A STAC Extension for discovering and cataloguing 3D city models

Hidemichi Baba[1,*], Hugo Ledoux[1]

[1] Delft University of Technology, the Netherlands—[h.b.baba, h.ledoux]@tudelft.nl

Keywords: 3D city models, CityJSON, CityGML, metadata, STAC, cloud-native geospatial

Abstract

National-scale 3D city model datasets are growing rapidly—Japan's PLATEAU project alone publishes over 170 000 files, and the Netherlands' 3D BAG covers all 10+ million buildings in the country—yet no standardised mechanism exists for discovering and cataloguing these datasets across repositories and data infrastructures. Essential properties such as coordinate reference systems, levels of detail, and city object types remain embedded inside data files in diverse formats (e.g. CityGML, CityJSON, or FlatCityBuf), invisible to search engines and catalogue services. This paper presents three contributions to address this gap. First, we define the STAC 3D City Models Extension, a formal extension to the *SpatioTemporal Asset Catalog* (STAC) specification that adds metadata fields for levels of detail, city object types, semantic surfaces, textures, materials, and attribute schemas. Second, we develop `city3dstac`, an open-source command-line tool written in Rust that automatically extracts metadata from 3D city models in different formats and generates standards-conformant STAC metadata. Third, we construct a prototype registry that currently contains 53 STAC Collections, of which 31 are fully indexed, totalling 14 766 STAC Items. The indexed Collections include the national-scale 3D BAG, American Cities, Estonia, and PLATEAU datasets. Existing STAC clients can be used for basic browsing of the catalogue without modification, while extension-aware clients can support richer 3D-specific filtering.

1. Introduction

Three-dimensional city models are widely used in urban planning, environmental simulation, emergency response, and noise modelling (Biljecki et al., 2015), and a growing number of cities and national mapping agencies worldwide now publish them as open data. These datasets are distributed in a growing variety of formats, e.g. Japan's PLATEAU[1] uses CityGML 2.0 (Gröger et al., 2012); the Netherlands' 3D BAG (Peters et al., 2022)[2] distributes CityJSON (Ledoux et al., 2019), which implements the CityGML 3.0 conceptual model (Kolbe et al., 2021). Newer encodings such as CityJSON Text Sequences (CityJSONSeq; used for streaming) (Ledoux et al., 2024) and FlatCityBuf (Baba et al., 2025) (for cloud-optimised binary access) are also starting to be used.

Despite this growth and despite the usefulness of 3D city models, there is no central registry for 3D city models, except for lists on the personal websites of some researchers. We believe that metadata can play a key role in solving this issue because, in general, they serve three broad purposes: (1) *documenting* a dataset's lineage and quality, (2) *cataloguing* datasets within a repository, and (3) enabling users to *discover* relevant datasets across repositories. Some formats embed metadata directly in the data itself: CityJSON includes a `metadata` object for fields such as point of contact and reference date, while CityGML has no dedicated metadata mechanism in the core standard, although Labetski et al. (2018) proposed a Metadata Application Domain Extension (ADE) that attaches rich metadata to CityGML files. However, metadata embedded inside data files cannot be queried without first downloading and parsing those files. In an era of national-scale datasets containing hundreds of thousands of files (which can be very large), the ability to discover and compare datasets *before* downloading them is essential. This paper therefore focuses on metadata for *cataloguing and discovery*: lightweight, standardised records that expose essential properties—coordinate reference systems, levels of detail (LoDs), city object types—in a searchable catalogue.

Our work acknowledges that geospatial data infrastructures are shifting towards cloud-native practices (Holmes, 2021). Within this landscape, the *SpatioTemporal Asset Catalog* (STAC) (Open Geospatial Consortium, 2025b) has emerged as a widely adopted metadata standard, particularly for earth observation data. Yet, while ISO 19115 (ISO, 2014) provides a comprehensive metadata framework for geospatial data in general, neither ISO 19115 nor STAC addresses the specific needs of 3D city models. This paper bridges that gap with the following three contributions:

1. We define the **STAC 3D City Models Extension**, a formal extension to the STAC specification that adds metadata fields for LoDs, city object types, semantic surfaces, textures, materials, and attribute schemas. As further described in Section 3, the extension is publicly available and already listed in the STAC Extensions registry.
2. We develop `city3dstac`, an **open-source command-line tool** written in Rust that automatically extracts metadata from CityJSON, CityGML, CityJSONSeq, and FlatCityBuf files and generates standards-conformant STAC Catalogues, Collections, and Items (these are defined in Section 2). We use it to index 31 of the 53 Collections currently present in the live registry, totalling 14 766 STAC Items. The indexed Collections include the national-scale 3D BAG, American Cities, Estonia, and PLATEAU datasets (Section 5).
3. We construct a **prototype registry**[3] of publicly available 3D city model datasets and demonstrate that existing STAC ecosystem tools—STAC Browser and STAC Map—can browse the catalogue without modification (Section 5).

