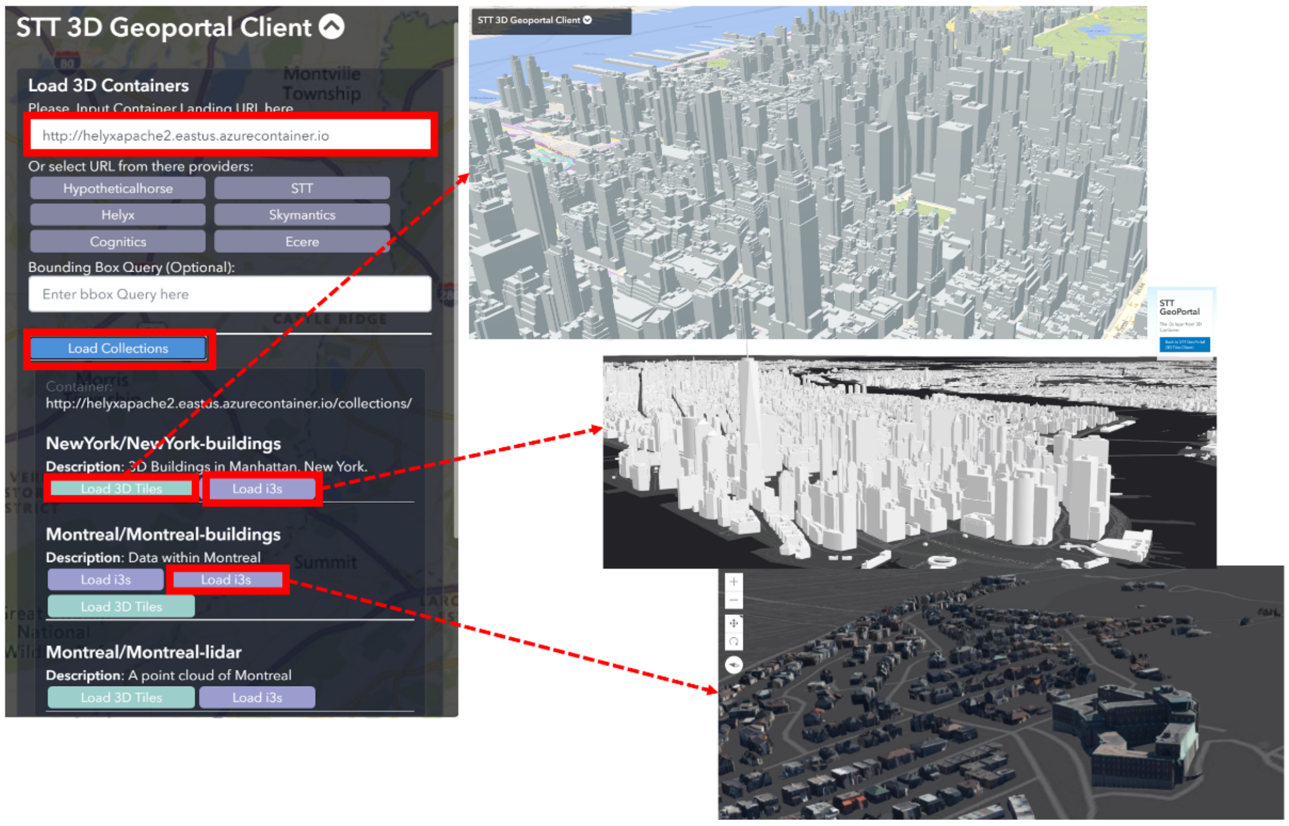


Figure 68. Client User Interface

In the menu section, users may input the target OGC API-Tiles-3D landing page URL or select a server from given server providers (Cesium, STT, Helyx, Skymantics, Cognitics, or Ecere). Then, users may input the bounding box to filter out the layer. After that, by clicking on the "Load Collections" button, the client will fetch the list of available data collections from the input server. For example, the figure below shows the STT client fetching the 3D data from the Helyx OGC API-Tiles-3D server. The 3D building models data are from the area of New York City, USA and Montreal, Canada.



http://helyxapache2.eastus.azurecontainer.io

Container:
http://helyxapache2.eastus.azurecontainer.io/collections/

NewYork/NewYork-buildings
Description: 3D Buildings in Manhattan, New York.

Montreal/Montreal-buildings
Description: Data within Montreal

Montreal/Montreal-lidar
Description: A point cloud of Montreal

Figure 69. Client Fetching 3D Data

Integration with the OGC 3D Portrayal Services

In the Pilot, the OGC 3D Portrayal Service (3DPS) was used in the geoportal clients extended to the 3D Data Container and Tiles API. Using the 3DPS, the clients could request getting only 3D contents that fall within the requested bounding box, while the 3D Data Container and Tiles API would respond with all the intersected 3D contents.

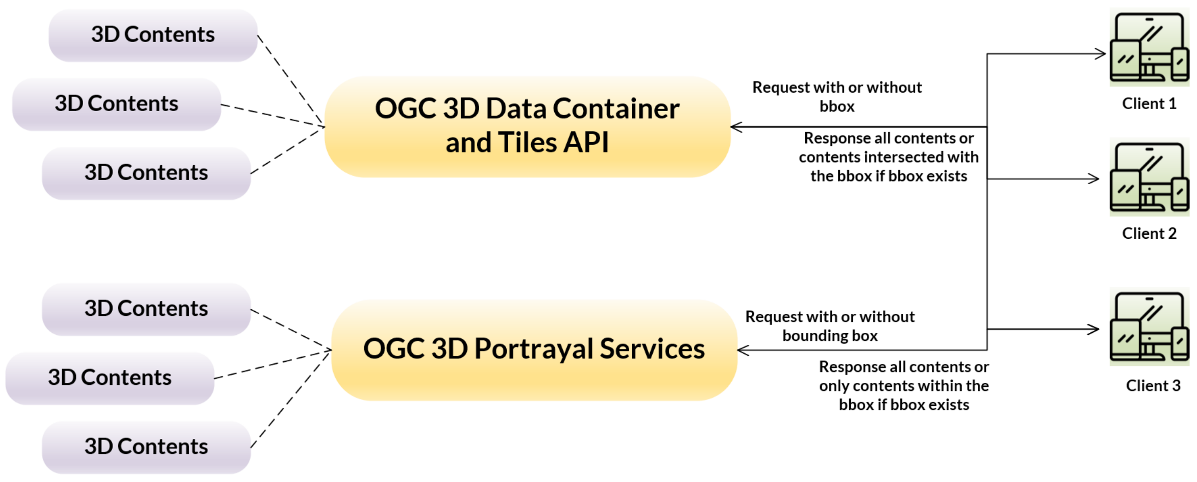

3D Contents
3D Contents
3D Contents
OGC 3D Data Container and Tiles API
Request with or without bbox
Response all contents or contents intersected with the bbox if bbox exists
Client 1
Client 2
3D Contents
3D Contents
3D Contents
OGC 3D Portrayal Services
Request with or without bounding box
Response all contents or only contents within the bbox if bbox exists
Client 3

Figure 70. Integration with OGC 3D Portrayal Services

For example, in the Steinbeis geoportal client, users can request 3D contents via the 3DPS interface 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.

