**Community Standard Justification: 20-005r1**

**TITLE: CityJSON v1.0**

**CONTRIBUTOR**

**Name: Linda van den Brink**

**Organization: Geonovum**

**Address: Barchman Wuytierslaan 10, 3818LH Amersfoort, the Netherlands**

**Phone: +31 334604100**

**Email: l.vandenbrink@geonovum.nl**

**DATE: January 15th, 2020**

# Introduction

This document provides a justification to the OGC Technical Committee (TC) for consideration of CityJSON (v1.0) as a Community standard. This justification, along with the submitted candidate Community standard, will form the basis for TC review and vote to approve the start of a Work Item as the first step in the Community standard process for this standard.

The submitters agree to abide by the TC Policies and Procedures and OGC Intellectual Property Rights Policy (<http://www.opengeospatial.org/ogc/policies)> during the processing of this submission.

Once approved, the Community standard Work Item defined by this document is valid for six (6) months.

# Overview of proposed submission

*Summarize the proposed Community standard. In this summary, provide an overview of the geospatial interoperability requirements the proposed standard supports, the history of its development and use, and use cases.*

*(note: the text here has generally been taken from the official website https://cityjson.org)*

**Description**

CityJSON is a JSON-based encoding for a subset of the OGC CityGML data model (version 2.0.0), which is an open standardized data model and exchange format to store digital 3D models of cities and landscapes.

CityJSON defines ways to describe most of the common 3D features and objects found in cities (such as buildings, roads, rivers, bridges, vegetation and city furniture) and the relationships between them. It also defines different standard levels of detail (LoDs) for the 3D objects, which allows us to represent different resolutions of objects for different applications and purposes.

A CityJSON file describes both the geometry and the semantics of the city features of a given area, e.g., buildings, roads, rivers, trees, and the city furniture. A CityJSON object, representing a city, is as ‘flat’ as possible, i.e., the hierarchy of CityGML has been flattened out and only the city objects which are ‘leaves’ of this hierarchy are implemented. This considerably simplifies the storage of a city model, and furthermore does not mean that information is lost.

The developers of CityJSON consider CityJSON to be an implementation of the CityGML data model.

**CityGML compatibility**

CityJSON implements most of the data model of CityGML v2.0.0, and all the CityGML modules have been mapped to CityJSON objects. However, for the sake of simplicity and efficiency, some features have been omitted and/or simplified. The few features that are not supported are documented at: <https://www.cityjson.org/citygml-compatibility/>.

Some features that are wished by practitioners, but that are not in CityGML (e.g., metadata support), are built-in CityJSON. Another advantage is that Extensions to the core model can be created easily and quickly, and that these do not require software used in the downstream to be modified, that is files containing Extensions can be processed as “normal” CityJSON files.

**Advantages of CityJSON**

The aim of CityJSON is to offer an alternative to the GML encoding of CityGML, which can be verbose and therefore complex to work with). CityJSON aims at being easy-to-use, both for reading datasets, and for creating them. It was designed with programmers in mind, so that tools and APIs supporting it can be quickly built. It was also designed to be compact (it typically compresses publicly available CityGML files by 6x), and to be friendly for web and mobile development.

A CityJSON file is on average about a factor 6 more compact than its CityGML equivalent, see https://github.com/tudelft3d/cityjson/wiki/Compression-factor-for-a-few-open-CityGML-datasets for some real-world examples.

**Applications and use cases**

All the applications and use cases that are theoretically possible with CityGML also apply to CityJSON. A list of these applications and use cases is found on the website: https://www.cityjson.org/applications/.

Notice that since JSON is used for the encoding, the CityJSON developers believe that CityJSON actually expands the possibilities of CityGML since it is easier for developers to implement CityJSON in their software, especially if that software is web-based. As an example, the CityJSON developers are not aware of any JavaScript code to parse CityGML files (the complete standard), while for CityJSON it would be simpler as JSON can be deemed as “native for the web.”

**Extensions to the core model**

It is possible to easily define Extensions to the core model. CityJSON uses JSON Schemas to document and validate the data model, schemas should be seen as basically validating the syntax of a JSON document.

A CityJSON Extension is a JSON file that allows documentation of how the core data model of CityJSON may be extended, and to validate CityJSON files containing new objects and/or attributes. This is conceptually akin to the Application Domain Extensions (ADEs) in CityGML; see Section 10.13 of the official CityGML documentation.

**Software**

Several software packages and libraries offer CityJSON support. The full list is at <https://www.cityjson.org/software/>.