* corresponding author

[1] `https://www.mlit.go.jp/en/toshi/daisei/plateau_en_2.html`

[2] `https://3dbag.nl/`

[3] `https://github.com/cityjson/city3d-stac-registry`

2. Related work

2.1 Metadata standards for geospatial data

At the international level, ISO 19115 (ISO, 2014) is the most widely adopted standard for geospatial metadata. It provides a comprehensive framework covering data discovery, fitness for use, data access, data transfer, and the use of digital data and services, classifying each element as mandatory, conditional, or optional (ISO, 2014). However, ISO 19115 presents two practical challenges. First, its XML-based encoding (ISO 19139 (ISO, 2019)) is verbose and difficult to integrate with modern web APIs and cloud-native workflows. Second, its scope is intentionally broad: it addresses every conceivable metadata purpose—from resource lineage to spatial resolution to distribution details—making full compliance burdensome for data providers who primarily need to make their datasets discoverable.

As geospatial data infrastructures shift towards cloud-native practices, JSON-based metadata standards have emerged as lighter-weight alternatives (Holmes, 2021). The *SpatioTemporal Asset Catalog* (STAC) (Open Geospatial Consortium, 2025b) is a community-driven specification that describes geospatial datasets as GeoJSON objects. It defines three core objects: a *Catalog* (a logical grouping), a *Collection* (Items sharing common properties, with spatial/temporal extent, licence, and summaries), and an *Item* (a GeoJSON Feature describing the spatial extent and linking to the actual data files). STAC supports two deployment modes (STAC Community, 2024b): a *static catalogue*, consisting of interlinked JSON files hosted on object storage or a web server with no server-side processing, and a *dynamic catalogue* (STAC API) exposing a `/search` endpoint for spatial and temporal queries. A key design principle is extensibility: developers define JSON schemas for additional fields and register them at the STAC Extensions registry[4]. This extensibility has encouraged a growing ecosystem of tools for creating, managing, and consuming STAC catalogues, including STAC Browser[5] and STAC Map[6]. Originally designed for satellite imagery, STAC has expanded to support point clouds, digital elevation models, and vector data. Prior to this work, no STAC extension existed specifically for 3D city models.

In parallel, the OGC has modernised its catalogue infrastructure: OGC API – Records (Open Geospatial Consortium, 2025a), approved in 2025, replaces the XML-based Catalogue Service for the Web (CSW) (Nogueras-Iso et al., 2005) with a RESTful, JSON/GeoJSON interface. The relationship between STAC and OGC API – Records is discussed in Section 6.1.

2.2 Metadata for 3D city models

The CityGML standard itself provides no dedicated metadata mechanism. To address this gap, Labetski et al. (2018) proposed a Metadata Application Domain Extension (ADE) for CityGML that organises metadata into three levels: city-model level (for data discovery), thematic-module level, and feature level. The city-model-level fields build on ISO 19115 (ISO, 2014) and extend it with properties specific to 3D city models, including LoDs, the presence of semantic surfaces, textures and materials, XLinks, external references, and thematic modules. The ADE is correspondingly comprehensive, inheriting much of ISO 19115's structure and obligation levels.

[4] `https://stac-extensions.github.io/`
[5] `https://radiantearth.github.io/stac-browser/`
[6] `https://developmentseed.org/stac-map/`

CityJSON (Ledoux et al., 2019) took a more pragmatic approach, incorporating only a subset of metadata fields from the ADE. These fields provide basic context but remain embedded inside the data file and are not exposed for external query. The same limitation applies to the Metadata ADE: both approaches attach metadata to the data itself, requiring users to download and parse files before assessing whether a dataset is relevant. A detailed comparison of ISO 19115 discovery metadata, the Metadata ADE, and the STAC extension proposed in this paper is provided in Table 1.

3. The STAC 3D City Models Extension

The STAC 3D City Models Extension is a formal extension to the STAC specification that adds metadata fields specific to 3D city model datasets. The extension schema is publicly available at `https://github.com/cityjson/stac-city3d` (version 0.2.0) and is listed in the STAC Extensions registry[7].

3.1 Design principles

The extension was designed around three core principles:

- **Minimal core.** Only metadata elements essential for *data discovery* are included. Richer descriptive and provenance metadata—such as those defined by ISO (2014) and Labetski et al. (2018)—are useful for archival purposes but too granular for catalogue-scale indexing.
- **Composability.** The extension is designed to complement, not replace, existing STAC extensions. Fields for CRS, file integrity, and statistics are delegated to the Projection, File, and Statistics extensions respectively, avoiding duplication.
- **Format neutrality.** The extension is not tied to a single encoding. Fields apply equally to CityJSON, CityGML, CityJSONSeq, and FlatCityBuf, with encoding information recorded separately via IANA media types on individual assets.

3.2 Field definitions

Table 1 positions each metadata element against two prior frameworks: the ISO 19115 geographic metadata standard (ISO, 2014) and the Metadata Application Domain Extension (ADE) for CityGML proposed by Labetski et al. (2018). For each concept, the table records the ISO 19115 obligation, whether the element is present in the Metadata ADE, and the corresponding STAC field or mechanism.

STAC core already covers the majority of ISO 19115 discovery elements. Our 3D City Models Extension adds fields specific to 3D city model content; elements such as XLinks, external references, and feature-level lineage are intentionally omitted as too format-specific or too granular for discovery (discussed in Section 6.2).