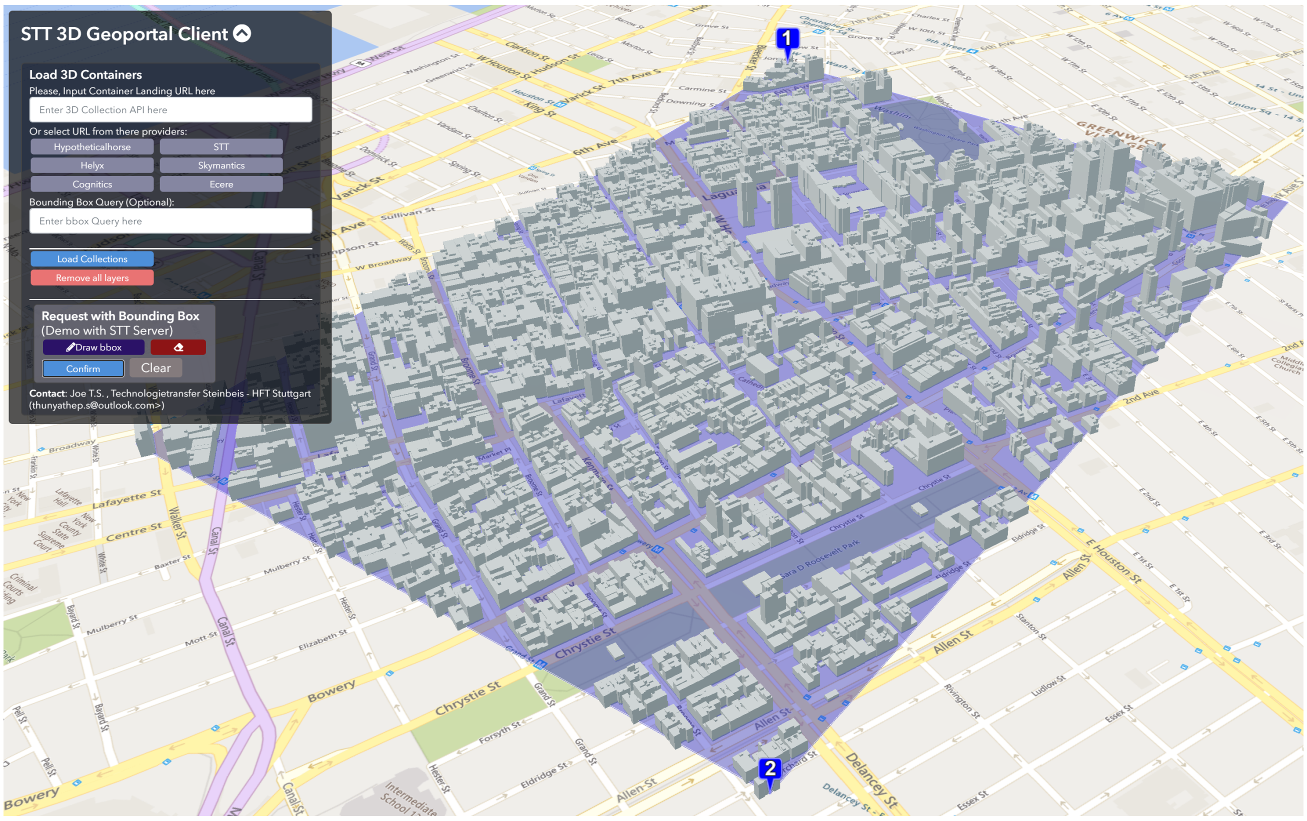

STT 3D Geoportal Client
Load 3D Containers
Please, Input Container Landing URL here
Enter 3D Collection API here
Or select URL from there providers:
Hypotheticalhorse
STT
Helyx
Skymantics
Cognitics
Ecere
Bounding Box Query (Optional):
Enter bbox Query here
Load Collections
Remove all layers
Request with Bounding Box (Demo with STT Server)
Draw bbox
Confirm
Clear
Contact: Joe T.S. , Technologietransfer Steinbeis - HFT Stuttgart (thunyathep.s@outlook.com>)
1
2

Figure 71. Request 3D Contents via 3DPS Interface by Bounding Area

Steinbeis client implementation using deck.gl

A second client was implemented by Steinbeis using the JavaScript library Deck.gl. This client is able to visualize both 3D Tiles and I3S content in a single client. For example, the figure below shows the visualization of the 3D building models in the New York City area in I3S (left) and 3D Tiles (right) format.

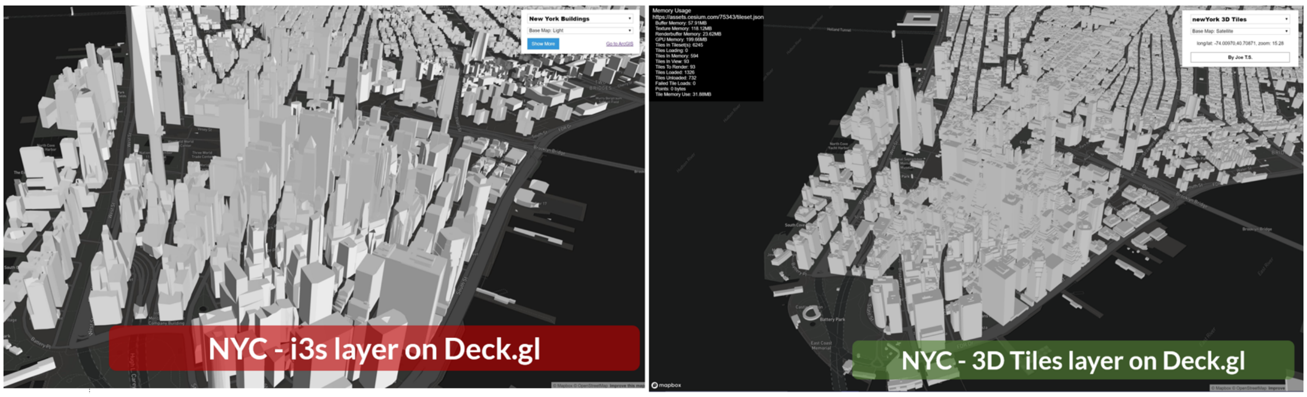
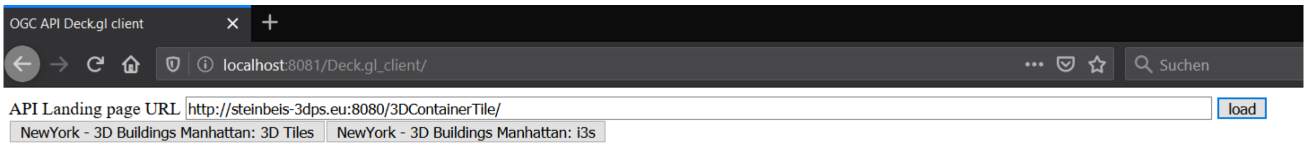


Figure 72. Client Implementation Using deck.gl

The Client uses native JavaScript to call and parse the API landing pages. An HTML site allows the user to input the URL of an API landing page and trigger the loading and processing by the client functionalities. The upper part of the website is used to specify and load the landing page. The loading buttons for each I3S or 3DTiles dataset are rendered below the input field. The response JSON objects are parsed to get the name, data type and URL of the 3D containers on the server. The extracted information about the datasets is then used to dynamically generate the loading buttons for the 3D content. The client configures the viewer section according to the same attributes, after clicking on the desired dataset.