It should be noticed that several programming languages have been used in implementations so far: Python, Java, C++, JavaScript, Ruby, Objective-C and Swift. This helps ensure that CityJSON is “implementable” for most developers, and not just within one environment.

**History of its development**

CityJSON was started, and is maintained, by the 3D geoinformation group at TU Delft (https://3d.bk.tudelft.nl). Others have since then joined its development, especially virtualcitySYSTEMS and Claus Nagel (https://www.virtualcitysystems.de).

The CityJSON developers invite anyone to contribute to the development and improvement of CityJSON, all discussions, issues, and developments are open to everyone on the GitHub repository of CityJSON: https://github.com/tudelft3d/cityjson

# Relationship to other OGC standards

*State whether this proposed Community standard has any dependencies on OGC standards or is itself normatively referenced by an OGC standard and list those standards, as applicable.*

While CityJSON itself is not officially standardized, it is an encoding for a subset of the OGC CityGML data model (version 2.0.0). The few features that are not currently supported are either because they are seldom used, or because they would overcomplicate and decrease the strengths of the JSON encoding; see <https://www.cityjson.org/citygml-compatibility/>.

There are also capabilities for bidirectional conversion between CityJSON and CityGML. Thus, using CityJSON means that you are using the CityGML data model.

# Alignment with OGC Standards Baseline

*Describe where this proposed standard fits with respect to the existing OGC standards baseline and standards in development in the OGC and whether this proposed standard may compete with or enhance an existing OGC standard*.

CityJSON is an encoding for a subset of the OGC CityGML data model (version 2.0.0). It offers an alternative encoding to the official OGC GML encoding of CityGML.

# Evidence of implementation

The following implementations use the proposed Community standard.

* an overview of the software are at <https://www.cityjson.org/software/>
* some scripts and examples of reading/writing/processing CityJSON in different languages are available at <https://github.com/tudelft3d/cityjson-software/>

**Implementation name**: 3dcitydb

**Date of most recent version**: 2019-08-06

**Implementation description**: The 3D City Database is a free 3D geo database to store, represent, and manage virtual 3D city models on top of a standard spatial relational database. The database model contains semantically rich, hierarchically structured, multi-scale urban objects facilitating complex GIS modeling and analysis tasks, far beyond visualization. In 2012, the 3D City Database received the Oracle Spatial Excellence Award for Education and Research. The schema of the 3D City Database is based on the OGC City Geography Markup Language (CityGML), an international standard for representing and exchanging virtual 3D city models issued by the Open Geospatial Consortium (OGC). It fully supports to read and write CityJSON.

**Implementation URL**: https://github.com/3dcitydb/3dcitydb/

**Is implementation complete**? Yes

**Implementation name**: azul

**Date of most recent version**: 2019-07-09

**Implementation description**: azul is a 3D viewer for macOS 10.14 and higher. It is intended for viewing 3D city models in CityGML 1.0 and 2.0, CityJSON 1.0, IndoorGML, OBJ, OFF and POLY. It supports loading multiple files, selecting objects by clicking them or selecting them in the sidebar, toggling the visibility of individual items, and browsing their attributes. It is currently pre-release software, but it is pretty stable and most datasets already work without problems. It is available under the GPLv3 licence. Written in Swift and C++.

**Implementation URL**: https://github.com/tudelft3d/azul

**Is implementation complete**? No

**If not, what portions of the proposed Community standard are implemented?**

Everything is supported except the textures, which will be added later this year.

**Implementation name**: citygml4j

**Date of most recent version**: 2019-08-11

**Implementation description**: citygml4j is an open source Java class library and API for facilitating work with the [OGC City Geography Markup Language (CityGML)](http://www.opengeospatial.org/standards/citygml). citygml4j makes it easy to read, process and write CityGML datasets, and to develop CityGML-aware software applications. Starting from version 2.6.0, citygml4j supports parsing and writing [CityJSON](http://www.cityjson.org/), a format for encoding a subset of the CityGML data model using JSON instead of GML. Written in Java.

**Implementation URL**: https://github.com/citygml4j/citygml4j

**Is implementation complete**? Yes

**Implementation name**: cjio

**Date of most recent version**: 2019-06-25

**Implementation description**: Python CLI to process and manipulate [CityJSON](http://www.cityjson.org) files. The different operators can be chained to perform several processing operations in one step, the CityJSON model goes through them and different versions of the CityJSON model can be saved as files along the pipeline. Written in Python.