The 3D-specific fields are as follows:

city3d:lods Level of detail (LoD) values are stored as strings to support both integer LoDs (`"0"`–`"3"`) and the refined decimal scale of Biljecki et al. (2016) (e.g. `"1.2"`, `"2.2"`). Strings avoid floating-point precision issues where, for example, 1.1 may be stored as 1.09999, breaking equality checks—a lesson from CityJSON v1, which switched LoDs from numbers to strings for the same reason (Ledoux et al., 2019). In Collections, the field is a sorted list of all LoDs across member Items.

[7] `https://stac-extensions.github.io/`

https://doi.org/10.5194/isprs-annals-XII-4-W1-2026-25-2026

Table 1. Mapping of metadata elements across ISO 19115 (ISO, 2014), the CityGML Metadata ADE (Labetski et al., 2018), and the STAC 3D City Models Extension. ISO obligation: M = Mandatory, O = Optional, C = Conditional.

Element	ISO	ADE	STAC	Notes
ISO 19115 Discovery Elements				
Metadata identifier	O	Yes	STAC core (`id`)	Unique metadata identifier
Resource title	M	Yes	STAC core (`title`)	
Resource date	O	Yes	STAC core (`datetime` for Items; `extent.temporal` for Collections)	Reuse CityJSON `referenceDate` if present
Resource identifier	O	Yes	STAC core (`id`)	Reuse CityJSON `identifier` if present
Point of contact	O	Yes	STAC core (`providers`)	Reuse CityJSON `pointOfContact` if present
Geographic location	C	Yes	STAC core (`bbox`, `geometry`)	2D and 3D bbox; always in WGS 84
CRS / Projection	O	Yes	Projection Ext. (`proj:code`)	ISO 19115 `referenceSystemInfo`; via companion extension
Resource language	C	Yes	Not included	Available via separate Language Extension
Resource abstract	M	Yes	STAC core (`description`)	
Temporal/vertical extent	O	Yes	STAC core (`extent`)	
Online link	O	Yes	STAC core (`assets`, `links`)	
Keywords	O	Yes	STAC core (`keywords`)	
Access constraints	O	Yes	STAC core (`license`)	
Metadata date	M	Yes	STAC core (`created`, `updated`)	
Metadata point of contact	M	Yes	STAC core (`providers`)	Covered at Collection level
Spatial resolution	O	No	Not applicable	Relevant for raster data only
3D-Specific Elements (Labetski et al., 2018)				
Levels of detail	—	Yes	`city3d:lods`	Decimal LoDs per Biljecki et al. (2016)
Semantic surfaces	—	Yes	`city3d:semantic_surfaces`	Boolean flag
Textures	—	Yes	`city3d:textures`	Boolean flag
Materials	—	Yes	`city3d:materials`	Boolean flag
Thematic modules	—	Yes	`city3d:co_types`	ADE additionally records per-module counts
City object count	—	Yes	`city3d:city_objects`	Item: integer; Collection: {`min, max, total`}
XLinks	—	Yes	Intentionally omitted	CityGML-XML specific; violates format neutrality
External references	—	Yes	Intentionally omitted	Too format-specific for discovery
ADEs / Extensions	—	Yes	Intentionally omitted	Open for future inclusion
Format-Specific Elements				
Encoding format	—	—	Asset `type` (IANA media type)	CityJSON, CityGML, FlatCityBuf, etc.
Format version	—	—	`city3d:version`	Encoding version, e.g. CityJSON 2.0 or CityGML 2.0/3.0
Attribute schema	—	—	`city3d:attributes`	Name, type, description per city object type

city3d:co_types The type names follow the CityGML data model (e.g. `Building`, `Bridge`, `Vegetation`). Application Domain Extension types are distinguished by a `+` prefix (e.g. `+Noise`). This field is functionally equivalent to the thematic-module list of the Metadata ADE (Labetski et al., 2018), though the ADE additionally records per-module city object counts, which the STAC extension does not.

city3d:city_objects In Items, this is a single integer count. In Collection `summaries`, it is an object with three aggregate statistics: `min`, `max`, and `total` across all member Items.

city3d:version `city3d:version` records the version of the 3D city model encoding used by the asset, for example CityJSON 2.0 or CityGML 2.0/3.0. At Item level it is a string; at Collection level it is summarised as the set of versions present across Items.

city3d:attributes Each entry describes a semantic attribute present on city objects: `name` and `type` are required; `description` and `required` are optional. Supported types are `String`, `Number`, `Boolean`, `Date`, `Array`, and `Object`.

city3d:semantic_surfaces CityJSON and CityGML allow surfaces to have semantic labels (e.g. `RoofSurface` or `WallSurface`); this Boolean flag allows practitioners to assess the geometric-semantic richness of a dataset without parsing the full file.

city3d:textures, city3d:materials Two other Boolean flags are used to indicate whether texture coordinates and material definitions are present on city objects.

3.3 Companion extensions

The 3D City Models Extension is designed to complement a set of existing STAC Extensions. Although not enforced by the schema, the following are highly recommended for complete metadata:

- **Projection Extension**[8] (`proj:*`) — declares the native CRS of each dataset (e.g. `proj:code` for EPSG codes).
- **File Extension**[9] (`file:*`) — provides asset-level checksums, file sizes, and content statistics.
- **Statistics Extension**[10] — exposes summary statistics over numeric properties in Collection `summaries`.