API Landing page URL http://steinbeis-3dps.eu:8080/3DContainerTile/ load
NewYork - 3D Buildings Manhattan: 3D Tiles | NewYork - 3D Buildings Manhattan: i3s

Figure 73. Landing Page URL Input

The framework can also load different 2D background maps to augment the 3D container content in the viewer. These maps use a classical regular tiling approach. A Mapbox background map is used in the screenshots above. Loading another dataset from the same server requires the reload of the whole website, and a new request to the API landing page.

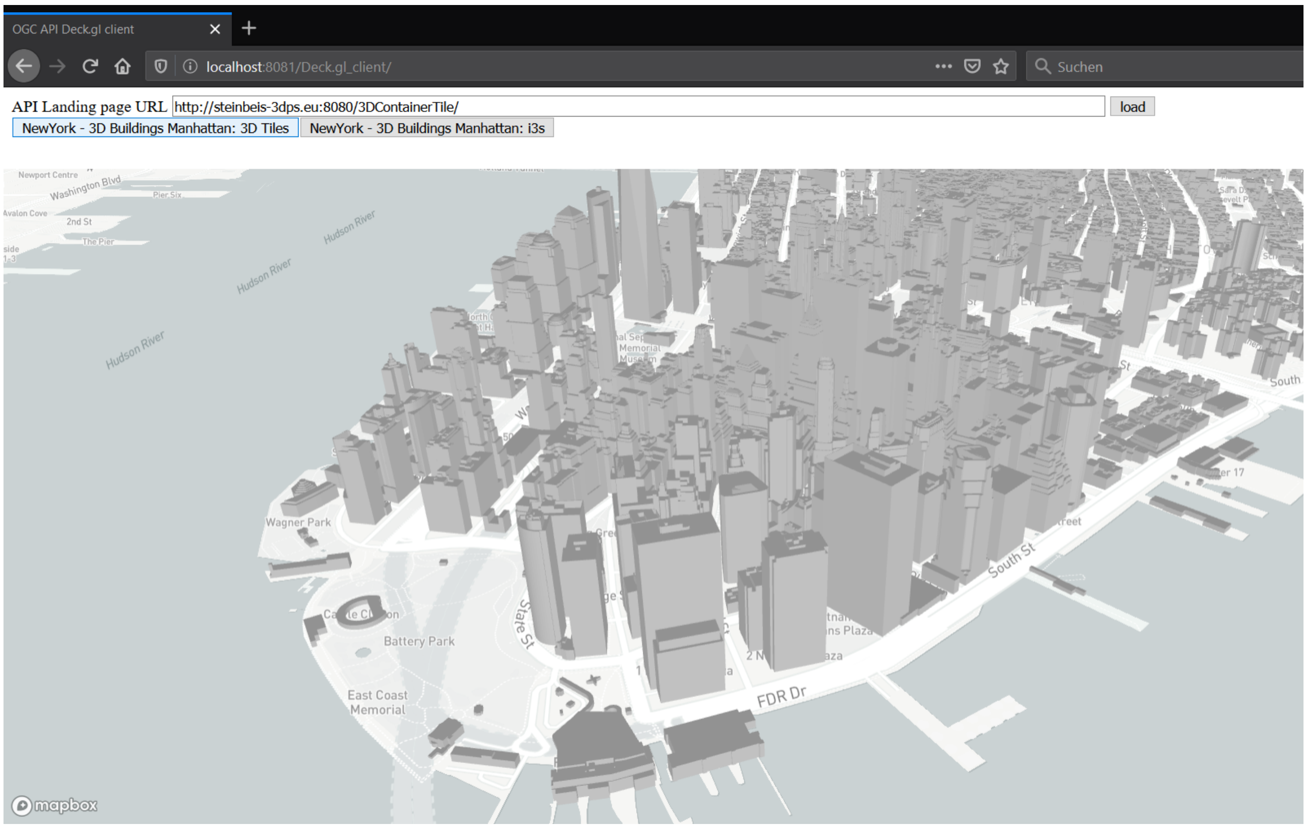

OGC API Deck.gl client
localhost:8081/Deck.gl_client/
Suchen
API Landing page URL
http://steinbeis-3dps.eu:8080/3DContainerTile/
load
NewYork - 3D Buildings Manhattan: 3D Tiles
NewYork - 3D Buildings Manhattan: i3s
Hudson River
Battery Park
East Coast Memorial
Wagner Park
FDR Dr
South St
mapbox

Figure 74. 2D Background Maps Augment 3D Container Content in Viewer

Integration with the OGC API features

In the Pilot, the geoportal client can be extended with the OGC API features, which provide a function to query and export geospatial contents through the HTTP GET request. As shown in the figure below, the clients can both visualize and export the 3D contents over the internet by using the OGC 3D Data Container and Tiles API and OGC API Features service.