**Implementation URL**: https://github.com/tudelft3d/cjio

**Is implementation complete**? Yes (all features of CityJSON are supported)

**Implementation name**: FME support

**Date of most recent version**: in beta

**Implementation description**: A Safe FME reader and a writer. This work is beta at this moment, Safe and TU Delft are working on it.. The code was started by Safe and is openly available, and CityJSON support will be in FME 2020 (in beta)..

**Implementation URL**: https://github.com/safesoftware/fme-CityJSON

**Is implementation complete**? No.

**If not, what portions of the proposed Community standard are implemented?**

The first release will support everything, except textures.

**Implementation name**: QGIS plugin

**Date of most recent version**: 2019-09-03

**Implementation description**: Python plugin for QGIS 3 which adds support for loading [CityJSON](http://www.cityjson.org) datasets in QGIS. Written in Python.

**Implementation URL**: https://github.com/tudelft3d/cityjson-qgis-plugin

**Is implementation complete**? No.

**If not, what portions of the proposed Community standard are implemented?**

Textures are not supported.

Writing of files not supported yet, only reading/importing.

**Implementation name**: WFS implementation by VCS

**Date of most recent version**: 2019-05-13

**Implementation description**: virtualcitySYSTEMS has a WFS interface for the 3D City Database that supports CityJSON as output format. For their 3D web map client, they just added the possibility to load and visualize CityJSON data directly in the browser, either by uploading files or by querying the database via the WFS.

**Implementation URL**: not an open implementation

**Is implementation complete**? Yes.

**Implementation name**: web-viewer

**Date of most recent version**: 2019-09-19

**Implementation description**: A simple web-viewer for CityJSON files: drag-and-drop a file, wait a few milliseconds, and you see it and can query its attributes. Acts as the “official” viewer for CityJSON and publicly available at https://viewer.cityjson.org/

**Implementation URL**: https://github.com/tudelft3d/cityjson-viewer

**Is implementation complete**? No.

**If not, what portions of the proposed Community standard are implemented?**

Textures are not supported.

Geometry templates are not supported.

**Implementation name**: NINJa

**Date of most recent version**: 2019-11-21

**Implementation description**: Web-viewer and online editor for CityJSON files. Files can be visualised, modified (their attributes and feature types) and then saved. The demo is publicly available at https://liberostelios.github.io/ninja

**Implementation URL**: https://github.com/liberostelios/ninja

**Is implementation complete**? No.

**If not, what portions of the proposed Community standard are implemented?**

At this moment, only textures are missing. But we are working on adding them.

**Implementation name**: *dataset: all buildings in the USA*

**Date of most recent version**: 2019-07-11

**Implementation description**: The project Open City Model has converted all buildings (about 125 millions) in the USA to LoD1 buildings and offers them in CityJSON.

**Implementation URL**: https://github.com/opencitymodel/opencitymodel

**Is implementation complete**? No. In the sense that the 3D buildings are not textured.

**Implementation name**: *dataset: all publicly available CityGML datasets*

**Date of most recent version**: -

**Implementation description**: With citygml4j, all CityGML datasets can be automatically converted and we are hosting a few examples on the official website of CityJSON. These contain all the features of CityJSON (textures, material, geometry templates, compression).

**Implementation URL**: https://www.cityjson.org/datasets/

**Is implementation complete**? Yes.

**Implementation name**: *dataset: 3D building open data of public housing buildings in Singapore*

**Date of most recent version**: 2019-08-25

**Implementation description**: The footprints in Singapore were used to create a 3D building model of public housing, and it is being distributed as open data. The model is available in CityJSON and OBJ.

**Implementation URL**: https://ual.sg/post/2019/08/25/release-of-3d-building-open-data-of-hdbs-in-singapore/

**Is implementation complete**? Yes

*Repeat for each implementation.*

*Optionally, provide a narrative description of the extent of implementation of the proposed Community standard for those proposed standards that are very widely used.*

# Public availability

Is the proposed Community standard currently publicly available? **Yes**

URL: https://cityjson.org

# Supporting member(s)

*List the supporting organizations. There must be at least three OGC organizations of which at least one must be an OGC Voting Member.*

1. Geonovum
2. Delft University of Technology
3. Kadaster International
4. virtualcitySYSTEMS
5. National University of Singapore
6. [Forum Virium Helsinki Oy](https://forumvirium.fi/)
7. Ordnance Survey

# Intellectual property rights

Will the contributor retain intellectual property rights? **Yes**

If yes, the contributor will be required to work with OGC staff to properly attribute the submitter’s intellectual property rights.

If no, the contributor will assign intellectual property rights to the OGC.

NOTE: The IP is in the public domain (CC0 license), so it will not be assigned to OGC.