For example, a STAC Item for a 3D BAG tile carries `proj:code: "EPSG:7415"` from the Projection Extension alongside `city3d:lods: ["1.2", "2.2"]`, allowing a client to filter datasets by both CRS and level of detail. Similarly, the File Extension can record asset checksums and file sizes, enabling integrity verification before downloading a dataset.

3.4 Worked examples

Most 3D city model datasets are tiled: a single city or country is split into spatial tiles, each stored as a separate file containing a few hundred to a few thousand buildings. In STAC terms, each tile maps to one *Item*, and the full dataset maps to one *Collection*.

[8] `https://github.com/stac-extensions/projection`
[9] `https://github.com/stac-extensions/file`
[10] `https://github.com/stac-extensions/stats`

A city that distributes its model as a single file is still represented as a Collection with one Item.

Figure 1a shows an excerpt of a STAC Collection for the 3D BAG dataset, highlighting the `city3d` extension fields in the `summaries` object. Figure 1b shows a corresponding STAC Item for one tile, where the `city3d` fields appear directly in `properties`. Note that `city3d:city_objects` is a single integer in an Item but an aggregate {`min, max, total`} in a Collection, and that the Item carries a 3D bounding box (six values) in its `bbox` field.

4. The city3dstac tool

`city3dstac` is an open-source command-line tool, written in Rust, that automatically generates STAC metadata from 3D city model files. The tool is published on `crates.io`[11] and its source code is available on GitHub[12].

4.1 Supported formats

At the moment, the tool supports four 3D city model formats: CityJSON (`.json`), CityJSONSeq (`.jsonl`), CityGML (`.gml`, `.xml`) in versions 2.0 and 3.0, and FlatCityBuf (`.fcb`). For each format, it extracts all fields defined by the 3D City Models Extension (Section 3) and produces standards-conformant STAC Items, Collections, and Catalogues. The tool supports both local files and remote URLs, depending on the workflow: `item` can read a single local or remote file directly, `collection` can generate a Collection from either a local directory or a list of remote URLs, and `catalog` aggregates multiple Collections.

4.2 CLI commands and workflows

The tool exposes four commands.

`item` Generates a single STAC Item from one input file, either local or remote. When given a remote URL, the tool fetches the file directly and uses the URL as the asset `href`; for local files, the path is used as a relative reference.

`collection` Generates STAC Items from input files and aggregates them into a STAC Collection. It can operate either on a local directory, which is scanned recursively, or on a given list of remote URLs. The command also takes Collection-level metadata such as the title, description, licence, and provider information. In practice, each dataset is described by a simple YAML configuration file that specifies the data locations together with these descriptive fields. Users do not need to download remote datasets in advance: the tool can read the files directly and extract the metadata needed to generate the Items and the resulting Collection. The output follows the standard STAC layout: one `collection.json` alongside an `items/` sub-directory.

`catalog` Aggregates multiple existing Collections into a single STAC Catalogue. This enables federated cataloguing of datasets from different cities or providers while keeping each dataset as a separate Collection. If every data provider published standardised metadata (e.g. ISO 19115 records), the Collection configurations could be generated automatically; in practice, a human must read each portal and create them.

`update-collection` Reads a set of existing STAC Item files and derives a Collection by aggregating their metadata. This command targets object-storage workflows in which Items are generated independently and later merged. Aggregation follows explicit rules: LoDs and city object types are merged and sorted; `city3d:city_objects` is summarised as {min, max, total}; Boolean flags are summarised as the set of values observed across Items, as required by STAC Collection summaries; and bounding boxes are unioned.

4.3 STAC GeoParquet output

In addition to JSON-based STAC output, the tool optionally generates a STAC GeoParquet file (`items.parquet`) alongside each Collection. STAC GeoParquet (Radiant Earth Foundation, 2026) is a community specification, currently at draft stage, that stores STAC Items in columnar Apache Parquet format rather than as individual JSON files. This output proved essential at scale: the 3D BAG collection alone comprises 8 941 STAC Items, and loading that many individual JSON files makes STAC Browser slow to render. A single GeoParquet file enables bulk access to all Item metadata in a compact, indexed format that libraries such as `GeoPandas`[13] and `DuckDB`[14] can query efficiently. The `--geoparquet` flag activates this output for the `collection`, `update-collection`, and `catalog` commands.

5. The 3D City Model STAC Registry

A growing number of cities and national mapping agencies publish 3D city models as open data, yet each portal uses its own metadata scheme and no interoperable discovery mechanism exists to search across them. We have built the 3D City Model STAC Registry[15] to address this gap by providing a centralised, machine-readable catalogue of publicly available 3D city model datasets worldwide. Using the STAC extension (Section 3) and `city3dstac` (Section 4), metadata from individual open data portals are extracted and aggregated into a single STAC Catalogue. Each data provider's datasets are grouped under a dedicated Collection while remaining queryable through a unified interface. Our registry is maintained as a public GitHub repository, and anyone can propose a new dataset by opening a pull request that adds the corresponding metadata configuration. Once a dataset is registered, an automated workflow generates the STAC Catalogue, Collections, and Items, and publishes the resulting STAC assets for public access.

The generated STAC files are hosted in a single object storage location rather than alongside the original datasets.