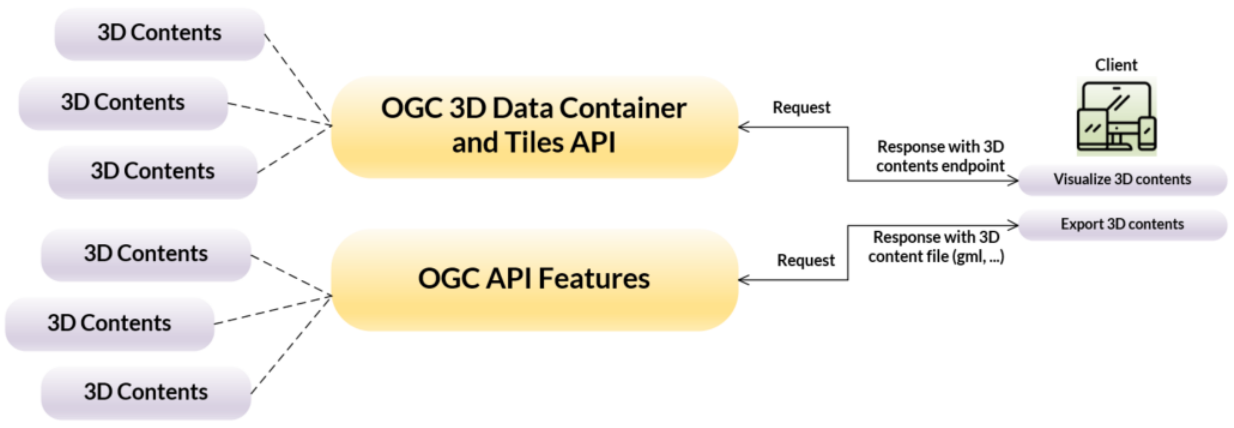
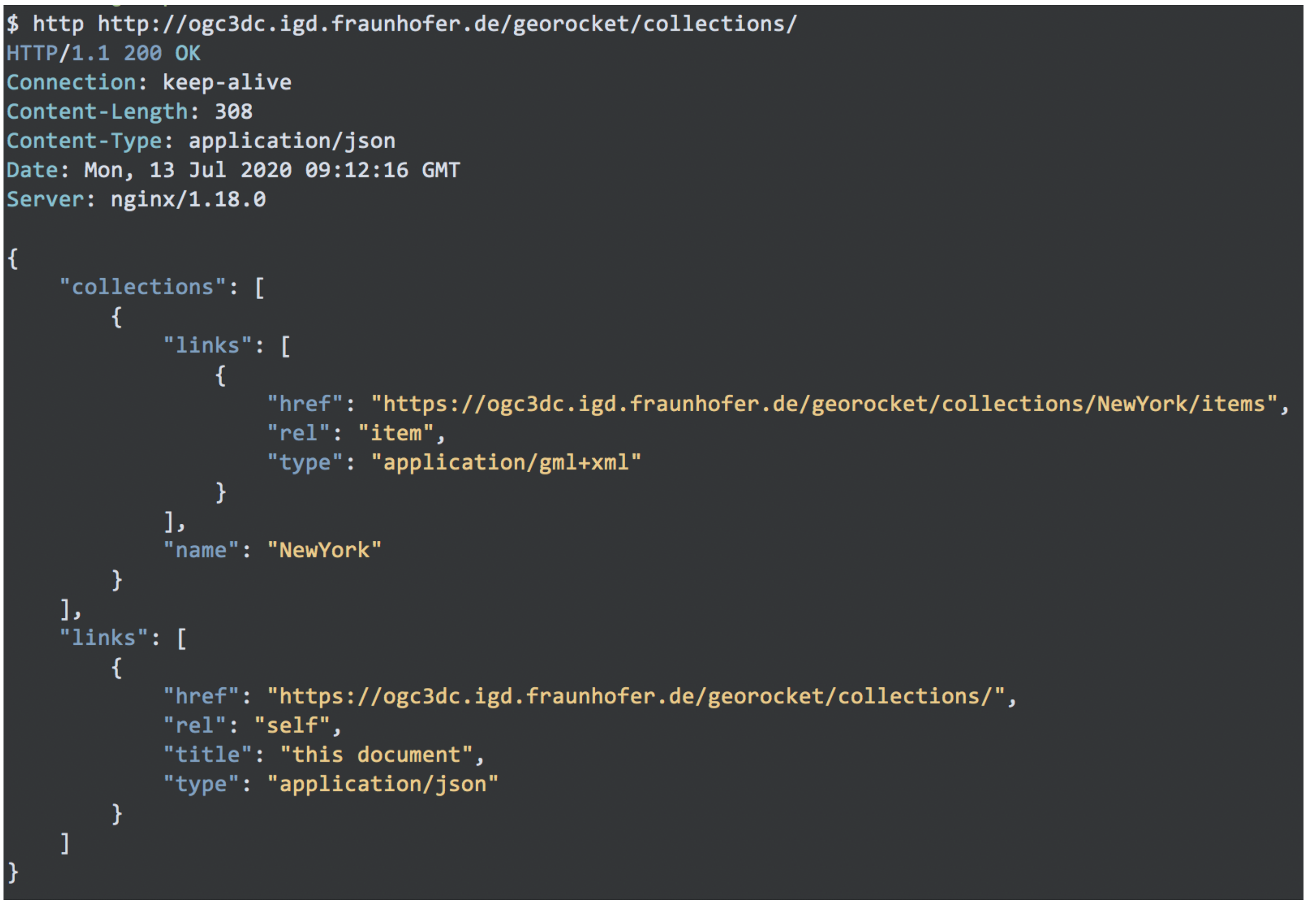


Figure 75. Integration with OGC API Features

A server was set that provided the original CityGML files of the New York dataset. The server is based on the open-source product GeoRocket (https://georocket.io) developed by the Fraunhofer Institute for Computer Graphics Research IGD. Besides the native GeoRocket HTTP API, an interface was developed that adheres to the OGC API – Features specification.

The server is available under http://ogc3dc.igd.fraunhofer.de/georocket. The OGC API – Features “collections” interface can be accessed via http://ogc3dc.igd.fraunhofer.de/georocket/collections/, for example.



```
$ http http://ogc3dc.igd.fraunhofer.de/georocket/collections/
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 308
Content-Type: application/json
Date: Mon, 13 Jul 2020 09:12:16 GMT
Server: nginx/1.18.0

{
    "collections": [
        {
            "links": [
                {
                    "href": "https://ogc3dc.igd.fraunhofer.de/georocket/collections/NewYork/items",
                    "rel": "item",
                    "type": "application/gml+xml"
                }
            ],
            "name": "NewYork"
        }
    ],
    "links": [
        {
            "href": "https://ogc3dc.igd.fraunhofer.de/georocket/collections/",
            "rel": "self",
            "title": "this document",
            "type": "application/json"
        }
    ]
}
```

Here’s an example request to download the first 100 buildings from Manhattan Soho via a bounding box:

```
http://ogc3dc.igd.fraunhofer.de/georocket/collections/NewYork/items?bbox=-
74.00488275167392,40.72182445888009,-73.9954120347942,40.72649940579886&limit=100
```

The result will be provided as a CityGML file.

In the Fraunhofer client (http://ogc3dc.igd.fraunhofer.de/), a client application was implemented that serves two purposes: 1. It is a generic viewer for 3D tiles provided by other server implementations in the pilot and 2. a user interface to query CityGML data from GeoRocket with a study area in New York City, USA.