5.1 Indexed datasets

At the time of writing, the live registry contains 53 STAC Collections. Four national-scale datasets and 27 subnational, city, or experimental datasets have been fully indexed—31 Collections in total—amounting to 14 766 STAC Items. A further 22 Collections are catalogued but not yet indexed: their source files are exposed only through interactive download portals rather than stable, directly-crawlable URLs, so no Items have been generated for them yet. PLATEAU is among the indexed national-scale datasets; indexing is complete, aggregating 170 413 source CityGML files into 306 city-level Items at the time of writing. Table 2 summarises all Collections currently present in the registry, including those not yet indexed.

[11] `https://crates.io/crates/city3d_stac`
[12] `https://github.com/cityjson/city3d-stac-tool`
[13] `https://geopandas.org/`
[14] `https://duckdb.org/`
[15] `https://github.com/cityjson/city3d-stac-registry`

```json
{
  "type": "Collection",
  "stac_version": "1.1.0",
  "stac_extensions": [
    "https://cityjson.github.io/stac-city3d/v0.2.0/schema.json",
    "https://stac-extensions.github.io/projection/v2.0.0/schema.json",
    ...
  ],
  "id": "netherlands-3d-bag",
  "title": "Netherlands 3D BAG",
  "description": "All 10 million+ buildings in the Netherlands with LoD1 and LoD2 representations.",
  "keywords": ["3d city model", ...],
  "license": "CC-BY-4.0",
  "providers": [
    {
      "name": "3D Geoinformation Group (TU Delft)",
      "description": "3D Geoinformation Research Group at TU Delft",
      "roles": ["processor", "host"],
      "url": "https://3d.bk.tudelft.nl/"
    },
    {
      "name": "3DGI",
      "description": "A software development and consulting company with a focus on 3D geoinformation and open source technologies",
      "roles": ["processor", "host"],
      "url": "https://3dgi.xyz/"
    },
    ...
  ],
  "extent": {
    "spatial": {
      "bbox": [[3.3577097845746073, 50.75159395422835, -2147483.75, 7.226748690270318, 53.498808967181894, 345.61053466796875]]
    },
    "temporal": {"interval": [["2026-05-05T08:30:58.629276899Z", null]]}
  },
  "summaries": {
    "city3d:lods": ["0", "1.2", "1.3", "2.2"],
    "city3d:version": ["2.0"],
    "city3d:materials": [false],
    "city3d:city_objects": {
      "min": 2,
      "max": 10370,
      "total": 21555522
    },
    "city3d:textures": [false],
    "city3d:semantic_surfaces": [true],
    "city3d:co_types": ["Building", "BuildingPart"],
    "proj:code": ["EPSG:7415"]
  },
  "links": [
    {
      "href": "./items/8-760-72.city.json",
      "rel": "item",
      "type": "application/json",
      "title": "8-760-72.city.json"
    },
    ...
  ]
}
```


(a) STAC Collection for the 3D BAG dataset, with extension fields in `summaries`.


```json
{
  "type": "Feature",
  "stac_version": "1.1.0",
  "stac_extensions": [
    "https://cityjson.github.io/stac-city3d/v0.2.0/schema.json",
    "https://stac-extensions.github.io/projection/v2.0.0/schema.json"
  ],
  "id": "8-760-72.city.json",
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [
        [6.0791248228374, 50.9115960199424],
        [6.092980641781406, 50.9115960199424],
        [6.092980641781406, 50.919816409052984],
        [6.0791248228374, 50.919816409052984],
        [6.0791248228374, 50.9115960199424]
      ]
    ]
  },
  "bbox": [
    6.0791248228374, 50.9115960199424, 0.000499725341796875, 6.092980641781406,
    50.919816409052984, 124.6405029296875
  ],
  "properties": {
    "datetime": null,
    "city3d:version": "2.0",
    "city3d:city_objects": 882,
    "city3d:lods": ["0", "1.2", "1.3", "2.2"],
    "city3d:co_types": ["Building", "BuildingPart"],
    "city3d:attributes": [
      {
        "name": "b3_bag_bag_overlap",
        "type": "Number"
      },
      {
        "name": "b3_bouwlagen",
        "type": "Number"
      }
    ],
    "city3d:semantic_surfaces": true,
    "city3d:textures": false,
    "city3d:materials": false,
    "proj:code": "EPSG:7415"
  },
  "links": [
    {
      "href": "https://data.3dbag.nl/v20250903/tiles/8/760/72/8-760-72.city.json.gz",
      "rel": "city-model",
      "type": "application/city+json"
    },
    {
      "href": "../collection.json",
      "rel": "collection",
      "type": "application/json"
    }
  ],
  "assets": {
    "data": {
      "href": "https://data.3dbag.nl/v20250903/tiles/8/760/72/8-760-72.city.json.gz",
      "title": "3D city model data",
      "type": "application/city+json",
      "roles": ["data"]
    }
  },
  "collection": "netherlands-3d-bag"
}
```


(b) STAC Item for a single 3D BAG tile, with `city3d` fields in `properties` and a six-value 3D `bbox`.

Figure 1. Excerpts of `city3d`-enabled STAC metadata generated for the 3D BAG dataset: (a) a Collection and (b) an Item for a single tile.

Table 2. Overview of 3D city model datasets in the STAC registry workflow. Each row represents one STAC Collection. Items: number of STAC Items per Collection; one Item is one source file, except for PLATEAU, where each Item aggregates one city's files. Rows marked *not indexed* are catalogued but have no Items yet.