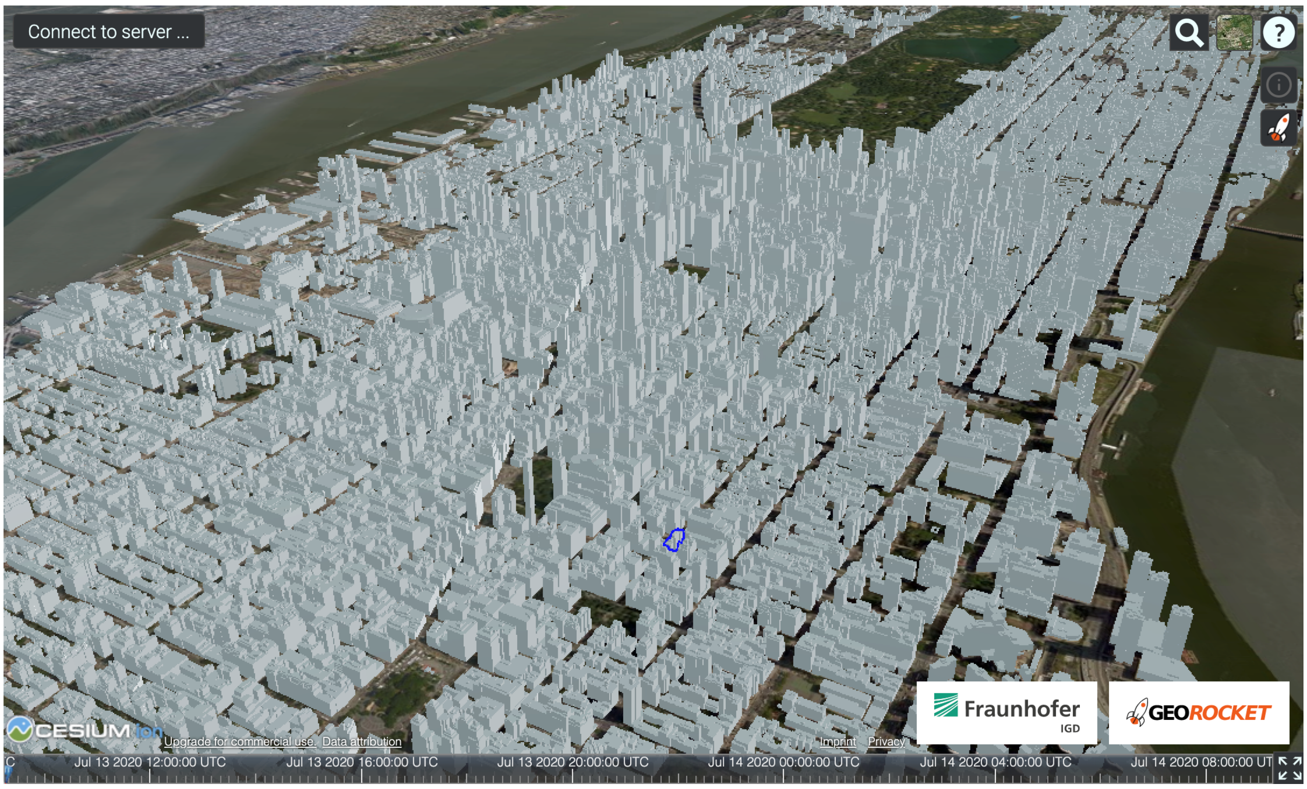


Figure 76. Fraunhofer Client

In the upper left corner, there is a drop-down menu to select different pilot server implementations:

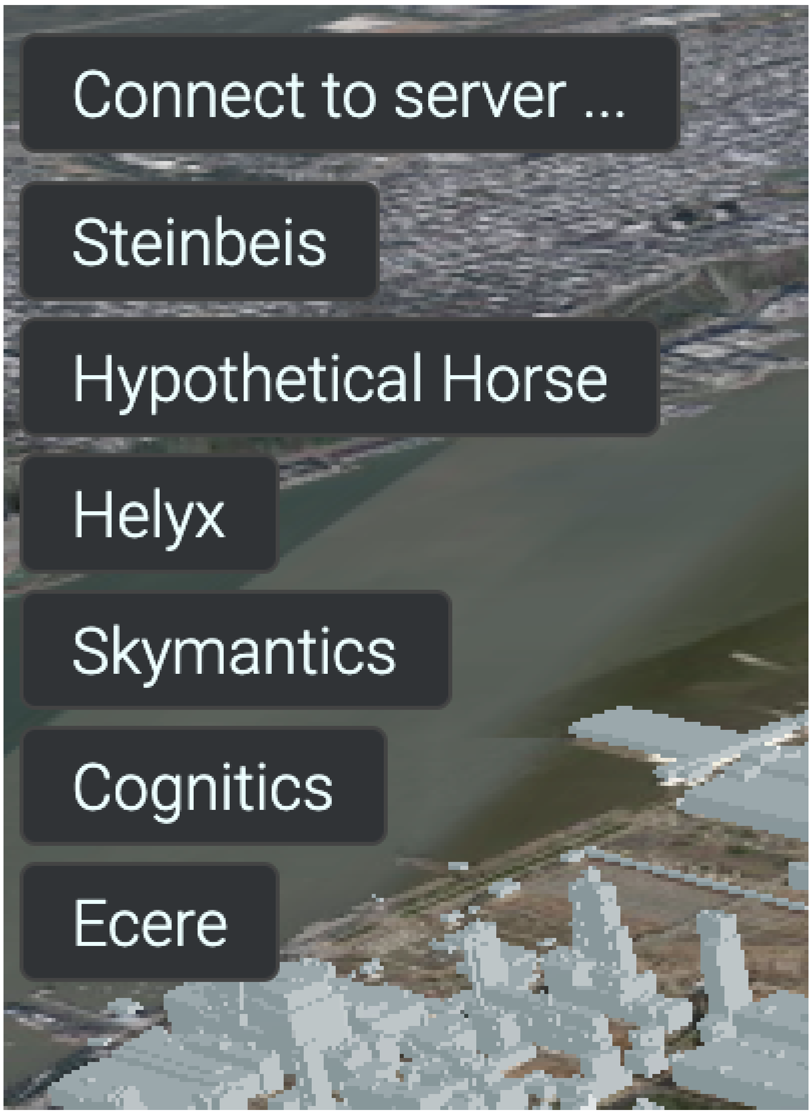
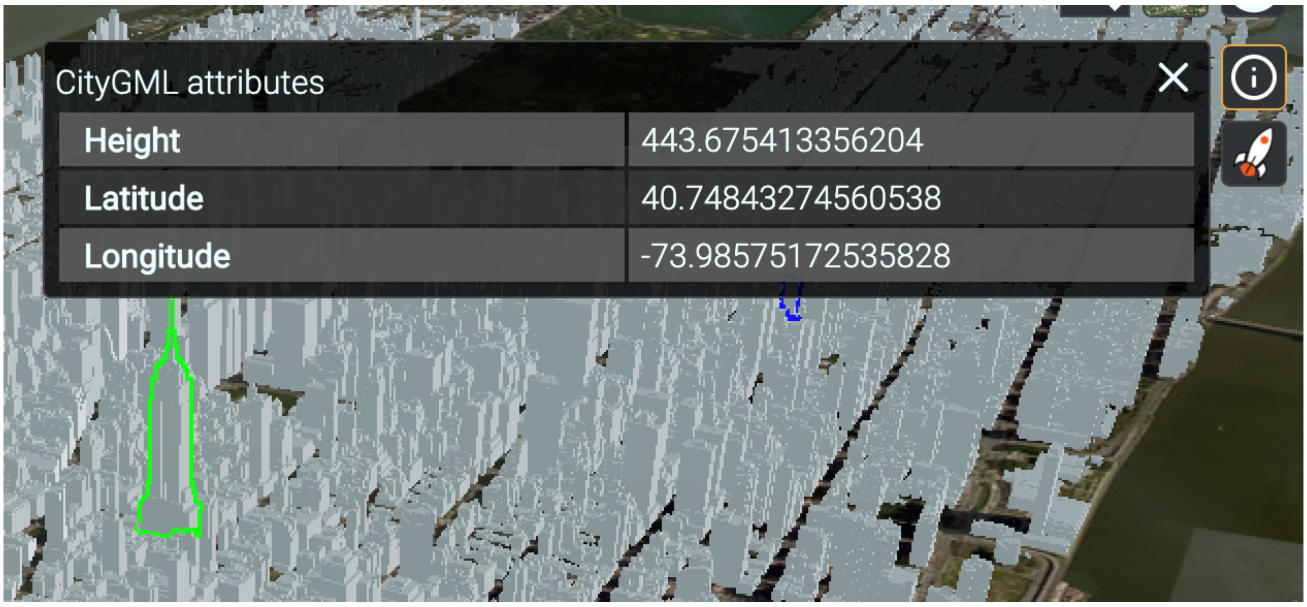