Collection	Country	Format	LoDs	Items	Licence
National-scale datasets					
3D BAG	Netherlands	CityJSON	0, 1.2, 1.3, 2.2	8 941	CC-BY-4.0
American Cities	USA	CityGML	1	5 229	ODbL-1.0
Estonia	Estonia	CityGML	1, 2	5	CC-BY-4.0
PLATEAU	Japan	CityGML	0, 1, 2, 3, 4[†]	306	CC-BY-4.0
Subnational, city, and experimental datasets					
Ingolstadt	Germany	CityGML	0, 1, 2, 3	3	Other
Linz	Austria	CityGML	3	156	CC-BY-4.0
Luxembourg (pilot)	Luxembourg	CityGML	2	1	CC0-1.0
Montréal	Canada	CityJSON	2	77	CC-BY-4.0
New York (OTI)	USA	CityJSON	2	1	Other
Rotterdam	Netherlands	CityJSON	2	1	Other
Singapore (HDB)	Singapore	CityJSON	1	1	MIT
The Hague	Netherlands	CityJSON	2	1	CC-BY-4.0
Zürich	Switzerland	CityJSON	2	1	CC0-1.0
Brandenburg	Germany	CityGML	2	1	dl-de/by-2-0
Bremen	Germany	CityGML	2	1	CC-BY-4.0
Hannover	Germany	CityGML	2	1	CC-BY-4.0
Kuopio	Finland	CityGML	2	1	CC-BY-4.0
Riga	Latvia	CityGML	4	1	CC-BY-4.0
Auvergne-Rhône-Alpes (11 sectors)	France	CityGML	2	1	etalab-2.0
Auvergne-Rhône-Alpes (ZAE)	France	CityGML	3	21	etalab-2.0
Châtel-Guyon & Riom	France	CityGML	2	6	etalab-2.0
Clermont-Ferrand	France	CityGML	2	1	etalab-2.0
Isère	France	CityGML	2	1	etalab-2.0
Leipzig	Germany	CityGML	2	1	dl-de/by-2-0
Niedersachsen	Germany	CityGML	2	1	Various
North Rhine-Westphalia	Germany	CityGML	2	1	dl-de/zero-2-0
Rheinland-Pfalz	Germany	CityGML	2	1	dl-de/by-2-0
Sachsen-Anhalt	Germany	CityGML	2	1	dl-de/by-2-0
Vichy (UNESCO)	France	CityGML	3	1	etalab-2.0
Mecklenburg-Vorpommern	Germany	CityGML	2	1	CC-BY-4.0
Vienna	Austria	CityJSON	2	1	CC-BY-4.0
Catalogued, not yet indexed					
Baden-Württemberg	Germany	CityGML	–	*not indexed*	dl-de/by-2-0
Bavaria	Germany	CityGML	–	*not indexed*	CC-BY-4.0
Berlin	Germany	CityGML	–	*not indexed*	dl-de/zero-2-0
Bordeaux	France	Other (3DS)	–	*not indexed*	etalab-2.0
Brussels	Belgium	CityGML	–	*not indexed*	CC0-1.0
Dresden	Germany	CityGML	–	*not indexed*	dl-de/by-2-0
Espoo	Finland	CityGML	–	*not indexed*	CC-BY-4.0
Freiburg	Germany	CityGML	–	*not indexed*	dl-de/by-2-0
Hamburg	Germany	CityGML	–	*not indexed*	dl-de/by-2-0
Helsinki	Finland	CityGML	–	*not indexed*	CC-BY-4.0
Hessen	Germany	CityGML	–	*not indexed*	dl-de/zero-2-0
Lyon	France	CityGML	–	*not indexed*	etalab-2.0
Melbourne	Australia	Other (OBJ/SLPK)	–	*not indexed*	CC-BY-4.0
New York (TUM)	USA	CityGML	–	*not indexed*	Other
Potsdam	Germany	CityGML	–	*not indexed*	dl-de/by-2-0
Poznań	Poland	CityJSON	–	*not indexed*	Proprietary
Prague	Czechia	CityGML	–	*not indexed*	CC-BY-4.0
Saxony	Germany	CityGML	–	*not indexed*	dl-de/by-2-0
Schleswig-Holstein	Germany	CityGML	–	*not indexed*	CC-BY-4.0
Toronto	Canada	Other (Shapefile/GDB)	–	*not indexed*	OGL-Canada-2.0
Vantaa	Finland	CityGML	–	*not indexed*	CC-BY-4.0
Wellington	New Zealand	Other (GDB/DWG/SKP)	–	*not indexed*	CC-BY-4.0

[†] LoD 4 was defined in CityGML 2.0 for interior models but removed in CityGML 3.0.

https://doi.org/10.5194/isprs-annals-XII-4-W1-2026-25-2026

We have manually browsed the websites of several data providers to gather the datasets for the registry. Those were obtained through mostly the lists at `https://3d.bk.tudelft.nl/opendata/` and `https://github.com/OloOcki/awesome-citygml`, and we have searched for others. To build the registry, we filled out manually the metadata fields and the direct URL of each of the datasets of a city, for instance. Based on those files, we have converted them into STAC Items and Collections using our tool described in Section 4.