Figure 77. Pilot Server Implementations Menu

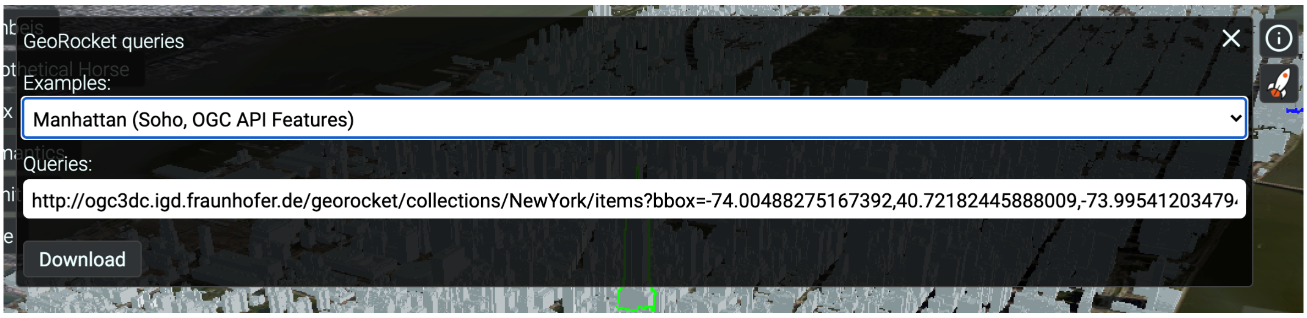
In the upper right corner, there are tools to interact with the 3D scene and the server. Click on a 3D building (to select it) and then press the info button (labeled with a capital “I”) to get metadata from the CityGML file:


CityGML attributes

Height	443.675413356204
Latitude	40.74843274560538
Longitude	-73.98575172535828

Figure 78. CityGML File Metadata

Furthermore, the “GeoRocket” button can query CityGML data from the server:


GeoRocket queries

Examples:

Manhattan (Soho, OGC API Features)

Queries:

http://ogc3dc.igd.fraunhofer.de/georocket/collections/NewYork/items?bbox=-74.00488275167392,40.72182445888009,-73.99541203479

Figure 79. GeoRocket Query

Chapter 10. Technology Integration Experiments (TIEs)

The Technology Integration Experiments were organized along three uses cases: (1) core functionality, (2) spatial extension, and (3) 3D model extension. Core functionality is focused on requesting, receiving, and parsing container content. Core functionality discussed in Table 11. Core functionality TIE results in Table 12. Spatial extension functionality requesting, receiving, and parsing container content with a bounding box query. Spatial extension functionality discussed in Table 13. Spatial extension TIE results in Table 14.

All Pilot participants successfully completed the core functionality and spatial extensions from their respective client and server implementations.

10.1. 3D Tiles Static API - Core API Functionality

#	Function	Description	Client Action	Server Response	Success Criterion
1	Landing Page	Request, receive, parse landing page	Form landing page request, parse response, display and/or act on it	Receive landing page request, response with accepted / requested format (f=json, html)	Client displays API "data" link and/or navigates to it
2	Collections Info	Request, receive, parse array of 3dContainer json objects in Collections document	Follow "data" link in landing page and receive / parse / display Collections document	Receive URL dereference from client, return Collections document	Client displays in menu and/or map form the Collections document and constituent 3dContainers

#	Function	Description	Client Action	Server Response	Success Criterion
3	3D Container Info	Request, receive, parse a 3dContainer json document	Form 3dContainer request, parse response, display in client for user selection as menu and/or map	Receive request, form and return requested / acceptable format	Client displays and/or navigates to 3dContainer spatial extent, displays links to "contents" and "children"
4	3D Content Distribution	Parse and resolve links in 3dContainer "content" array	Parse 3dContainer "content" array for user selection, then dereference selection and parse media returned by dereference	Return media referenced by the client with the media type specified in the link item.	Client displays content described and/or contained by the returned 2D / 3D media

Table 11. Core Functionality

Service\Client	Cesium D112	Cognitics D112	Ecere D112	Skymantics D112	Steinbeis D110
Cesium D102	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
Cognitics D102	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
Ecere D102	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
Helyx D100	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
Skymantics D102	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
Steinbeis D100	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4

Table 12. Technology Integration Experiments (TIE)

10.2. 3D Tiles API - Extension Functionality

#	Function	Description	Client Action	Server Response	Success Criterion
5	Collections Bbox Query	Query published Collections by bbox extent	Client adds bbox parameter to Collections URL, sends GET request to server.	Server receives GET request from client, returns Collections document with collections array of only the highest level 3dContainers whose extent(s) are wholly within the requested bbox	Client forms and issues bbox query, receives and displays filtered Collections document.
6	3dContainer Bbox Query	Query published 3dContainers by bbox extent	Client adds bbox parameter to 3dContainer URL, sends GET request to server.	Server receives GET request from client, returns 3dContainer document with "children" array of only highest level 3dContainer children whose extent(s) are wholly within the requested bbox	Client forms and issues bbox query, receives and displays filtered 3dContainer document

Table 13. Advanced Spatial Query Functionality

Service\Client	Cesium D112	Cognitics D112	Ecere D112	Skymantics D112	Steinbeis D110
Cesium D102	5, 6	5, 6		5, 6	5, 6

Service\Client	Cesium D112	Cognitics D112	Ecere D112	Skymantics D112	Steinbeis D110
Cognitics D102	5, 6	5,6		5, 6	5,6
Ecere D102					
Helyx D100	5				5
Skymantics D102		5, 6		5, 6	5, 6
Steinbeis D100	5, 6	5, 6		5, 6	5, 6

Table 14. Technology Integration Experiments (TIE)

Technology Integration Experiments using OGC API - Tiles were also performed between the Ecere server (D102) & client (D112).

Chapter 11. Pilot Recommendations

11.1. 3D rendering interoperability sprints

3D rendering is a core part of the 3D Information Management domain, and as such, dedicating effort to ensure high levels of interoperability among different vendors is worth the resources. The organization of regular 3D rendering interoperability sprints could be an effective way to spot these issues and reach this goal. Vendors' development teams should be motivated to participate as it is in their own interest to find issues in their products as early as possible and ensure high levels of interoperability.

11.2. 3D Tiles vs I3S benchmarks

If 3D Tiles and I3S are going to be treated as complementary formats (and not implicitly as competitors) there needs to exist documentation about which format is better for which use case. One way to generate this documentation is to organize benchmarks and measure different variables (such as performance, network use, encoding, styling, and so on, for different types of models such as buildings, vegetation, objects, and so on). These metrics would be an objective reference that could help better understand when to use which format.

These benchmarks could be extended with sprints or pilots to promote the development and popularization of transformation tools, such as I3S-to/from-3DTiles, that could help bridge the existing gap in interoperability, with currently few servers and clients supporting a wide range of standards.

11.3. Understanding the potential of 3D container hierarchy

3D container hierarchies offer considerable freedom to structure data content and to host that content. However, at this moment there are no guidelines or best practices as to how to implement such structures. A logical next step would be to start working on the creation of guidelines and best practices by requesting feedback from relevant players, or by organizing a pilot to compare different data structures, or by other similar means. One possible outcome could well be that the same content can be organized in different viable structures, depending on its usage and the property of the content.

Additionally, it could be very useful to explore the legal point of view of creating 3D container hierarchies by combining content served from different sources, such as guidelines on how to choose a suitable license for a content that could be used by a 3D Container federated catalog, or what uses could be allowed/forbidden to a 3D Container federated catalog.

11.4. Additional opportunities for future work

These opportunities for future work are drawn from the Summary ER:

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 (.3dtiles) 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).

In terms of standardization, the Pilot developed a draft API and resource model that have proven mature enough to be introduced into 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 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.