It should be noted that some cities/regions expose their 3D models via a web service and/or a dedicated graphical interface that allows users to draw a region and receive a link through an email (in most cases, their datasets are stored in a database on the server). For these cases, we have included the web service URL in the STAC Collection, as it is currently impossible to obtain a file that can be parsed. We are looking into ways to bypass this limitation.

5.2 Interoperability evaluation

To evaluate whether the generated catalogue integrates with the existing STAC ecosystem, we tested the registry with two clients. STAC Browser[16] (Radiant Earth) renders the catalogue as a browsable web interface, displaying Collection-level summaries and Item-level metadata without any modification (Figure 2a). This confirms that basic browsing of 3D city model metadata requires no modification to existing STAC clients, while richer 3D-specific filtering can be enabled by clients that understand the extension fields.

To demonstrate richer interaction, we also deployed a customised STAC Map instance[17] and extended it with filtering controls specific to 3D city models (Figure 2b). Users can filter Items by LoD, city object type, and the presence of semantic surfaces, textures, and materials—properties that would not be queryable without the extension fields defined in Section 3.

6. Discussion and future work

6.1 STAC and OGC API – Records

STAC and OGC API – Records (Open Geospatial Consortium, 2025a) address similar problems and share substantial architectural overlap. Both build on OGC API – Features and expose metadata as GeoJSON resources through RESTful interfaces. An OGC API – Records *record* is conceptually similar to a STAC Collection, and the OGC API – Features query endpoint parallels the STAC API `/search` endpoint (Holmes, 2021). In practice, the two standards are largely interoperable: implementations such as `pycsw`[18] already support both protocols from a single deployment.

A key difference lies in their deployment assumptions. OGC API – Records requires an HTTP-based API that generates responses dynamically, which presupposes server-side infrastructure—a database, an application server, and operational maintenance. This infrastructure must be maintained, and the catalogue becomes unavailable when the server is down. STAC, by contrast, defines a *static catalogue* mode in which the entire catalogue is a set of interlinked JSON files hosted on object storage such as Amazon S3 or Google Cloud Storage (STAC Community, 2024b). Static catalogues need no server-side infrastructure and scale with the underlying object storage: our 14 766-Item registry is served entirely from a single bucket. STAC also notes that a dynamic catalogue may keep a cached static catalogue as a backup of its fields (STAC Community, 2024a). This static mode has no equivalent in the OGC standards suite (Holmes, 2021).

For 3D city model metadata, this distinction is particularly relevant. Many open data initiatives are maintained by small teams with limited infrastructure budgets. A static STAC catalogue can be generated once by `city3dstac`, hosted on any object storage or static web server, and consumed by any STAC-compatible client—without requiring the initiative to operate a catalogue API. Where a provider does operate server-side infrastructure, supporting both OGC API – Records and the STAC API simultaneously is straightforward given their architectural similarity, and the static STAC catalogue can serve as a low-cost redundancy layer.

6.2 Omitted metadata elements

The STAC 3D City Models Extension intentionally omits several metadata elements present in ISO 19115 (ISO, 2014) or the Metadata ADE of Labetski et al. (2018). These omissions reflect the extension's focus on data discovery rather than archival completeness, and each warrants justification.

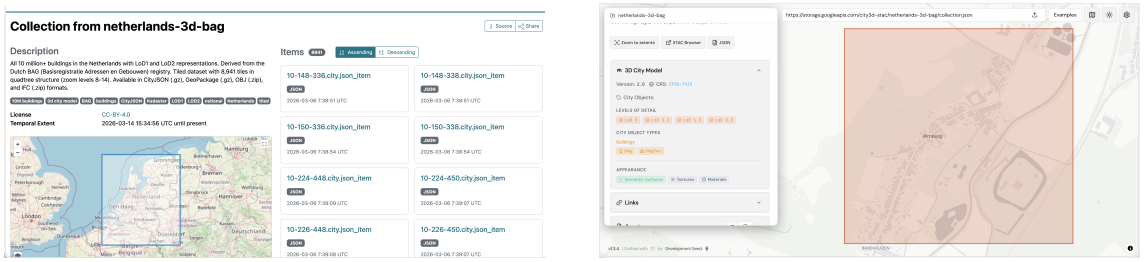
Resource lineage. ISO 19115 defines lineage as the processing history that produced a dataset. For 3D city models this is particularly complex: datasets may derive from satellite imagery, airborne LiDAR, street-level imagery, building information models, cadastral footprints, or combinations thereof, and the appropriate granularity varies by use case—an urban planner may need only the data source, whereas a researcher assessing geometric accuracy needs the full pipeline. Encoding lineage for all these cases would require a schema far richer than discovery needs, so we omit it and consider it better addressed by dataset-level documentation or provenance-specific standards.

XLinks, ADEs, and feature-level lineage. The Metadata ADE records XLinks (shared geometry references between CityGML objects), Application Domain Extensions, and feature-level provenance. Geometry-reuse mechanisms exist across formats—XLinks in CityGML, geometry templates in CityJSON—but their presence does not affect a user's motivation to download a dataset, so it is not relevant for discovery; likewise, a user searching for building models is unlikely to filter by ADE, though ADEs remain open for future inclusion. Feature-level lineage is rarely populated in practice, as most datasets are produced at city or national scale with uniform processing, and would impose significant overhead for minimal discovery benefit.

6.3 Future work

We plan to incorporate the temporal dimension, letting users search across the multiple versions that datasets such as 3D BAG provide, whereas we currently index only the latest version of each. We also plan to support additional formats such as IFC, Rhino, and 3D Tiles; although not strictly semantic 3D formats, they are the only ones available in some regions. We aim to grow the GitHub registry into a reference discovery point for 3D city models worldwide, curating further datasets over time. As STAC and the proposed extension become more widely

[16] `https://radiantearth.github.io/stac-browser/`
[17] `https://cityjson.github.io/city3d-stac-map`
[18] `https://pycsw.org/`



(a) STAC Browser rendering the 3D city model catalogue. The interface displays Collection-level summaries and Item-level metadata.

(b) Forked STAC Map displaying spatial footprints of 3D BAG Items over the Netherlands. Each rectangle corresponds to a STAC Item; the left panel shows filtering controls for LoD, city object types, and appearance properties.

Figure 2. Ecosystem interoperability: (a) unmodified STAC Browser renders the catalogue directly; (b) a forked STAC Map adds 3D-specific filtering on extension fields.

used, we expect this workflow to shift from manual curation towards provider-side publication, where providers publish STAC metadata alongside their data so that registries can harvest and aggregate distributed Collections rather than maintain a separately curated index.

Acknowledgements

This research was carried out within the framework of the MultiRoofs project (`https://multiroofs.nweurope.eu/`), co-funded by the European Union through the Interreg North-West Europe Programme. We also thank Chaeyeon Moon, MSc student at TU Delft Geomatics, for helping to curate the list of open data cities.

References

Baba, H., Ledoux, H., Peters, R., 2025. FlatCityBuf: A new cloud-optimised CityJSON format. *Proceedings 20th 3D GeoInfo Conference*, XLVIII-4/W15-2025, ISPRS, Tokyo, Japan, 17–24.

Biljecki, F., Ledoux, H., Stoter, J., 2016. An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59, 25–37.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., Çöltekin, A., 2015. Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, 4(4), 2842–2889.

Gröger, G., Kolbe, T. H., Nagel, C., Häfele, K.-H., 2012. OGC City Geography Markup Language (CityGML) Encoding Standard. `http://www.opengis.net/spec/citygml/2.0`. OGC Encoding Standard, document 12-019, version 2.0.0.

Holmes, C., 2021. SpatioTemporal Asset Catalogs and the Open Geospatial Consortium. `https://medium.com/radiant-earth-insights/spatiotemporal-asset-catalogs-and-the-open-geospatial-consortium-659538dce5c7`. Radiant Earth Insights blog post, 19 January 2021.

ISO, 2014. Geographic information — Metadata — Part 1: Fundamentals (ISO 19115-1:2014). `https://www.iso.org/standard/53798.html`.

ISO, 2019. Geographic information — XML schema implementation — Part 1: Encoding rules (ISO/TS 19139-1:2019). `https://www.iso.org/standard/67253.html`.

Kolbe, T. H., Kutzner, T., Smyth, C. S., Nagel, C., Roensdorf, C., Heazel, C., 2021. OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard. `https://docs.ogc.org/is/20-010/20-010.html`. OGC Standard, document 20-010.

Labetski, A., Kumar, K., Ledoux, H., Stoter, J., 2018. A metadata ADE for CityGML. *Open Geospatial Data Software and Standards*, 3(1).

Ledoux, H., Ohori, K. A., Kumar, K., Dukai, B., Labetski, A., Vitalis, S., 2019. CityJSON: a compact and easy-to-use encoding of the CityGML data model. *Open Geospatial Data, Software and Standards*, 4(1).

Ledoux, H., Stavropoulou, G., Dukai, B., 2024. Streaming CityJSON datasets. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVIII-4/W11-2024, 57–63. http://dx.doi.org/10.5194/isprs-archives-XLVIII-4-W11-2024-57-2024.

Nogueras-Iso, J., Zarazaga-Soria, F. J., Béjar, R., Álvarez, P. J., Muro-Medrano, P. R., 2005. OGC Catalog Services: a key element for the development of Spatial Data Infrastructures. *Computers & Geosciences*, 31(2), 199–209.

Open Geospatial Consortium, 2025a. OGC API — records — part 1: Core. `https://ogcapi.ogc.org/records/`. OGC Standard, approved April 2025.

Open Geospatial Consortium, 2025b. SpatioTemporal Asset Catalog (STAC). `https://www.ogc.org/standards/stac/`. OGC Community Standard, document 25-004.

Peters, R., Dukai, B., Vitalis, S., van Liempt, J., Stoter, J., 2022. Automated 3D reconstruction of LoD2 and LoD1 models for all 10 million buildings of the Netherlands. *Photogrammetric Engineering and Remote Sensing*, 88(3), 165–170.

Radiant Earth Foundation, 2026. STAC GeoParquet specification. `https://radiantearth.github.io/stac-geoparquet-spec/0.7.0/`. Draft specification, version 0.7.0, for storing STAC Items in Apache Parquet format.

STAC Community, 2024a. STAC best practices. `https://github.com/radiantearth/stac-spec/blob/master/best-practices.md`. STAC Specification v1.1.0.

STAC Community, 2024b. The STAC Specification. `https://stacspec.org/en/about/stac-spec/`. Version 1.1.0.