Chapter 12. Conclusions

The goal of the 3D Data Container and Tiles API Pilot was to develop and test an integrated suite of draft specifications for 3D geospatial resources compatible with existing OGC 3D data delivery standards in order to:

- Support smooth transitions between 2D and 3D environments;
- Allow applications working with 2D tile resources to get 3D resources (and vice-versa); and,
- Enable access by 3D bounding volume to multiple data models, datasets, and distributions.

To achieve these goals, the Pilot participants developed a draft GeoVolume resource (originally 3D Container resource) and GeoVolumes API providing browse and query access to 3D geospatial data for streamed data delivery by means of nested geospatial volume container resources. The 3D data resources supported by this API include feature geometries, feature attribute values, elevation models, texture data, and so on. The API provides both link-follow and bbox query methods for access to 2D and 3D resources in a manner independent of the underlying data store, supporting multiple standard geospatial distribution formats such as 3D Tiles, I3S, CDB, and CityGML.

A number of data server and client implementations were developed in the course of the Pilot in order to test interoperable data delivery via the draft API. The GeoVolumes API developed by the Pilot participants was defined using the OpenAPI 3.0 definition language and conforms to the draft OGC API - Common - Part 1: Core [http://docs.opengeospatial.org/DRAFTS/19-072.html] building blocks such as landing page, API definition, conformance, and collections information. The Pilot participants developed and tested the GeoVolumes API in order to advance open-standard 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.

The present GeoVolumes draft API specification includes the 6 basic and extended forms of functionality implemented and tested during the Pilot. The Pilot participants successfully implemented and demonstrated interoperability between components with this API. The experiences documented in this Engineering Report also suggest that further work is needed to extend the API for additional data types and nested volume schemes. Future work should also address the integration of the GeoVolumes API with other OGC API building blocks to provide seamless interaction from navigating volumes of interest to interacting with specific dataset distributions and elements.

The participants noted during the Pilot that by providing a high level division of space into regions of interest and connecting those to information about 3D datasets, the GeoVolumes API does embody some functions in common with catalog API’s. While not a general purpose catalog, the API can provide some capabilities for spatial indexing of metacontent which should be further explored and defined, specifically with regard to complementing the developing OGC API - Records specification.

Appendix A: UML Model

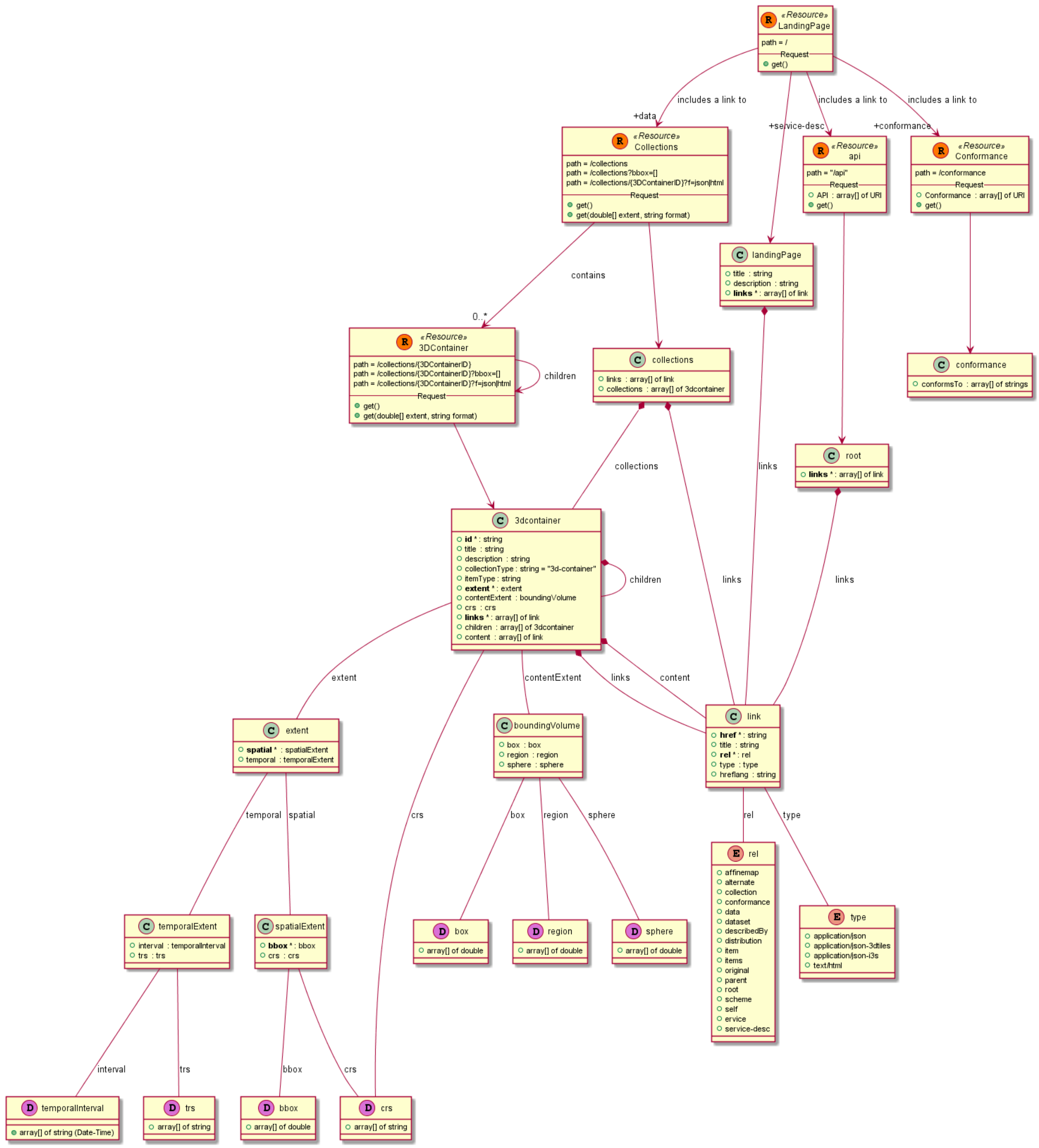


Figure 80. Geo-volume UML Model

Instructions and guidelines on the usage of UML models are provided in OGC document OGC-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867].

Appendix B: Comparison of 3D Tiles and I3S Community Standards

3D-Tiles	I3S
	Layer type: 3D Objects (e.g. building exteriors), Integrated Mesh (3D Surface mesh with texture), Points (eg. Point features symbolized by 3d symbols) and Point Cloud (part of OGC I3S 1.1 Community Standard - currently under final OGC TC approval process)
Tile format: Batched 3d model (.b3dm) - heterogenous e.g. textured terrain, 3d buildings, interiors etc.	Tile format: All I3S layer types support a batched geometry model persistency format (a binary typed array geometry buffer as defined in I3S standard) in support of 3D buildings, textured surfaces, point cloud, symbolized point features etc..
Tile format: Point cloud	Attribute support: I3S supports a compact per feature level attribution model
Tile format: Composite (cmpt) concatenate tiles of different formats	Symbolization: 3D model instancing is supported thru 3D symbolization in I3S 3D Point Layer type
Tile format: Instanced 3d models (i3dm) - 3d model instances e.g. trees. Embeds glTF	

Comparison of spatial reference considerations

I3S	3D Tiles
I3S allows usage of all spatial reference including custom Coordinate Reference Systems (CRS). Either OGC/ISO WKT1 or WKT2 encodings can be used in specifying the spatial reference object.	A tileset's global coordinate system will often be in a WGS84 earth centered, earth fixed (ECEF) reference frame (EPSG 4979) but it doesn't have to be.
I3S has a concept of Global and Local scenes. All I3S profiles support writing 3D content in two modes: global and local. In global mode, only the geographic CRS (WGS84, WKID 4326) is supported for both index and vertex positions. The coordinates are represented using longitude, latitude and elevation. In local mode, all other geodetic CRS, including projected coordinate systems, are allowed.	Coordinates are in WGS84 datum, and are specified in radians (Angles are in radians)

I3S	3D Tiles
Vertex positions are specified using a geodetic CRS, where x,y,z axes are all in the same unit and with a per-node offset. In Global Scenes X and Y Coordinate bounds of the layer and XY components of the vertex position are specified in decimal degrees. Elevation (the z component of the vertex position) is specified in meters. The Minimum Bounding Volume (MBV) radius unit (for MBS) or halfSize unit (for OBB) is specified in meters.	The region bounding volume specifies bounds using a geographic coordinate system (latitude, longitude, height) specifically using EPSG 4979
Axis order explicitly defined by the CRS shall be used when present. When the axis order is not defined by the CRS, Easting, Northing, Height axis order shall be used.	boundingVolume.region is an array of 6 numbers defining the geographic region, with the order [west,south,east,north,minimum height, maximum height]
All I3S layers indicate coordinate reference system via the spatialReference property in the 3dSceneLayerInfo resource	Right handed Cartesian coordinate system (cross product of x and y gives z)
	A tile transform may be applied to transform a tile's local coordinate system to the parent tile's coordinate system

Comparison of height handling

I3S	3D Tiles
The WKT string representation of the CRS now includes the vertical coordinate reference system utilized by the layer.	
Height and node minimum bounding spheres shall be specified in meters for Global scenes. For a local scenes, both height and bounding volumes are defined using the same xy units .	The unit for all linear distances is meters.
Vertical coordinate reference system may be either ellipsoidal (height defined to a reference ellipsoid) or gravity-related (reference geoid / gravity surface)	Heights are in metres above the WGS 84 ellipsoid
Height shall always point upwards towards the sky	Z axis as up for local Cartesian coordinate systems

I3S	3D Tiles
3dSceneLayerInfo resource also includes a coarse metadata property called heightModelInfo, which can be used by a client application to quickly identify if the layers' height model is either gravity-related or ellipsoidal	

Bounding volumes

I3S	3d Tiles
The I3S standard supports both an Oriented bounding box (OBB) or a Minimum Bounding Sphere (MBS) as a node's bounding volume.	Options include an oriented bounding box, a bounding sphere and a geographic region defined by minimum and maximum latitudes, longitudes and heights.
For Global Scenes: The location of all index-related data structures such as node bounding spheres SHALL be specified using a single, global geographic WGS 84 CRS. Coordinate bounds for such structures SHALL be in the range (-180.0000, -90.0000, 180.0000, 90.0000). Height and node minimum bounding sphere (MBS) radius SHALL be specified in meters	
Minimum bounding sphere is an array of four doubles, corresponding to x, y, z and radius of the minimum bounding sphere of a node.	Sphere is 4 numbers that define a sphere (x,y,z,radius)
When the nodes' Bounding volume is MBS it is defined by the center point of the minimum bounding sphere as well as the radius of the sphere. It is notated as an array of four doubles, corresponding to x, y, z and radius of the minimum bounding sphere of the node. For a global scene, i.e. ellipsoidal coordinate systems, the values of the array correspond to longitude in decimal degrees, latitude of in decimal degrees, elevation in meters and radius in meters. For all other CRS, the values of x,y,z and r are in the same unit as the xy units defined in the CRS of the layer.	The boundingVolume.box property is an array of 12 numbers that define an oriented bounding box in a right-handed 3-axis (x, y, z) Cartesian coordinate system where the z-axis is up. 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.

I3S	3d Tiles
When the bounding volume is Oriented Bounding Box (OBB) it is represented by a center position, an extent (halfSize) and a quaternion for orientation. The center represents the center point of the oriented bounding box. For a global scene it is specified as longitude, latitude in decimal degrees, and elevation (Z) in meters. Obb.center consists of 3 double arrays. In local scenes, it is specified using the units of the CRS. The half size "extent" values are measured from the center to the sides of the box. For a global scene, such as the XY coordinate system in WGS1984, the center is specified in latitude/longitude in decimal degrees, elevation (Z) in meters. The quaternion is used to encode orientation/rotation of the bounding box and is specified as 4 components. The quaternion components are in the order x, y, z, w.	The boundingVolume.region property is an array of six numbers that define the bounding geographic region with latitude, longitude, and height coordinates with the order [west, south, east, north, minimum height, maximum height]

Tree structure

I3S	3D Tiles
Spatial extent of the data is split into regions called nodes, with roughly equal amounts of data, and is organized into a hierarchical and navigable data structure. Each node has an address and can be thought of as equivalent to tiles.	Data is split into regions called nodes, with roughly equal amounts of data, and is organized into a hierarchical and navigable data structure.
Agnostic with respect to model used to index objects / features (supports Quadtrees, Octrees, as well as density dependent partitioning (e.g. R-Trees)	Agnostic with respect to model used to index objects (supports quadtrees, Octrees, K-d trees, grids etc.)
Nodes are organized in a bounding volume tree hierarchy (BVH)	Nodes are organized in a bounding volume tree hierarchy (BVH)
Each node has an ID — there are two types of Node ID formats supported — as string based treekeys, or as integers based on a fixed linearization of the nodes.	

I3S	3D Tiles
Treekeys are strings in which levels are separated by dashes: “3-1-0” has 3 numeric elements, hence the node is on level 4 (“root” node is at level 1) and the node “3-1” is its parent. The root node always gets ID “root”	
Tree starts with a root	Tree starts with a root

Example Esri implementation of I3S REST request:

https://tiles.arcgis.com/tiles/z2tnIkrLQ2BRzr6P/arcgis/rest/services/New_York_LoD2_3D_Buildings/SceneServer/layers/0/nodes/5-1-0-0-0/features/0

Possible similarities for API development - general resources:

- Some overarching service / collection information request;
- Tiling scheme;
- A link to their root node (tileset.json or layers/0/node);
- A section to describe a single node within a particular root node;
- A data resource;
- geometry?
- texture?
- attributes?

/collection/tiling-scheme/root/node/resource/

Appendix C: Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
May 1, 2020	T. Miller, G. Trenum	.1	all	initial version

Table 15. Revision History

Appendix D: Bibliography

[1] Maso Pau, J.: OGC Testbed-15: Maps and Tiles API Engineering Report. OGC 19-069,Open Geospatial Consortium, http://docs.opengeospatial.org/per/19-069.html (2019).

[2] Portele, C.: OGC Testbed-15: Styles API Engineering Report. OGC 19-010r2,Open Geospatial Consortium, http://docs.opengeospatial.org/per/19-010r2.html (2019).

[3] Meek, S., Brown, T., Portele, C.: OGC® Routing Pilot ER. OGC 19-041r3,Open Geospatial Consortium, http://docs.opengeospatial.org/per/19-041r3.html (2019).